Performing Computations on Hierarchically Shared Secrets

Giulia Traverso, Denise Demirel, Johannes Buchmann

TU Darmstadt (Germany)

Abstract. Hierarchical secret sharing schemes distribute a message to a set of shareholders with different reconstruction capabilities. In distributed storage systems, this is an important property because it allows to grant more reconstruction capability to better performing storage servers and vice versa. In particular, Tassa's conjunctive and disjunctive hierarchical secret sharing schemes are based on Birkhoff interpolation and perform equally well as Shamir's threshold secret sharing scheme. Thus, they are promising candidates for distributed storage systems. A key requirement is the possibility to perform function evaluations over shared data. However, practical algorithms supporting this have not been provided yet with respect to hierarchical secret sharing schemes. Aiming at closing this gap, in this work, we show how additions and multiplications of shares can be practically computed using Tassa's conjunctive and disjunctive hierarchical secret sharing schemes. Furthermore, we provide auditing procedures for operations on messages shared hierarchically, which allow to verify that functions on the shares have been performed correctly. We close this work with an evaluation of the correctness, security, and efficiency of the protocols we propose.

Keywords: hierarchical secret sharing, Birkhoff interpolation, verifiable secret sharing, auditing, multi-party computation, distributed storage systems, cloud computing.

1 Introduction

In this work, we provide procedures allowing to evaluate functions on shares that have been generated by using a hierarchical secret sharing scheme. The primary focus of this paper is the application of secret sharing to the model of distributed storage systems. In distributed storage systems [24], shares of a document are generated and distributed to storage servers owned by multiple storage providers. As well as for cloud computing, means to measure the performance and the quality of service offered by storage providers is needed (for more details see recommendations by NIST [22] about the introduction of trust in cloud computing). This enables users to grant more reconstruction capability to the better performing storage servers, i.e. to the storage providers offering a better service, and vice versa. The problem is to find the most suitable secret sharing schemes to be applied to distributed storage systems. We argue that these are Tassa's conjunctive and disjunctive hierarchical secret sharing schemes [33] rather than the commonly used Shamir's threshold secret sharing scheme [31]. In fact, Tassa's conjunctive and disjunctive hierarchical secret sharing schemes allow for the same features as Shamir's threshold secret sharing scheme. Furthermore, Tassa's schemes achieve also optimal storage consumption and arranges the shares such that none of the storage providers has enough information to break the confidentiality of the document. More precisely, regarding storage consumption, on the one hand, Tassa's conjunctive and disjunctive hierarchical secret sharing schemes generate shares with different reconstruction capabilities but that have all the same length. On the other hand, Shamir's threshold secret sharing scheme generates shares that are all equivalent in their reconstruction capability. In this case, granting more reconstruction capability to the better performing storage servers means to overload them with multiple shares to manage. Thus, more shares than the amount of storage servers have to be generated and, overall, more storage space is consumed. Regarding confidentiality, when Shamir's threshold secret sharing scheme is used, the number of storage providers deployed must be larger than the reconstructing threshold. Otherwise, storage providers would have enough shares to retrieve the data within the storage servers they own. On the contrary, Tassa's conjunctive and disjunctive hierarchical secret sharing schemes allow for more flexibility because the reconstruction capability of the storage servers can be arranged such that no storage provider has enough information to retrieve the document. Clearly, if users can rely on less storage providers for a given reconstructing threshold, then this allow for a better trade-off between data protection and storage cost.

This paper shows how to practically compute additions and multiplications over hierarchically shared data when Tassa's conjunctive and disjunctive secret sharing schemes are used. So far solutions to perform operations on shared messages have only been instantiated for Shamir's threshold secret sharing schemes and have been generalized for any linear secret sharing scheme in [10]. Thus, in this work we fill this gap by introducing procedures allowing to evaluate functions on shares that have been generated using a hierarchical secret sharing scheme. More precisely, we show how to perform linear operations and multiplications on messages that have been shared and need to be reconstructed using the Birkhoff interpolation formula. Tassa's conjunctive and disjunctive hierarchical secret sharing schemes are based on Birkhoff interpolation and are linear schemes. Thus, we adapt to our setting the general procedure for computations over linear secret sharing schemes introduced in [10]. This is not trivial because in practice multiplications are split in a preprocessing and an on-line phase which both have to be adapted to the hierarchical setup. Furthermore, we prove that these procedures compute the outcome of functions correctly and provide perfect secrecy, i.e. only qualified subsets are able to retrieve the input messages and the result of computations. Moreover, we provide an audit procedure for computations over messages shared hierarchically. Lastly, we discuss security and efficiency of the algorithms introduced. The rest of the paper is organized as follows. First, the related work and the preliminaries are discussed in Section 2 and Section 3, respectively. Then, our contribution is presented. More precisely, in Section 4 it is shown how to perform operations over messages shared using Birkhoff interpolation-based hierarchical secret sharing schemes. Furthermore, it is proven that those procedures compute the outcome of functions correctly and provide perfect secrecy. Since multiplications on hierarchical shares require larger changes on the preprocessing phase we detail this process in Section 5. In Section 6, it is described how to perform the auditing procedure. Security and efficiency of the proposed protocols are discussed in Section 7. Conclusions can be found in Section 8.

2 Related Work

Independently introduced by Shamir [31] and Blakley [4], threshold secret sharing is a cryptographic primitive enabling a dealer to share a message equally among a set of players, called shareholders. The message can be reconstructed by subsets of a certain amount of shareholders while subsets smaller than the threshold do not learn any information about the data shared. The best known and most widely used secret sharing scheme is Shamir's threshold secret sharing, introduced in 1979 in [31]. His solution is based on polynomials and on Lagrange interpolation. In addition, extensive research has been done with respect to Shamir's threshold secret sharing showing very desirable features from a practical point of view. For instance, it has been shown in [21] how shares can be periodically refreshed to ensure long-term protection of the shared message. Based on the general approach of [12], it has been shown in [19] how to modify the definition of eligible subsets of shareholders for the reconstruction of the message. In [26], it has been shown how to add shareholders. Furthermore, it is possible to perform operations over shared messages, provided that the set of shareholders and the threshold remain the same for all the shared messages, as shown in [18]. This enables general multi-party computation, as discussed in [3], [8], and [16]. Furthermore, in [30] it is shown how to perform an auditing procedure for computations over shared messages, which is based on the work done in [1] and in [11]. So called hierarchical secret sharing schemes [14] address scenarios where the shareholders are not equal in their reconstruction capability. This allows, for instance, to reflect the hierarchical structure of companies. Furthermore, they can be used to realize social secret sharing schemes in cloud storage solutions (see e.g. [25], [35]). Here the best performing storage servers are treated as the most powerful shareholders. A preliminary notion of hierarchical secret sharing has been discussed by Shamir already in his seminal work [31]. However, this approach overloads the most powerful shareholders. Brickell in [7] and Simmons in [32] presented a solution where each shareholder receives only one share which is of equal size, but with different reconstruction capabilities. However, the reconstruction process is highly expensive. Ghodosi et al. showed in [17] how to construct efficient schemes for specific instantiations. In 2007 Tassa solved these problems by introducing in [33] two schemes based on polynomials and Birkhoff interpolation (a generalization of Lagrange interpolation) for the reconstruction of the message. These are called conjunctive and disjunctive hierarchical secret sharing schemes, depending on which subsets of shareholders are eligible to access the shared message. It has been proven in [34] that Tassa's conjunctive and disjunctive hierarchical secret sharing schemes achieve the same flexibility as Shamir's threshold secret sharing scheme. More precisely, algorithms based on Birkhoff interpolation have been designed that allow Tassa's schemes to add shareholders, to periodically refresh the shares, and to modify the definition of eligible subsets for the reconstruction of the message. According to the notion of dynamic secret sharing[1] specified in [34], both Shamir's and Tassa's secret sharing schemes are dynamic. The last step to show that hierarchical secret sharing achieves the same functionalities as Shamir's threshold secret sharing is to prove that Tassa's schemes support the same operations over shared messages, i.e. linear combinations and multiplication. Conditions on the access structure allowing for multiplication have been investigated in [23]. However, they lead to schemes with either an increased length of the shares (which is not optimal for our application to distributed storage systems) or with stronger conditions on the access structure deviating from the original schemes proposed by Tassa. Furthermore, practical and ready to be used algorithms for linear operations and multiplications over Tassa's conjunctive and disjunctive hierarchical secret sharing schemes and for performing auditing over such computations were not proposed. And this is what we provide in this work.

3 Preliminaries

This section provides preliminaries with respect to secret sharing and explains in details how Tassa's conjunctive and disjunctive hierarchical secret sharing schemes are defined.

