TRANSACTIONS ON DATA PRIVACY 4 (2011) 147–165

Secure Distributed Subgroup Discovery in Horizontally Partitioned Data

Henrik Grosskreutz, Benedikt Lemmen, Stefan Rüping

Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany

E-mail: {firstname.lastname}@iais.fraunhofer.de

Abstract. Supervised descriptive rule discovery techniques like subgroup discovery are quite popular in applications like fraud detection or clinical studies. Compared with other descriptive techniques, like classical support/confidence association rules, subgroup discovery has the advantage that it comes up with only the top-k patterns, and that it makes use of a quality function that avoids patterns uncorrelated with the target. If these techniques are to be applied in privacy-sensitive scenarios involving distributed data, precise guarantees are needed regarding the amount of information leaked during the execution of the data mining. Unfortunately, the adaptation of secure multi-party protocols for classical support/confidence association rule mining to the task of subgroup discovery is impossible for fundamental reasons. The source is the different quality function and the restriction to a fixed number of patterns – i.e. exactly the desired features of subgroup discovery. In this paper, we present new protocols which allow distributed subgroup discovery while avoiding the disclosure of the individual databases. We analyze the properties of the protocols, describe a prototypical implementation and present experiments that demonstrate the feasibility of the approach.

1 Introduction

The question of the privacy of data can be an important aspect in the real-world application of data mining. In privacy-sensitive scenarios, in particular those with distributed data, a failure to guarantee certain privacy-preserving constraints means that data mining can not be applied at all. As an example, consider the case of competing mail order companies. To a large part, these companies make money by knowing their customers better than their competitors do. On the other hand, they lose money due to fraud. Typically, the risk of disclosing sensitive customer information by far outweighs the chances of reducing expenses by a joint fraud detection effort. Only privacy-preserving data mining techniques will allow an analysis of fraud patterns over all companies.

In applications like the above, descriptive techniques like rule mining are very popular, as they have the potential to provide more insight than numerical methods like SVMs or neural networks. Actually, protocols have been proposed that allow secure association rule mining over distributed databases [11, 26]. These, however, rely on the classical support/confidence framework, which has been observed to have effects undesired in some settings [27, 4, 10]: In particular, there is a danger to come up with huge amounts of rules which are not significant or do not express a correlation.

For this reason, several alternative (non-secure) rule mining approaches have been proposed, that deviate from the classical support/confidence framework. These include subgroup discovery [12] and correlated itemset mining [18], which are sometimes subsumed under the name *supervised descriptive rule discovery* [19]. The key differences compared

to classical support/confidence association rule mining are (i) the different quality function used to assess the patterns, and (ii) the intention to collect only a small set of k patterns instead of collecting all patterns satisfying some minimal threshold constraint. Interestingly, some recent association rule mining algorithms deviate from the classical support/confidence approach and also search for the top-k rules, applying quality functions similar to those used in subgroup discovery [17, 27, 10].

Unfortunately, it is *impossible* to adapt existing secure protocols for classical support/confidence association rule mining, like [11], to the task of subgroup discovery. The reason lies in the different quality function. The existing protocols rely on the property that in the support/confidence framework *every globally large itemset must be locally large at least at one of the sites*. However, an analogous property does not hold for the quality functions used in subgroup discovery: the globally best rules are not guaranteed to also be among the locally best rules of any site, neither exactly nor approximately [23, 29]. A second source of problems is the restriction to the top-k subgroups, which has the effect that it only becomes clear whether a subgroup is part of the result once *all* subgroups have been considered.

In this paper, we present secure protocols for the task of top-k subgroup discovery on horizontally partitioned data. In this setting, all sites use the same set of attributes and the quality of every subgroup depends on all databases. The approach finds patterns in the union of the databases, without disclosing the local databases. To the best of our knowledge, this is the first secure approach that tackles any of the above-mentioned supervised descriptive rule discovery tasks.

The remainder of this paper is structured as follows: After a discussion of related work in Section 2, we review of some basic definitions in Section 3. Thereafter, we present a top-1 subgroup discovery protocol in Section 4. We prove that it is secure in Section 5. Subsequently, we present protocols for collecting sets of subgroups in Section 6, where we also present a greedy approach to speedup the computation. Finally, we describe a prototypical implementation in Section 7, before we conclude in Section 8.

2 Related Work

Privacy-preserving data mining has emerged to address the situation when the use of data mining techniques is desired, but the data to be mined may not be disclosed or brought together for privacy considerations. The work presented in this paper falls into the category of approaches based on *secure multi-party computation* [21, 8]. Here, the premise is that the data is distributed over different parties, and the goal is to obtain data mining results without revealing the (private) data of one party to another. Data publishing issues, which are concerned with the amount of private information which can be deduced from a published set of patterns or data records (e.g. [22, 2]), are not considered here.

Secure multi-party computation originates in Yao' famous millionaires problem [30]: two millionaires want to find out who is richer without revealing the precise amount of their wealth. Yao has presented a generic solution that allows the secure evaluation of *any* two-party functionality [31, 15]. The solution is based on the encryption and secure evaluation of circuits. This technique has shown to be very useful to securely compute some subfunctionality, and has been applied for that purpose in several settings. While in principle, this generic solution could be used to securely compute any kind of (two-party) data mining task simply by using an appropriate circuit, this naive approach would result in huge circuits whose inputs would be entire databases – an approach which is not feasible in practice [21]. Another limitation of the approach is that it only considers two-party problems.