3.1 Threshold Secret Sharing Schemes

Secret sharing schemes are used to share a message $m \in \mathbb{F}_q$ across a set $S = \{s_1, \dots, s_n\}$ of n shareholders. More precisely, a dealer generates shares $\sigma_1, \dots, \sigma_n \in \mathbb{F}_q$ of message m and distributes each share $\sigma_i \in \mathbb{F}_q$ to the respective shareholder $s_i \in S$. Only specific subsets $A \subset S$ of shareholders can reconstruct the message provided that certain requirements are fulfilled. Instead, subsets $U \subset S$ not fulfilling such requirements cannot reconstruct the message and get no information about it. These subsets are called *authorized* and *unauthorized*, respectively. Denoted by $\mathcal{P}(S)$ the partition of set S, the *access structure* $\mathit{\Gamma} \subset \mathcal{P}(S)$ determines both sets, i.e. $A \in \mathit{\Gamma}$ and $U \notin \mathit{\Gamma}$. More formally, secret sharing is a pair of algorithms $(\mathsf{Share}, \mathsf{Reconstruct})$. Algorithm Share takes as input a message $m \in \mathbb{F}_q$ and the unique ID $i \in \mathcal{I}$ of shareholder $s_i \in S$ and outputs its share $\sigma_i \in \mathbb{F}_q$, for $i = 1, \dots, n$. Algorithm $\mathsf{Reconstruct}$ takes as input a subset $R \subset S$ and outputs

[1] Note that this is different from the notion of fully dynamic secret sharing discussed in [5], where one scheme supports different access structures for different secrets.

the message m if $R \in \Gamma$ and $\perp$ otherwise. Adapting the definition provided in [2] to the purpose of this paper, in the following we formalize the notions of correctness and perfect secrecy.

Definition 1. *Given the set $\mathbb{F}_q$ of messages and the set $S = \{s_1, \dots, s_n\}$ of shareholders, the pair of algorithms* $(\mathsf{Share}, \mathsf{Reconstruct})$ *is a secret sharing scheme realizing access structure $\Gamma \subset \mathcal{P}(S)$ if the following two requirements hold.*

1) Correctness*: if shares held by shareholders of an authorized set $A \in \Gamma$ are given as input to algorithm* Reconstruct*, then algorithm* Reconstruct *retrieves the message m shared during algorithm* Share*, for every message $m \in \mathbb{F}_q$.*
2) Perfect secrecy*: if shares held by shareholders of an unauthorized set $U \notin \Gamma$ are given as input to algorithm* Reconstruct*, then algorithm* Reconstruct *leaks no information about the message m shared during algorithm* Share*, for every message $m \in \mathbb{F}_q$.*

Linear threshold secret sharing schemes are amongst the most studied schemes also due to their usage in practical scenarios. The first (t, n)-threshold secret sharing scheme is proposed by Shamir in [31] and it is based on interpolation of polynomials. More precisely, a message m is shared using a polynomial $f(x) = a_0 + a_1 x + \dots + a_{t-1} x^{t-1}$ of degree $\deg(f(x)) = t - 1$, where $a_0 := m$ and coefficients $a_1, \dots, a_{t-1} \in \mathbb{F}_q$ are chosen uniformly at random. Algorithm Share computes share $\sigma_i \in \mathbb{F}_q$ for shareholder $s_i \in S$ as a point on polynomial $f(x)$, i.e. $\sigma_i := f(i)$, where $i \in \mathcal{I}$ is the ID of shareholder s_i. Algorithm Reconstruct is based on Lagrange interpolation of polynomials. Thus, on the one hand, authorized subsets $A \subset S$ are composed of t or more shareholders, that is $|A| \geq t$. In fact, when t or more points of polynomial $f(x)$ are collected, it is possible to correctly interpolate polynomial $f(x)$ and message m is retrieved as $f(0) = a_0$. On the other hand, unauthorized subsets $U \subset S$ are composed of $t - 1$ or less shareholders, that is $|U| \leq t - 1$. In fact, when only $t - 1$ or less points are collected, polynomial $f(x)$ cannot be reconstructed and no information about message $m \in \mathbb{F}_q$ is leaked.

3.2 Hierarchical Secret Sharing Schemes and Birkhoff Interpolation Problem

The so called *conjunctive* and *disjunctive* schemes proposed by Tassa in [33] are the first hierarchical secret sharing schemes based on Birkhoff interpolation of polynomials. More precisely, shares are either points on a polynomial or points on one of the derivatives of such polynomial. More precisely, a hierarchy is composed of levels $L_0, \dots, L_\ell$, where L_0 is the highest level, L_ℓ the lowest, and $\ell \leq n$. The cardinality of each level L_h is denoted by n_h and each shareholder is assigned to one level only. In addition, a threshold t_h is associated with each level L_h, for $h \in 0, \dots, \ell$, such that $0 < t_0 < \dots < t_\ell$. Tassa individuated two types of access structures, defining, respectively the conjunctive and the disjunctive hierarchical secret sharing. On the one hand, the conjunctive access structure determines that a subset $A \subset S$ is authorized if, *for all levels* L_h, it contains t_h shareholders assigned to levels equal or higher than L_h, for $h = 0, \ldots, \ell$. On the other hand, the disjunctive access structure specifies that a subset $A \subset S$ is authorized if, *for at least one level* L_h, it contains t_h shareholders assigned to levels equal or higher than L_h, for $h = 0, \ldots, \ell$. In the following, we write information relating to disjunctive hierarchical secret sharing in brackets. For conjunctive (disjunctive) hierarchical secret sharing schemes the unique ID of shareholder $s_{i,j} \in S$ is a pair $(i, j) \in \mathcal{I} \times \mathcal{I}$, where $i = 1, \ldots, n_h$ and $j := t_{h-1}$ $(j := t_\ell - t_h)$, for $h = 0, \ldots, \ell$ with $t_{-1} := 0$. The algorithms Share and $\mathsf{Reconstruct}$ of the conjunctive (disjunctive) hierarchical secret sharing are as follows.

Share The algorithm takes as input a message $m \in \mathbb{F}_q$ and generates a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ of degree $\deg(f(x)) = t-1$, where $a_0 := m$ $(a_{t-1} := m)$ and the coefficients $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ $(a_0, \ldots, a_{t-2} \in \mathbb{F}_q)$ are chosen uniformly at random. It outputs share $\sigma_{i,j} \in \mathbb{F}_q$ for shareholder $s_{i,j} \in S$ computed as $\sigma_{i,j} := f^j(i)$, where $f^j(x)$ is the j-th derivative of polynomial $f(x)$ and pair $(i, j) \in \mathcal{I} \times \mathcal{I}$ is the unique ID of shareholder $s_{i,j} \in S$, for $i = 1, \ldots, n_h$ and $h = 0, \ldots, \ell$.

$\mathsf{Reconstruct}$ The algorithm takes as input a set of shares held by a subset $R \subset S$ of shareholders. If R is unauthorized, i.e. $R \notin \Gamma$, then it outputs $\perp$. If R is authorized, i.e. $R \in \Gamma$, then it reconstructs polynomial $f(x)$ using Birkhoff interpolation and outputs $m = a_0$ $(m = a_{t-1})$.

The Birkhoff interpolation problem is a generalization of the Lagrange interpolation problem and describes the problem of finding a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ satisfying the equalities $f^j(i) = \sigma_{i,j}$. Given an authorized set $R \in \Gamma$ of shareholders for conjunctive (disjunctive) hierarchical secret sharing schemes, the Birkhoff interpolation problem can be solved as follows. The *interpolation matrix* associated to set R is a binary matrix E where entry $e_{i,j}$ is set to '1' if shareholder $s_{i,j}$ participates with share $\sigma_{i,j}$ and '0' otherwise. Let us denote by $I(E) = \{(i, j)$ such that $e_{i,j} = 1\}$ the set containing the entries of E in lexicographic order, i.e. the pair (i, j) precedes the pair (i', j') if and only if $i < i'$ or $i = i'$ and $j < j'$. The elements of $I(E)$ are denoted by $(i_1, j_1), (i_2, j_2), \ldots, (i_r, j_r)$, where $r := |R|$. Furthermore, we set $\varphi := \{\phi_0, \phi_1, \phi_2, \ldots, \phi_{t-1}\} = \{1, x, x^2, \ldots, x^t\}$ and denote by ϕ_k^j the j-the derivative of ϕ_k, for $k = 0, \ldots, t-1$. Then the matrix $A(E, X, \varphi)$ is defined as follows:

$$A(E, X, \varphi) = \begin{pmatrix} \phi_0^{j_1}(i_1) & \phi_1^{j_1}(i_1) & \phi_2^{j_1}(i_1) & \cdots & \phi_{t-1}^{j_1}(i_1) \\ \phi_0^{j_2}(i_2) & \phi_1^{j_2}(i_2) & \phi_2^{j_2}(i_2) & \cdots & \phi_{t-1}^{j_2}(i_2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \phi_0^{j_r}(i_r) & \phi_1^{j_r}(i_r) & \phi_2^{j_r}(i_r) & \cdots & \phi_{t-1}^{j_r}(i_r) \end{pmatrix}.$$

Polynomial $f(x)$ can be reconstructed in distributed fashion by computing

$$f(x) = \sum_{k=0}^{t-1} a_k x^k = \sum_{k=0}^{t-1} \sum_{l=1}^{r} a_{l,k} x^k,$$

where $a_{l,k} := \sigma_{i_l,j_l}(-1)^{l-1+k} \frac{\det(A_{l-1,k}(E,X,\varphi))}{\det(A(E,X,\varphi))}$ is computed locally by shareholder $s_{i_l,j_l} \in R$, for $l = 1, \ldots, r$, and matrix $A_{l-1,k}(E, X, \varphi)$ results from matrix $A(E, X, \varphi)$ by removing the l-th row and the $(k+1)$-th column (see [34], Theorem 1 for a formal proof). Appendix A discusses the necessary and sufficient requirements for Birkhoff interpolation problem to have a unique solution. Examples of Birkhoff interpolation problems can be found in [27].

4 Operations on Messages Distributed through Hierarchical Secret Sharing Schemes

In this section, we prove that Tassa's conjunctive and disjunctive hierarchical secret sharing schemes, based on Birkhoff interpolation, allow to perform operations over shared messages. More precisely, a message can be reconstructed which is the result of operations performed over previously shared messages. The operations supported are the sum of messages, the multiplication of a message by a scalar, and the product of messages.

4.1 Setting

Messages $m_1, m_2 \in \mathbb{F}_q$ are distributed to a set S of n shareholders according to the following assumptions.

(A1) The underlying access structure Γ remains the same for both messages m_1, m_2. More precisely, both polynomials $f(x)$ and $h(x)$ used to share m_1 and m_2, respectively, have the same degree. Furthermore, shareholder $s_{i,j}$ with unique ID (i, j) holds share $\sigma_{i,j}(m_1) := f^j(i)$ and $\sigma_{i,j}(m_2) := h^j(i)$.

(A2) The degree $t - 1$ of polynomials $f(x)$ and $h(x)$ is chosen such that $2t \leq n$, where n is the total number of shareholders.

(A3) The ID (i, j) of each shareholder $s_{i,j} \in S$ is chosen such that index $i \in \mathcal{I}$ is selected once within the whole hierarchy and such that the corresponding Birkhoff interpolation problem has a unique solution. The requirements to achieve this can be found in Appendix A.

(A4) The user communicates with the shareholders and the shareholders among each other using private channels.

(A5) A tamper-proof bulletin board is available to allow exchanging data during the preprocessing phase of the multiplication procedure. Note that this is a common assumption for auditable multi-party computation and a more formal definition can be found in [20].

Let us recall that index $j \in \mathcal{I}$ of the unique identity ID of shareholder $s_{i,j} \in S$ is defined as $j := t_{h-1}$ ($j := t_\ell - t_h$), for $h = 0, \ldots, \ell$ and $t_{-1} := 0$ (see Section 3). Algorithm Share defined in Section 3.2 is run separately to share and distribute message m_1 and message m_2 to the n shareholders of set S. More precisely, to share message m_1, algorithm Share selects a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$, where $a_0 := m_1$ ($a_{t-1} := m_1$) and $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ ($a_0, \ldots, a_{t-2} \in \mathbb{F}_q$) are chosen uniformly at random. It distributes to each shareholder $s_{i,j} \in S$ share $\sigma_{i,j}(m_1) = f^j(i)$. To share message m_2, algorithm Share generates a polynomial $h(x) = b_0 + b_1 x + \cdots + b_{t-1} x^{t-1}$, where $b_0 := m_2$ ($b_{t-1} := m_2$) and $b_1, \ldots, b_{t-1} \in \mathbb{F}_q$ ($b_0, \ldots, b_{t-2} \in \mathbb{F}_q$) are chosen uniformly at random. It distributes to each shareholder $s_{i,j} \in S$ share $\sigma_{i,j}(m_2) = h^j(i)$. Afterwards, algorithms Linear and Multiply are run by each shareholder individually to perform linear operations and multiplications on their shares of messages m_1 and m_2. Finally, the result $m \in \mathbb{F}_q$ of these operations on m_1, m_2 can be reconstructed by running algorithm Reconstruct defined in Section 3.2 on the shares computed by each shareholder.

4.2 Linear Operations

In this section, algorithm Linear is presented, which computes share $\sigma_{i,j}(m) \in \mathbb{F}_q$ for shareholder $s_{i,j} \in S$, to be used as input for algorithm Reconstruct to retrieve message $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$, for scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$.

Linear The algorithm takes as input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2) \in \mathbb{F}_q$ held by shareholder $s_{i,j} \in S$, and scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$. It outputs share $\sigma_{i,j}(m) := \lambda_1 \cdot \sigma_{i,j}(m_1) + \lambda_2 \cdot \sigma_{i,j}(m_2) \in \mathbb{F}_q$ for shareholder $s_{i,j} \in S$.

Theorem 1. *The algorithm* Linear *for conjunctive (disjunctive) hierarchical secret sharing introduced above computes the shares correctly. More precisely, on input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2)$ and scalars λ_1, λ_2, the shares computed by* Linear *reconstruct to message m, where $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$. Furthermore, perfectly secrecy, according to Definition 1, is maintained while performing* Linear.

Proof. Let $\sigma_{i,j}(m) \in \mathbb{F}_q$ be the shares computed by shareholders $s_{i,j} \in R$ using algorithm Linear, where $R \in \Gamma$ is an authorized set. To prove correctness, we have to show that algorithm Reconstruct outputs message $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$ when it takes as input shares $\sigma_{i,j}(m) \in \mathbb{F}_q$. More precisely, we have to show that the shares interpolate to a polynomial $p(x) = c_0 + c_1 x + \cdots + c_{t-1} x^{t-1}$ of degree $\deg(p(x)) = t - 1$, where $c_0 = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2 (c_{t-1} = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2)$. To prove perfect secrecy, we have to show, first, that algorithm Linear computes shares for message $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$ without leaking information about the shares for message m_1 and message m_2. Second, we have to show that any unauthorized set $U \notin \Gamma$ gets no information about $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$. In order to do that, we have to show that polynomial $p(x) = c_0 + c_1 x + \cdots + c_{t-1} x^{t-1}$ can be computed in distributed fashion by each shareholder $s_{i,j} \in R$. That is, correctness and perfect secrecy hold if each shareholder can compute a term $p_{(i,j),k}$ without leaking information to any other shareholder and such that:

$$p(x) = \sum_{k=0}^{t-1} c_k x^k = \sum_{k=0}^{t-1} \sum_{s_{i,j} \in R} p_{(i,j),k} x^k,$$

where $c_0 = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2 (c_{t-1} = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2)$.

Let us recall that message $m_1 \in \mathbb{F}_q$ is shared using polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$. Due to Birkhoff interpolation resolution formula (see Section 3.2), coefficient a_k of polynomial $f(x)$ can be computed as:

$$a_k = \sum_{l=1}^{r} a_{l,k} = \sum_{l=1}^{r} \sigma_l(m_1)(-1)^{l-1+k} \frac{\det(A_{l-1,k}(E,X,\varphi))}{\det(A(E,X,\varphi))},$$

for $k = 0, \ldots, t-1$, where $\sigma_l(m_1)$, for $l = 1, \ldots, r$, are the shares $\sigma_{i,j}(m_1)$ in lexicographic order ((i,j) precedes the pair (i',j') if $i < i'$ or $i = i'$ and $j < j'$). Similarly, message $m_2 \in \mathbb{F}_q$ is shared through polynomial $h(x) = b_0 + b_1 x + \cdots + b_{t-1} x^{t-1}$. Due to Birkhoff interpolation resolution formula, coefficient b_k of polynomial $h(x)$ can be computed as:

$$b_k = \sum_{l=1}^{r} b_{l,k} = \sum_{l=1}^{r} \sigma_l(m_2)(-1)^{l-1+k} \frac{\det(A_{l-1,k}(E,X,\varphi))}{\det(A(E,X,\varphi))},$$

for $k = 0, \ldots, t-1$, where $\sigma_l(m_2)$, for $l = 1, \ldots, r$, are the shares $\sigma_{i,j}(m_2)$ in lexicographic order. Because of the homomorphic property of polynomials, polynomial $p(x)$ can be computed as the linear combination of polynomial $f(x)$ and polynomial $h(x)$ with scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$. That is, $p(x) = \lambda_1 \cdot f(x) + \lambda_2 \cdot h(x)$. Therefore,

$$p(x) = \sum_{k=0}^{t-1} \lambda_1 \cdot a_k + \lambda_2 \cdot b_k = \sum_{k=0}^{t-1} \sum_{l=1}^{r} \lambda_1 \cdot a_{l,k} + \lambda_2 \cdot b_{l,k}.$$

This shows that the terms $p_{l,k} = p_{(i,j),k} := \lambda_1 \cdot a_{l,k} + \lambda_2 \cdot b_{l,k}$ computed by the shareholders $s_{i,j} \in R$ interpolate to polynomial $p(x)$ and correctness is provided. Regarding perfect secrecy, the computation of $p_{l,k}$ is performed solely by shareholder $s_l \in R$ using the information it has and without leaking $a_{l,k}$ nor $b_{l,k}$. Thus, no information about shares $\sigma_l(m_1), \sigma_l(m_2)$ is leaked. Moreover, being polynomial $p(x)$ of degree $\deg(p(x)) = t-1$, the original access structure Γ is maintained: subsets $U \subset S$ of shareholders such that $U \notin \Gamma$ not only cannot reconstruct $m = \lambda_1 \cdot m_1 + \lambda_2 \cdot m_2$, but also do not get any information about m_1 nor m_2. Thus, perfect secrecy of the underlying conjunctive (disjunctive) hierarchical secret sharing is still maintained even if algorithm Linear is run and the shares computed by this algorithm are used as input for algorithm Reconstruct.

4.3 Multiplication

In this section, algorithm Multiply is presented, which computes share $\sigma_{i,j}(m)$ for shareholder $s_{i,j} \in S$. Share $\sigma_{i,j}(m)$ is used as input for algorithm Reconstruct to retrieve message $m = m_1 \cdot m_2$. Algorithm Multiply uses algorithm Linear (see Section 4.2) to compute message m as linear combinations of the shares for message m_1 and message m_2. More precisely, it builds on the multiplication algorithm discussed in [30], requiring for each multiplication a preprocessing phase in which the shareholders jointly compute shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma)$ to messages $\alpha, \beta, \gamma \in \mathbb{F}_q$ such that $\alpha \cdot \beta = \gamma$. Note that, according to Assumption (A1) in Section 4.1, for algorithm Multiply to work the values α, β, and γ have to be shared according to the access structure Γ. More details about how to achieve this are provided in Section 5.

Multiply The algorithm selects a triple (α, β, γ) generated during the preprocessing phase and it takes as input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2) \in \mathbb{F}_q$ and shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma) \in \mathbb{F}_q$ held by shareholder $s_{i,j} \in S$. It outputs share $\sigma_{i,j}(m) \in \mathbb{F}_q$ for message $m = m_1 \cdot m_2$, which is computed performing the following steps.

First, shareholder $s_{i,j}$ computes share $\sigma_{i,j}(\delta) := \sigma_{i,j}(m_1) - \sigma_{i,j}(\alpha)$ and share $\sigma_{i,j}(\varepsilon) := \sigma_{i,j}(m_2) - \sigma_{i,j}(\beta)$ using algorithm Linear. Second, shareholders from an authorized set $R \in \Gamma$ run algorithm Reconstruct with shares $\sigma_{i,j}(\delta), \sigma_{i,j}(\varepsilon)$ as input to publicly reconstruct values δ, ε using the bulletin board. Third, shareholder $s_{i,j} \in S$ computes the share $\sigma_{i,j}(m) := \sigma_{i,j}(\gamma) + \varepsilon \cdot \sigma_{i,j}(m_1) + \delta \cdot \sigma_{i,j}(m_2) - \delta\varepsilon$ using algorithm Linear.

Theorem 2. *The algorithm* Multiply *for conjunctive (disjunctive) hierarchical secret sharing introduced above computes the shares correctly. More precisely, on input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2)$, the shares computed by* Multiply *reconstruct to message m, where $m = m_1 \cdot m_2$. Furthermore, perfect secrecy, according to Definition 1, is maintained while performing* Multiply*.*

Proof. The correctness relies on the correctness of algorithm Linear, presented in Section 4.2. In fact, share $\sigma_{i,j}(m)$ is defined as the linear combination of shares $\sigma_{i,j}(\gamma), \sigma_{i,j}(m_1), \sigma_{i,j}(m_2)$ for messages γ, m_1, m_2, respectively, and scalars δ, ε. More precisely, in the first step the scalars δ and ε are computed in distributed fashion using algorithm Linear, such that $\delta = m_1 - \alpha$ and $\varepsilon = m_2 - \beta$. After those values have been reconstructed in the second step, in the third step each shareholder computes a share to message m by computing $\sigma_{i,j}(m) = \sigma_{i,j}(\gamma) + \varepsilon \cdot \sigma_{i,j}(m_1) + \delta \cdot \sigma_{i,j}(m_2) - \delta\varepsilon$ using algorithm Linear. Therefore, if algorithm Reconstruct takes as input shares $\sigma_{i,j}(m) \in \mathbb{F}_q$ held by shareholders $s_{i,j} \in R$, where $R \in \Gamma$ is an authorized set, then it retrieves:

$$m = \gamma + \varepsilon \cdot m_1 + \delta \cdot m_2 - \delta\varepsilon$$

$$= \gamma + (m_2 - \beta) \cdot m_1 + (m_1 - \alpha) \cdot m_2 - (m_2 - \beta)(m_1 - \alpha)$$

$$= \gamma + m_1 \cdot m_2 - \beta \cdot \alpha$$

Since $\alpha \cdot \beta = \gamma$ this leads to

$$m = m_1 \cdot m_2,$$

showing that algorithm Multiply is correct. Thus, algorithm Reconstruct interpolates to a polynomial $p(x) = c_0 + c_1 x + \cdots + c_{t-1} x^{t-1}$ of degree $\deg(p(x)) = t - 1$ and retrieves message $m_1 \cdot m_2$ as $c_0(c_{t-1})$. The perfect secrecy of algorithm Multiply is implied by the perfect secrecy of algorithm Linear (proven in Section 4.2) and by the perfect secrecy of the preprocessing phase, which is discussed in Section 5.

5 Preprocessing Phase

In this section, we introduce the preprocessing phase enabling the multiplication between two shared messages (see Section 4.3). Preprocessing has been common practice for multi-party computation since it has been introduced by Beaver in [1], because it lowers the communication complexity of the algorithm Multiply. More precisely, during the preprocessing phase a triple (α, β, γ) is generated such that the following conditions hold.

- $\alpha \cdot \beta = \gamma$.
- Assumption (1) of Section 4.1 holds, i.e. each shareholder $s_{i,j} \in S$ with ID $(i, j) \in \mathcal{I} \times \mathcal{I}$ holds shares $\sigma_{i,j}(\alpha) := f_\alpha^j(i), \sigma_{i,j}(\beta) := f_\beta^j(i)$, and $\sigma_{i,j}(\gamma) := f_\gamma^j(i)$, where $f_\alpha(x), f_\beta(x)$, and $f_\gamma(x)$ are the polynomials of degree $t - 1$ sharing α, β, and γ, respectively.

In [11] it is shown how to generate such triples, but it is assumed that Shamir's threshold secret sharing scheme is used. Thus, here we present a preprocessing phase for Tassa's conjunctive (disjunctive) hierarchical secret sharing scheme.

PreMult The algorithm outputs for each shareholder $s_{i,j} \in S$ a triple of shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma) \in \mathbb{F}_q$, such that for each triple it holds that $\sigma_{i,j}(\gamma) = \sigma_{i,j}(\alpha\beta)$. This is done in three main steps.

First, each shareholder $s_{i,j}$ randomly chooses a pair of shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$, as shown in Appendix B.1. Second, shareholders $s_1, \ldots, s_r \in R$ from an authorized set $R \in \Gamma$ compute for each shareholder $s_{i,j}$ terms $\delta_{l,(i,j)}$ and $\varepsilon_{l,(i,j)}$. Third, using $\delta_{l,(i,j)}$ and $\varepsilon_{l,(i,j)}$ each shareholder $s_{i,j} \in S$ computes its share $\sigma_{i,j}(\gamma) \in \mathbb{F}_q$.

More precisely, in the second step each shareholder $s_l \in R$, for $l = 1, \ldots, r$, computes the input $\delta_{l,(i,j)}$ and $\varepsilon_{l,(i,j)}$ for $s_{i,j}$ by performing the following steps.

First, shareholder $s_l \in R$ uses its shares $\sigma_l(\alpha)$ and $\sigma_l(\beta)$ and the unique ID (i, j) of shareholder $s_{i,j}$ to compute the values $\lambda_{l,(i,j)}$ and $\mu_{l,(i,j)}$ defined as:

$$\lambda_{l,(i,j)} := \sigma_l(\alpha) \sum_{k=j-1}^{t-1} \frac{k!}{(k-j+1)!} (-1)^{l-1+k} \frac{\det(A_{l-1,k}(E, X, \varphi))}{\det(A(E, X, \varphi))} i^{k-j+1}$$

and

$$\mu_{l,(i,j)} := \sigma_l(\beta) \sum_{k=j-1}^{t-1} \frac{k!}{(k-j+1)!} (-1)^{l-1+k} \frac{\det(A_{l-1,k}(E, X, \varphi))}{\det(A(E, X, \varphi))} i^{k-j+1},$$

where $A(E, X, \varphi)$ and $A_{l-1,k}(E, X, \varphi)$ are the matrices defined in Section 3. Then, it randomly splits $\lambda_{l,(i,j)}$ and $\mu_{l,(i,j)}$ into r values, i.e. $\lambda_{l,(i,j)} = \lambda_{1,l,(i,j)} + \cdots + \lambda_{r,l,(i,j)}$ and $\mu_{l,(i,j)} = \mu_{1,l,(i,j)} + \cdots + \mu_{r,l,(i,j)}$ and sends $\lambda_{m,l,(i,j)}$ and $\mu_{m,l,(i,j)}$ to shareholder $s_m \in R$, for $m = 1, \ldots, r$ and $m \neq l$, using a private channel. Afterwards, it collects all values $\lambda_{l,m,(i,j)}$ and $\mu_{l,m,(i,j)}$ received from shareholder $s_m \in R$, for $m = 1, \ldots, r$ and $m \neq l$, and computes $\delta_{l,(i,j)} := \sum_{m=1}^{r} \lambda_{l,m,(i,j)}$ and $\varepsilon_{l,(i,j)} := \sum_{m=1}^{r} \mu_{l,m,(i,j)}$. Finally, it sends $\delta_{l,(i,j)}$ and $\varepsilon_{l,(i,j)}$ to shareholder $s_{i,j}$ using a private channel.

In the third step, all shareholders within the set S compute their shares. More precisely, each shareholder $s_{i,j} \in S$ computes share $\sigma_{i,j}(\gamma)$ using the values $\delta_{l,(i,j)}$ and $\varepsilon_{l,(i,j)}$ received from shareholder $s_l \in R$, for $l = 1, \ldots, r$, as

$$\sigma_{i,j}(\gamma) := \sigma_{i,j}(\alpha\beta) = \left(\sum_{l=1}^{r} \delta_{l,(i,j)} \right) \cdot \sigma_{i,j}(\beta) + \sigma_{i,j}(\alpha) \cdot \left(\sum_{l=1}^{r} \varepsilon_{l,(i,j)} \right).$$

Theorem 3. *The algorithm* PreMult *for conjunctive (disjunctive) hierarchical secret sharing introduced above computes the multiplicative triples correctly. More precisely, on input the shares $\sigma_{i,j}(\alpha)$ and $\sigma_{i,j}(\beta)$, the shares computed by algorithm* PreMult *reconstructs to γ, where $\gamma = \alpha\beta$. Furthermore, perfect secrecy, according to Definition 1, is maintained while performing* PreMult*.*

Proof. Let $\sigma_{i,j}(\alpha\beta)$ be the share computed by shareholder $s_{i,j} \in R$ using algorithm PreMult, where $R \in \Gamma$ is an authorized set. Correctness of algorithm PreMult is provided if the shares held by shareholders in R it outputs interpolate to a polynomial $p(x) = c_0 + c_1 x + \cdots + c_{2(t-1)} x^{2(t-1)}$, where $c_0 = \alpha\beta (c_{2(t-1)} = \alpha\beta)$. Polynomial $p(x)$ is defined as $p(x) = f_\alpha(x) \cdot f_\beta(x)$, given that α is shared using polynomial $f_\alpha(x)$ and β is shared using polynomial $f_\beta(x)$. We have to show that, for each share $\sigma_{i,j}(\gamma)$ computed by algorithm PreMult, it holds that $\sigma_{i,j}(\gamma) = \sigma_{i,j}(\alpha\beta)$, where $\sigma_{i,j}(\alpha)$ and $\sigma_{i,j}(\beta)$ were randomly selected from shareholder $s_{i,j}$. In this case $\sigma_{i,j}(\gamma)$ can be written as:

$$\sigma_{i,j}(\alpha\beta) = p^j(i) = [f_\alpha(i) \cdot f_\beta(i)]^j = f_\alpha^j(i) \cdot f_\beta^{j-1}(i) + f_\alpha^{j-1}(i) \cdot f_\beta^j(i).$$

The terms $f_\alpha^j(i)$ and $f_\beta^j(i)$ constitute the random values $\sigma_{i,j}(\alpha)$ and $\sigma_{i,j}(\beta)$ selected by shareholder $s_{i,j} \in S$. It is left to check that $\sum_{l=1}^{r} \delta_{l,(i,j)}$ and $\sum_{l=1}^{r} \varepsilon_{l,(i,j)}$ correspond to $f_\alpha^{j-1}(i)$ and $f_\beta^{j-1}(i)$, respectively. From the second step, we recall that $\delta_{l,(i,j)} = \sum_{m=1}^{r} \lambda_{l,m,(i,j)}$. Thus, it follows that:

$$\sum_{l=1}^{r} \delta_{l,(i,j)} = \sum_{l=1}^{r} \sum_{m=1}^{r} \lambda_{l,m,(i,j)} = \sum_{l=1}^{r} f_{\alpha,l}^{j-1}(i) = f_\alpha^{j-1}(i),$$

where polynomial $f_{\alpha,l}^{j-1}(x)$ is the $(j-1)$-th derivative of polynomial $f_{\alpha,l}(x) = \sum_{k=0}^{t-1} \alpha_{l,k} x^k$, where $\alpha_{l,k}$ is the reconstructing term of Birkhoff interpolation formula (see Section 3.2). Note that the last equality of the expression above holds because the coefficients of $f_\alpha(x)$ can be computed in distributed fashion, see Theorem 2 in [34]. The equality $\sum_{l=1}^{r} \varepsilon_{l,(i,j)} = f_\beta^{j-1}(i)$ can be shown analogously.

Moreover, since polynomial $p(x) = c_0 + c_1 x + \cdots + c_{2(t-1)} x^{2(t-1)}$ is the product of polynomials $f_\alpha(x)$ and $f_\beta(x)$, then $c_0 = a_0 b_0 = \alpha\beta (c_{2(t-1)} = a_{t-1} b_{t-1} = \alpha\beta)$. Thus, correctness holds. To prove perfect secrecy, we have to show that no information is leaked when share $\sigma_{i,j}(\alpha\beta)$ is generated for shareholder $s_{i,j} \in S$. Regarding the terms $\sum_{l=1}^{r} \delta_{l,(i,j)}$ and $\sum_{l=1}^{r} \varepsilon_{l,(i,j)}$, we have to show that they do not leak information about shares $\sigma_l(\alpha)$ and $\sigma_l(\beta)$ of shareholder $s_l \in R$, respectively. That is the case because shareholder $s_l \in R$ uses additive secret sharing [13] to split $\lambda_{l,(i,j)}$ and $\mu_{l,(i,j)}$ into r random values $\lambda_{m,l,(i,j)}$ and $\mu_{m,l,(i,j)}$, respectively. Furthermore, perfect secrecy holds also because index $i \in \mathcal{I}$ of each identity ID $(i,j) \in \mathcal{I} \times \mathcal{I}$ is used once, as required by Assumption (A3) of Section 4.1. Otherwise, points $f_\alpha^{j-1}(i)$ and $f_\beta^{j-1}(i)$ might correspond to already existing shares $\sigma_{i,j-1}(\alpha)$ and $\sigma_{i,j-1}(\beta)$ for α and β, respectively, already computed for shareholder $s_{i,j-1} \in S$. Moreover, because each share $\sigma_{i,j}(\gamma)$ is a point on polynomial $p(x)$ or on one of its derivatives, the underlying conjunctive (disjunctive) hierarchical secret sharing scheme ensures that unauthorized subsets gain no information about α, β, γ.

6 Auditing Procedure for Computations over Hierarchically Shared Messages

In this section, we provide measures allowing a third party to verify that a function on shares has been computed correctly, i.e that the message reconstructed from the computed shares is the correct outcome. We present auditing procedures suitable for the algorithms Linear, PreMult, and Multiply for conjunctive (disjunctive) hierarchical secret sharing schemes. This is achieved through commitment schemes and techniques applied in verifiable secret sharing [29].

6.1 Verifiable Secret Sharing and Commitments Schemes

Verifiable secret sharing was introduced in [9] to allow shareholders to check the consistency of shares received from the message dealer. More precisely, audit data are generated that allow the shareholders to check whether the shares of each authorized subset of shareholders lead to the same message during the reconstruction algorithm. To provide verifiable secret sharing usually *commitment schemes* are used, which come with two properties. First, bindingness ensures that it is not possible to change the message committed to. Second, hidingness ensures that no information about the message is leaked. Furthermore, there are several commitment schemes with homomorphic properties available, i.e. operations performed on the values committed to can be transferred to operations performed on the commitments. Verifiable secret sharing uses Feldman commitment [15], which is unconditionally binding and computationally hiding, or Pedersen commitment [28], which is computationally binding and unconditionally hiding. In the following, we use Feldmann commitment for the sake of simplicity, but our solutions work with both schemes. In the following, we recall the definition of Feldman commitment and Pedersen commitment (in brackets).

Definition 2 ([15],[28]). *Feldman (Pedersen) commitment scheme is a triple* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *of the following algorithms.*

Setup *It takes as input a security parameter λ and it outputs a prime q, a group $\mathbb{G}$ of order q, and a generator $g \in \mathbb{G}$ (distinct generators $g, h \in \mathbb{G}$).*

Commit *It takes as input a message $m \in \mathbb{F}_q$ (and randomness $r \in \mathbb{F}_q$) and it outputs commitment $c = g^m$ $(c = g^m h^r)$.*

Open *It takes as input a commitment $c \in \mathbb{G}$, a message $m \in \mathbb{F}_q$ (and randomness $r \in \mathbb{F}_q$) and it outputs '1' if $c = g^m$ (if $c = g^m h^r$) and '0' otherwise.*

6.2 Auditing Procedure for Conjunctive (Disjunctive) Hierarchical Secret Sharing Schemes

In this section, we present auditing procedures for computations on messages shared hierarchically by using Tassa's conjunctive (disjunctive) hierarchically secret sharing schemes, based on Birkhoff interpolation. More precisely, first, we present algorithms Audit.Setup and Audit.Share, which describes the steps to be performed during the setup phase and after algorithm Share, respectively. Then, we present algorithm Audit.Linear which is run after algorithm Linear to verify the correctness of linear operations. Finally, we present algorithms Audit.PreMult and Audit.Multiply, which allow auditing of multiplications.

Setup and Share. Algorithm Audit.Setup sets up the cryptographic primitives, i.e. commitment schemes and bilinear maps[2], needed for the auditing procedures. This can be run by any party. However, the parameters must be made publicly available for the dealer of the input messages and the auditor running the auditing procedures. Then, to allow operations to be audited, the dealer commits to messages shared by running Audit.Share.

Audit.Setup The algorithm takes as input a security parameter λ and it outputs two large primes p, q such that $q|(p-1)$. It also outputs a generator g of the q-th order subgroup $\mathbb{F}_q$ of $\mathbb{F}_p^*$.

Audit.Share The dealer of messages $m_1, m_2 \in \mathbb{F}_q$ calls algorithm Commit.Share during algorithm Share and computes commitment $c(m_1) := g^{m_1} \mod p$ to message m_1 and commitment $c(m_2) := g^{m_2} \mod p$ to message m_2. It publishes the commitments on the bulletin board.

Linear Operations. In the following, algorithm Audit.Linear run by the auditor to verify the result of linear operations over shared messages is presented. We assume that either the shareholders or the message dealer published the used scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$ on the bulletin board.

Audit.Linear The algorithm takes as input the commitments to the input values $c(m_1), c(m_2)$ and the scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$ from the bulletin board and the claimed result m. If $g^m = c(m_1)^{\lambda_1} \cdot c(m_2)^{\lambda_2}$ it returns '1' and '0' otherwise.

[2] For a formal definition of bilinear maps we refer to [6].

Multiplication. In the following, the auditing procedure for products over shared messages is presented. More precisely, first algorithm Audit.PreMult is introduced, which computes commitments to the multiplicative triples generated during algorithm PreMult of Section 5. Second, algorithm Audit.PreMult showing the auditing procedure for algorithm Multiply is presented.

Note that, algorithm PreMult is performed in distributed fashion by the shareholders of an authorized set $R \in \Gamma$. That is, each shareholder $s_{i,j} \in S$ receives input from each shareholder contained in R to compute share $\sigma_{i,j}(\alpha\beta)$. If one of the inputs is not valid, then shareholder $s_{i,j}$ cannot compute a valid share for $\alpha\beta$. This also affects the correctness of algorithm Multiply. In the following, it is explained what audit data have to be generated such that shareholder $s_{i,j}$ can detect inconsistent input sent by other malicious shareholders during algorithm PreMult of Section 5 performing the preprocessing phase.

Audit.PreMult The algorithm is run by the auditor to verify whether the shares $\sigma_{i,j}(\alpha\beta)$, output of algorithm PreMult, have been computed correctly. The algorithm takes as input from the bulletin board commitments $c_{k,\alpha}, c_{k,\beta}$, for $k = 0, \ldots, t-1$, to the coefficients of the polynomials $f_\alpha(x), f_\beta(x)$ sharing α and β, respectively. Appendix B.2 shows how commitments $c_{k,\alpha}$ and $c_{k,\beta}$ are computed. Then, each shareholder $s_{i,j} \in S$ has valid input $\delta_{l,i,j}$ and $\varepsilon_{l,i,j}$, for $l = 1, \ldots, r$, to compute share $\sigma_{i,j}(\alpha\beta)$ if and only if

$$g^{\sum_{l=1}^{r} \delta_{l,i,j}} \equiv \prod_{k=j-1}^{t-1} c_{k,\alpha}{}^{\frac{k!}{(k-j+1)!} i^{k-j+1}} = g^{f_\alpha^{(j-1)}(i)},$$

and if and only if

$$g^{\sum_{l=1}^{r} \varepsilon_{l,i,j}} \equiv \prod_{k=j-1}^{t-1} c_{k,\beta}{}^{\frac{k!}{(k-j+1)!} i^{k-j+1}} = g^{f_\beta^{(j-1)}(i)}.$$

If one of the both equalities is not satisfied, then it outputs '0' and aborts. Otherwise, each shareholder $s_{i,j} \in S$ holding shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma)$ computes commitments $c_{i,j}(\alpha) := g^{\sigma_{i,j}(\alpha)}, c_{i,j}(\beta) := g^{\sigma_{i,j}(\beta)}$, and $c_{i,j}(\gamma) := g^{\sigma_{i,j}(\gamma)}$ for $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$, and $\sigma_{i,j}(\gamma)$, respectively. It publishes $c_{i,j}(\alpha), c_{i,j}(\beta)$, and $c_{i,j}(\gamma)$ on the bulletin board and outputs '1'.

If algorithm Audit.PreMult has been run successfully, i.e. it outputs '1', then the shareholders can perform a multiplication on their shares and the auditor can call algorithm Audit.PreMult to verify the correctness of the result computed.

Audit.Multiply The algorithm takes as input the values δ, ε, the commitments to the shares of the multiplicative triple, i.e. $c_{i,j}(\alpha), c_{i,j}(\beta)$, and $c_{i,j}(\gamma)$, for $s_{i,j} \in S$, the commitments to the input values, i.e. $c(m_1), c(m_2)$, and the claimed result m. Then, it first audits that the equation $\alpha\beta = \gamma$ was fulfilled and then that m has been computed correctly performing the following steps.

First, the auditor computes the reconstruction vector $(w_1, \ldots, w_r)^3$ for shareholders $s_1, \ldots, s_r \in R$, with $R \in \mathit{\Gamma}$ authorized set, which computed the input for γ during PreMult. Then, it computes the following commitments:

$$c(\alpha) := \prod_{l=1}^{r} c_l(\alpha)^{w_l}; \quad c(\beta) := \prod_{l=1}^{r} c_l(\beta)^{w_l}; \quad c(\gamma) := \prod_{l=1}^{r} c_l(\gamma)^{w_l},$$

where $c_l(\alpha), c_l(\beta), c_l(\gamma)$, for $l = 1, \ldots, r$, are commitments $c_{i,j}(\alpha), c_{i,j}(\beta), c_{i,j}(\gamma)$, respectively, in lexicographic order. The multiplicative triple (α, β, γ) was correct if and only if $e(c(\alpha), c(\beta)) = e(c(\gamma), g)^4$. If the equation does not hold it outputs '0' and aborts the algorithm. Otherwise, the auditor takes from the bulletin board commitments $c(m_1), c(m_2)$ and the values δ, ε reconstructed during algorithm Multiply. If it holds that $c(\alpha)^{-1} \cdot c(m_1) = g^{\delta}$ and $c(\beta)^{-1} \cdot c(m_1) = g^{\varepsilon}$ and $g^m = c(\gamma) \cdot c(m_1)^{\varepsilon} \cdot c(m_2)^{\delta} \cdot g^{-\delta\varepsilon}$ it returns '1' and '0' otherwise.

7 Security and Efficiency

Security. We have proven that algorithm Linear of Section 4.2, algorithm Multiply of Section 4.3, and algorithm PreMult of Section 5 do not compromise the perfect secrecy and correctness of the underlying conjunctive (disjunctive) hierarchical secret sharing scheme. The adversary these algorithms can cope with is active, i.e. not only it knows data private to shareholders (like the passive adversary), but also it can make them deviate from the protocols. More precisely, assumptions (A1)-(A4) of Section 4.1 set requirements for, respectively, the access structure, the threshold, the identities of the shareholders, and the channels through which shareholders communicate. These assumptions together with verifiable secret sharing ensure that a honest majority of shareholders is able to correctly reconstruct the message, while maintaining the secrecy of their shares, even if all other shareholders are corrupted by the adversary and cheat. Assumption (5) prevents the adversary from tampering the bulletin board and, together with the auditing procedure, ensures correctness when operations on data are performed. As it is shown in [34], conjunctive (disjunctive) hierarchical secret sharing schemes support proactive secret sharing [21]. This means that, provided that the shares are refreshed periodically, our protocols can cope with a mobile adversary, which is only bounded in the amount of shareholders it can corrupt within a certain time interval, but not over time. Furthermore, we provide an auditing procedure in Section 6 allowing to detect misbehaviors. The protocols described use Feldman commitment, which ensures only computationally hidingness. However, the auditing procedure can be easily adapted to Pedersen commitment to achieve unconditionally hidingness, which preserves even perfect secrecy of the underlying conjunctive (disjunctive) hierarchical secret sharing scheme.

[3] For conjunctive (disjunctive) hierarchical secret sharing schemes the interpolation vector is composed of the entries $w_l := (-1)^{l-1}\frac{\det(A_{l-1,0}(E,X,\varphi))}{\det(A(E,X,\varphi))}$ $\big(w_l := (-1)^{l+t-2}\frac{\det(A_{l-1,t-1}(E,X,\varphi))}{\det(A(E,X,\varphi))}\big)$ according to the notation of Section 3.2.

[4] Here the definition of bilinear maps is used.

Efficiency. With respect to efficiency, the algorithms Share, Reconstruct, Linear, Multiply, and PreMult for conjunctive (disjunctive) hierarchical secret sharing perform equally well as Shamir's threshold secret sharing. Besides polynomials' evaluation, algorithm Share requires also to compute up to $t-1$ polynomials' derivatives. However, the additional multiplications due to derivation are balanced by the fewer multiplications needed when evaluating derivatives of polynomials. Algorithm Recostruct is the most expensive algorithm and requires in both Tassa's and Shamir's scheme to perform Gaussian elimination to find a solution to a system of t linear equations. Algorithm Linear and Multiply require that the shareholders perform steps very similar to the corresponding algorithms for Shamir's secret sharing (see for instance [3], [30]). Algorithm PreMult requires more work with respect to the preprocessing phase compared to Shamir's threshold secret sharing. In fact, algorithm PreMult is computed in distributed fashion because additional information is needed to compute the shares. Despite the fact that only additions and polynomials' evaluation are performed to compute such additional information, algorithm PreMult increases the communication cost and requires secure channels. For Shamir's threshold secret sharing scheme this additional information needs not to be computed and the communication complexity is, thus, lower. For the same reasons, the auditing procedure during the on-line phase of Tassa's schemes has computational complexity similar to the one for Shamir's scheme while the auditing procedure during the off-line phase is more expensive. In fact, to perform algorithms Audit.Linear and Audit.Multiply, the auditor takes steps very similar to the corresponding auditing procedure for Shamir's secret sharing schemes, because algorithms Linear and Multiply are defined similarly. Instead, algorithm Audit.PreMult requires the computation of commitments in a distributed fashion, which increases the communication and the computation cost. However, we recall that the preprocessing phase is off-line and can be performed in advance. Regarding the on-line phase, which is the time critical phase, the schemes of both Shamir and Tassa perform equally well.

8 Conclusion

In this work, we showed how to practically compute linear operations and multiplications over shared messages when Tassa's conjunctive and disjunctive hierarchical secret sharing schemes are used. Together with the property of modifying the access structure and changing the set of shareholders shown in [34], we proved that Birkhoff interpolation-based secret sharing schemes allow for the same functionalities as Shamir's secret sharing scheme, which is based on Lagrange interpolation. Furthermore, we showed how to perform the preprocessing phase enabling to reconstruct the product of two shared messages and provided auditing procedures to check that the operations were performed correctly. Moreover, the protocols we proposed do not lower the overall security of the underlying conjunctive and disjunctive hierarchical secret sharing scheme and do not increase in the on-line phase the computation overhead with respect to the same protocols for Shamir's secret sharing scheme. From a theoretical point of view, this result can be inferred from the approach presented in [10], which shows how secure multi-party computation can be built from linear secret sharing schemes. However, such approach is very general and does not show how this can be done for specific schemes, which is what we provide here for Birkhoff interpolation-based secret sharing schemes. From a practical point of view, this result is more interesting because it impacts the framework of cloud computing and distributed storage systems. More precisely, the possibility to perform operations over hierarchically shared messages sets Tassa's conjunctive and disjunctive hierarchical secret sharing schemes as promising candidates for distributed storage systems, where the storage servers are granted with different reconstruction capabilities depending on their performance [25], [35]. In fact, Tassa's conjunctive and disjunctive hierarchical secret sharing schemes together with the auditing procedure we presented would allow computations on documents outsourced to the cloud and stored in distributed fashion.

References

1. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. pp. 420–432 (1991), http://dx.doi.org/10.1007/3-540-46766-1_34
2. Beimel, A.: Secret-sharing schemes: A survey. In: Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings. pp. 11–46 (2011), http://dx.doi.org/10.1007/978-3-642-20901-7_2
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10 (1988), http://doi.acm.org/10.1145/62212.62213
4. Blakley, G.R., et al.: Safeguarding cryptographic keys. In: Proceedings of the national computer conference. vol. 48, pp. 313–317 (1979)
5. Blundo, C., Cresti, A., Santis, A.D., Vaccaro, U.: Fully dynamic secret sharing schemes. In: Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings. pp. 110–125 (1993), http://dx.doi.org/10.1007/3-540-48329-2_10
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2003), https://doi.org/10.1137/S0097539701398521
7. Brickell, E.F.: Some ideal secret sharing schemes. In: Advances in Cryptology EUROCRYPT'89. pp. 468–475. Springer (1990)
8. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 11–19 (1988), http://doi.acm.org/10.1145/62212.62214
9. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985. pp. 383–395 (1985), http://dx.doi.org/10.1109/SFCS.1985.64

10. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. pp. 316–334 (2000), https://doi.org/10.1007/3-540-45539-6_22
11. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. pp. 572–590 (2007), http://dx.doi.org/10.1007/978-3-540-74143-5_32
12. Desmedt, Y., Jajodia, S.: Redistributing secret shares to new access structures and its applications. Tech. rep., Technical Report ISSE TR-97-01, George Mason University (1997)
13. Doganay, M.C., Pedersen, T.B., Saygin, Y., Savas, E., Levi, A.: Distributed privacy preserving k-means clustering with additive secret sharing. In: Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society, PAIS 2008, Nantes, France, March 29, 2008. pp. 3–11 (2008), http://doi.acm.org/10.1145/1379287.1379291
14. Farràs, O., Padró, C.: Ideal hierarchical secret sharing schemes. In: Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings. pp. 219–236 (2010), http://dx.doi.org/10.1007/978-3-642-11799-2_14
15. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Foundations of Computer Science, 1987., 28th Annual Symposium on. pp. 427–438. IEEE (1987)
16. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In: Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998. pp. 101–111 (1998), http://doi.acm.org/10.1145/277697.277716
17. Ghodosi, H., Pieprzyk, J., Safavi-Naini, R.: Secret sharing in multilevel and compartmented groups. In: Information Security and Privacy. pp. 367–378. Springer (1998)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229 (1987), http://doi.acm.org/10.1145/28395.28420
19. Gupta, V., Gopinath, K.: $\mathrm{G}_{\mathrm{its}}{}^{2}$ VSR: : An information theoretical secure verifiable secret redistribution protocol for long-term archival storage. In: Security in Storage Workshop, 2007. SISW'07. Fourth International IEEE. pp. 22–33. IEEE (2007)
20. Heather, J., Lundin, D.: The append-only web bulletin board. In: Formal Aspects in Security and Trust, 5th International Workshop, FAST 2008, Malaga, Spain, October 9-10, 2008, Revised Selected Papers. pp. 242–256 (2008), https://doi.org/10.1007/978-3-642-01465-9_16
21. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Advances in Cryptology CRYPTO'95, pp. 339–352. Springer (1995)
22. Hogan, M., Liu, F., Sokol, A., Tong, J.: NIST Cloud Computing Standards Roadmap. NIST Special Publication 35 (2011)
23. Käsper, E., Nikov, V., Nikova, S.: Strongly multiplicative hierarchical threshold secret sharing. In: Information Theoretic Security - Second International Confer-

ence, ICITS 2007, Madrid, Spain, May 25-29, 2007, Revised Selected Papers. pp. 148–168 (2007), https://doi.org/10.1007/978-3-642-10230-1_13

24. Loruenser, T., Happe, A., Slamanig, D.: Archistar: towards secure and robust cloud based data sharing. In: Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on. pp. 371–378. IEEE (2015)
25. Nojoumian, M., Stinson, D.R.: Social secret sharing in cloud computing using a new trust function. In: Tenth Annual International Conference on Privacy, Security and Trust, PST 2012, Paris, France, July 16-18, 2012. pp. 161–167 (2012), http://dx.doi.org/10.1109/PST.2012.6297936
26. Nojoumian, M., Stinson, D.R., Grainger, M.: Unconditionally secure social secret sharing scheme. Information Security, IET 4(4), 202–211 (2010)
27. Pakniat, N., Eslami, Z., Nojoumian, M.: Ideal social secret sharing using Birkhoff interpolation method. IACR Cryptology ePrint Archive 2014, 515 (2014), http://eprint.iacr.org/2014/515
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology CRYPTO'91. pp. 129–140. Springer (1992)
29. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 73–85. ACM (1989)
30. Schabhüser, L., Demirel, D., Buchmann, J.A.: An unconditionally hiding auditing procedure for computations over distributed data. In: 2016 IEEE Conference on Communications and Network Security, CNS 2016, Philadelphia, PA, USA, October 17-19, 2016. pp. 552–560 (2016), http://dx.doi.org/10.1109/CNS.2016.7860547
31. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (Nov 1979), http://doi.acm.org/10.1145/359168.359176
32. Simmons, G.J.: How to (really) share a secret. In: Proceedings on Advances in cryptology. pp. 390–448. Springer-Verlag New York, Inc. (1990)
33. Tassa, T.: Hierarchical threshold secret sharing. Journal of Cryptology 20(2), 237–264 (2007)
34. Traverso, G., Demirel, D., Buchmann, J.A.: Dynamic and verifiable hierarchical secret sharing. In: Information Theoretic Security - 9th International Conference, ICITS 2016, Tacoma, WA, USA, August 9-12, 2016, Revised Selected Papers. pp. 24–43 (2016), http://dx.doi.org/10.1007/978-3-319-49175-2_2
35. Traverso, G., Demirel, D., Habib, S.M., Buchmann, J.A.: As^3: Adaptive social secret sharing for distributed storage systems. In: 14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12-14, 2016. pp. 528–535 (2016), https://doi.org/10.1109/PST.2016.7907011

Appendix

A Requirements for Birkhoff interpolation matrices Intepolation

In this section the necessary requirements and a sufficient condition for the interpolation matrix E are presented, such that the corresponding Birkhoff interpolation problem is well posed. For the corresponding proofs we refer to [33].

Lemma 1. *Let $A \subset S$ be an authorized subset of shareholders, i.e. $A \in \Gamma$, and E the corresponding interpolation matrix, where the entries $e_{i,j}$ of the matrix E satisfy the following condition:*

$$\sum_{j=0}^{k} \sum_{i=1}^{r} e_{i,j} \geq k+1, \quad 0 \leq k \leq d, \tag{1}$$

where d is the highest derivative order in the problem and $r := |A|$ is the number of interpolating points.

Before providing the sufficient condition (Theorem 4), the following definition is needed.

Definition 3 ([33]). *In the interpolation matrix E a* 1-sequence *is a maximal run of consecutive* 1*s in a row of the matrix E itself. Namely, it is a triplet of the form (i, j_0, j_1) where $1 \leq i \leq r$ and $0 \leq j_0 \leq j_1 \leq d$, such that $e_{i,j} = 1$ for all $j_0 \leq j \leq j_1$, while $e_{i,j_0-1} = e_{i,j_1+1} = 0$. A* 1*-sequence (i, j_0, j_1) is called* supported *if E has* 1*s both to the northwest and southwest of the leading entry in the sequence, i.e. there exist indexes nw and sw, where $i_{nw} < i < i_{sw}$ and $j_{nw}, j_{sw} < j_0$ such that $e_{i_{nw},j_{nw}} = e_{i_{sw},j_{sw}} = 1$.*

Theorem 4. *The interpolation Birkhoff problem for an authorized subset A and the corresponding interpolation matrix E has a unique solution, if the interpolation matrix E satisfies (1) and contains no supported* 1*-sequence of odd length.*

In case the Birkhoff interpolation problem is instantiated over a finite field $\mathbb{F}_q$ with $q > 0$ a prime number, then also the following condition has to hold.

Theorem 5. *The Birkhoff interpolation problem for an interpolation matrix E has a unique solution over the finite field $\mathbb{F}_q$, if Theorem 4 holds and in addition also the following inequality is satisfied:*

$$q > 2^{-d+2} \cdot (d-1)^{\frac{(d-1)}{2}} \cdot (d-1)! \cdot x_r^{\frac{(d-1)(d-2)}{2}}, \tag{2}$$

where d is the highest derivative order of the problem.

B Computation of shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ and commitments $c_{k,\alpha}, c_{k,\beta}$

In this section, we explain how the inputs to algorithm PreMult of Section 5 and algorithm Audit.PreMult of Section 6.2 are computed. We recall that Assumptions (A1), (A2), (A3), (A4), and (A5) of Section 4.1 hold and that information relating to disjunctive hierarchical secret sharing schemes is put in brackets.

B.1 Computation of shares $\boldsymbol{\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)}$

In this section, algorithm RandShares is presented, which computes random shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ reconstructing to messages α, β, respectively. Algorithm RandShares constitutes the first step of algorithm PreMult of Section 5. The challenge is that messages α, β are unknown and the shareholders have to find a way to coordinate their choices for shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ without leaking information. The strategy we adopt is to make use of the underlying conjunctive (disjunctive) hierarchical secret sharing scheme to generate the shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$. In the following, we present algorithm RandShares to compute shares $\sigma_{i,j}(\alpha)$ for shareholders $s_{i,j}$ reconstructing to message α. Algorithm RandShares can be run analogously to generate shares $\sigma_{i,j}(\beta)$ reconstructing to message β.

RandShares The algorithm takes as input values $\alpha_{i,j} \in \mathbb{F}_q$ chosen uniformly at random by shareholders $s_{i,j} \in S$. It outputs shares $\sigma_{i,j}(\alpha)$ of message $\alpha \in \mathbb{F}_q$ for shareholders $s_{i,j} \in S$. To do that, each shareholder $s_{i,j} \in S$ has to perform the following steps.

1) It chooses a secret message $\alpha_{i,j} \in \mathbb{F}_q$ uniformly at random.
2) It runs algorithm Share of Section 3.2 to generate a polynomial $f_{\alpha_{i,j}}(x)$ of degree $t-1$ defined as $f_{\alpha_{i,j}}(x) := a_{0,(i,j)} + a_{1,(i,j)}x + \cdots + a_{t-1,(i,j)}x^{t-1}$, where $a_{0,(i,j)} = \alpha_{i,j}$ $(a_{t-1,(i,j)} = \alpha_{i,j})$ and coefficients $a_{1,(i,j)}, \ldots, a_{t-1,(i,j)} \in \mathbb{F}_q$ $(a_{0,(i,j)}, \ldots, a_{t-2,(i,j)} \in \mathbb{F}_q)$ are chosen uniformly at random. Shares $\sigma_{i',j'}(\alpha_{i,j})$ for shareholders $s_{i',j'} \in S$ with ID $(i', j') \neq (i, j)$ are computed as $\sigma_{i',j'}(\alpha_{i,j}) := f^{j'}_{\alpha_{i,j}}(i')$. Share $\sigma_{i,j}(\alpha_{i,j})$ for shareholder $s_{i,j}$ itself is computed as $\sigma_{i,j}(\alpha_{i,j}) := f^{j}_{\alpha_{i,j}}(i)$.
3) It sends shares $\sigma_{i',j'}(\alpha_{i,j})$ to shareholders $s_{i',j'} \in S$ with ID $(i', j') \neq (i, j)$ using a private channel and keeps share $\sigma_{i,j}(\alpha_{i,j})$.
4) It runs algorithm Linear of Section 4.2 to compute share $\sigma_{i,j}(\alpha)$ using share $\sigma_{i,j}(\alpha_{i,j})$ and all the shares $\sigma_{i,j}(\alpha_{i',j'})$ received from shareholders $s_{i',j'}$ as $\sigma_{i,j}(\alpha) := \sum_{(i',j') \neq (i,j)} \sigma_{i,j}(\alpha_{i',j'}) + \sigma_{i,j}(\alpha_{i,j})$.

In the following, we prove correctness of algorithm RandShares and we show that perfect secrecy, according to Definition 1, is provided.

Theorem 6. *The algorithm* RandShares *for conjunctive (disjunctive) hierarchical secret sharing introduced above computes the shares $\sigma_{i,j}(\alpha)$ correctly. More precisely, on input random secret messages $\alpha_{i,j}$, the shares computed by algorithm* RandShares *reconstruct to a common value α. Furthermore, perfect secrecy, according to Definition 1, is maintained while performing* RandShares*.*

Proof. Let $\sigma_{i,j}(\alpha) \in \mathbb{F}_q$ be the shares computed using algorithm RandShares and held by shareholders $s_{i,j} \in R$, where $R \in \Gamma$ is an authorized set. To prove correctness, we have to show that algorithm Reconstruct outputs a message α when it takes as input shares $\sigma_{i,j}(\alpha)$ held by shareholders of an authorized set R. This means that correctness holds provided that algorithm Reconstruct can be successfully run by shareholders of any authorized set. This is implied by the correctness of algorithm Linear, presented in Section 4.2. In fact, each share $\sigma_{i,j}(\alpha)$ is computed as a sum of shares $\sigma_{i,j}(\alpha_{i',j'})$ and share $\sigma_{i,j}(\alpha_{i,j})$. Thus, for the homomorphic property of polynomials, shares $\sigma_{i,j}(\alpha)$ is either a point of polynomial $f_\alpha(x) := a_{0,\alpha} + a_{1,\alpha}x + \cdots + a_{t-1,\alpha}x^{t-1} = \sum_{(i,j)} f_{\alpha_{i,j}}(x)$ or a point on one of its derivatives, where $a_{0,\alpha} = \sum_{(i,j)} \alpha_{i,j}(a_{t-1,\alpha} = \sum_{(i,j)} \alpha_{i,j})$. Because of the underlying conjunctive (disjunctive) hierarchical secret sharing scheme, any authorized set R of shareholders can run algorithm Reconstruct over their shares and retrieve message $\alpha := \sum_{(i,j)} \alpha_{i,j}$. This proves correctness. With respect to perfect secrecy, the underlying conjunctive (disjunctive) hierarchical secret sharing scheme guarantees that shares $\sigma_{i,j}(\alpha)$ are computed without leaking information about the secret messages $\alpha_{i,j}$. Furthermore, this implies that unauthorized sets of shareholders not only cannot successfully run algorithm Reconstruct to retrieve α, but also no information about it is gained.

B.2 Computation of commitments $c_{k,\alpha}, c_{k,\beta}$

In this section, algorithm Audit.RandShares is presented, which computes commitments $c_{k,\alpha}, c_{k,\beta}$ to the coefficients of the polynomials sharing messages α, β, respectively. Algorithm Audit.RandShares constitutes the first step of algorithm Audit.PreMult of Section 6.2. More precisely, commitments $c_{k,\alpha}, c_{k,\beta}$, for $k = 0, \ldots, t-1$, are used to check the validity of terms $\delta_{l,i,j}$ and $\varepsilon_{l,i,j}$ for the computation of shares $\sigma_{i,j}(\alpha\beta)$. Note that commitments $c_{k,\alpha}, c_{k,\beta}$ can be correctly computed provided that an auditing procedure verifying the validity of shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ for shareholders $s_{i,j}$ is performed, where shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ are the output of algorithm RandShares of Appendix B.1. For consistency with algorithm Audit.PreMult, Feldman commitment is used. However, the algorithm can be easily adapted to Pedersen commitment. In the following, we present algorithm Audit.RandShares to compute commitment $c_{k,\alpha}$, for $k = 0, \ldots, t-1$. Algorithm Audit.RandShares can be run analogously to generate commitment $c_{k,\beta}$, for $k = 0, \ldots, t-1$.

Audit.RandShares The algorithm is run by an auditor to verify that shares $\sigma_{i,j}(\alpha)$ was computed correctly. This is performed in the following steps.

1) Each shareholder $s_{i,j} \in S$ running algorithm Share to share the secret message $\alpha_{i,j} \in \mathbb{F}_q$ among all other shareholders $s_{i',j'} \in S$ for $(i',j') \neq (i,j)$ calls algorithm Commit.Share and computes commitments $c_{k,\alpha_{i,j}} := g^{a_{k,(i,j)}} \bmod p$, to coefficient $a_{k,(i,j)}$ of polynomial $f_{\alpha_{i,j}}(x)$, for $k = 0, \ldots, t-1$. It publishes the commitments on the bulletin board.
2) Each shareholder $s_{i,j} \in S$ has valid input $\sigma_{i,j}(\alpha_{i',j'})$, for $(i',j') \neq (i,j)$, to compute share $\sigma_{i,j}(\alpha)$ if and only if

$$g^{\sigma_{i,j}(\alpha_{i',j'})} \equiv \prod_{k=j}^{t-1} c_{k,\alpha_{i',j'}}{}^{\frac{k!}{(k-j)!} i^{k-j}} = g^{f^j_{\alpha_{i',j'}}(i)}.$$

If the above equality is not satisfied, then it outputs ‘0’ and aborts. Otherwise, it publishes ‘1’ on the bulletin board and Step 3) can be performed.

3) The auditor uses commitments $c_{k,\alpha_{i,j}}$ published by shareholders $s_{i,j} \in S$ on the bulletin board to compute commitments

$$c_{k,\alpha} := \prod_{(i,j)} c_{k,\alpha_{i,j}},$$

for $k = 0, \ldots, t-1$. It publishes the commitments on the bulletin board.