While there also exist generic constructions for the multi-party case, these have additional drawbacks which prevent their use in most applications [21].

For these reasons, custom secure multi-party solutions have been developed for many different data-mining problems. These include privacy-preserving protocols for decision tree induction [14], for nearest neighbor search [24] or for support vector machine classification [32]. The work most similar to ours is clearly the privacy-preserving association rule mining algorithm presented in [11].

Obviously, a nearby question is whether this approach can be extended to subgroup discovery. This, however, is not the case. The reason lies in the different quality function. The protocol proposed in [11] relies on the support/confidence framework, where every globally large itemset must be locally large at least at one of the sites. This allows the use of a distributed Apriori-style approach, similar to [5]. This approach is not possible for subgroup discovery due to its different quality function: In subgroup discovery, the global quality of *every* subgroup depends on *all* private databases, and, as shown by Scholz [23], the globally best subgroups are not guaranteed to also be among the locally best rules of any site, neither exactly nor approximately. Due to this fundamental difference, distributed Apriori-style approaches like [5, 11] are not possible for subgroup discovery.

This brings us to the related work in the area of subgroup discovery. Although many different subgroup discovery algorithms have been proposed, to the best of our knowledge the only approach dealing with distributed subgroup discovery is the *distributed global subgroup mining* protocol by Wurst and Scholz [29]. In this approach, every (non-prunable) candidate subgroup is allocated to one site which is responsible for its processing. The processing of the subgroups (may) include polling the local support counts from all participating sites. To speedup the computation, the authors propose *pruning based on partially available counts*: a special kind of optimistic estimate pruning [28] based on the supports polled from a subset of all sites.

The distributed global subgroup mining protocol of Wurst and Scholz, however, does not consider issues like privacy or secure multi-party computation and hence there are no obvious guarantees concerning the amount of information disclosed during the computation. In fact, sophisticated pruning strategies like *pruning based on partially available counts* rely on the disclosure of private information (i.e. of local supports) and thus violate the concept of secure computation. But even if such pruning strategies are disabled, the standard (non-secure) top-k subgroup discovery approaches cannot be applied in the context of secure multi-party computation. The problem is that in order to collect only the top-k subgroups, they make use of an increasing quality threshold which changes in view of the best subgroups collected so far. This approach, however, reveals information about the quality of the subgroups visited (resp. about their quality ordering), which again violates the concept of secure computation.

3 Preliminaries

In this section, we will briefly go over the most important notions from secure multi-party computation and subgroup discovery.

3.1 Privacy-Preserving Data-Mining and Secure Multi-Party Computation

The following section reviews the most important definitions of secure multi-party computation. The presentation is based on Goldreich [8], with some simplifications as we consider a less general scenario.

3.1.1 Multi-Party Computation

A multi-party problem, or *task*, is casted by specifying a mapping from sequences of inputs (one input per party) to sequences of outputs (one output per party). We refer to this mapping as the desired *functionality*, denoted $f : (\{0,1\}^*)^S \to (\{0,1\}^*)^S$, where S denotes the number of parties. That is, f is a mapping of the combined input $\bar{x} = (x_1, \ldots, x_S)$ to the combined output $(f_1(\bar{x}), \ldots, f_S(\bar{x}))$: the first party with input x_1 wishes to obtain $f_1(\bar{x})$, the second party with input x_2 wishes to obtain $f_2(\bar{x})$, etc. Clearly, the output of every party can depend on all inputs, $x_1, \ldots, x_S$.

3.1.2 A Definition of Secure Computation

In this paper, we consider multi-party computation in the *semi-honest model*, arguably the most commonly used model in privacy-preserving data-mining (e.g. [14, 24, 11]). The semi-honest model assumes that every party follows the protocol properly with the exception that it keeps a record of all intermediate computations and messages. One motivation for the semi-honest model is that parties who want to mine data for their mutual benefit will follow the protocol to get correct results. A second argument motivating the use of the semi-honest model is that records of the messages exchanged during the execution of a protocol are likely to be available anyway through some standard logging of the operating system, which makes "totally honest" behavior hard to enforce while semi-honest behavior may be assumed in many setting.

Intuitively, a protocol π securely computes a functionality f if whatever a semi-honest party can obtain after participating in the protocol could be obtained from the input and output available to that party. This is formalized according to the simulation paradigm, which requires that every party's view in a protocol execution can be simulated given only its input and output. As the actual execution may involve messages based on random numbers, the simulator does not need to generate *exactly* what is seen during the actual execution. The following definition makes this precise:

Definition 1. Let $f = (f_1, \ldots, f_S)$ be a deterministic S-ary functionality and π be a multi-party protocol for computing f. For a party i, let $\mathsf{view}_i^\pi(\bar{x})$ denote its view, i.e. its input x_i and the sequence of messages it has received. We say that π *securely computes* f in the presence of semi-honest adversaries without collusion if there exist probabilistic polynomial-time algorithms $S_i, 1 \leq i \leq S$, also called *simulators*, such that for every party the view is computationally indistinguishable from the simulation:

$$\{(S_i(x_i, f_i(\bar{x}))\}_{\bar{x}\in(\{0,1\}^*)^S} \equiv^C \{\mathsf{view}_i^\pi(\bar{x})\}_{\bar{x}\in(\{0,1\}^*)^S} \quad (1)$$

Here, $\equiv^C$ denotes computational indistinguishability. Loosely speaking, the simulations are computationally indistinguishable from the views if two cannot be reliably distinguished, i.e. if for every polynomial-time distinguisher D the probability that D distinguishes a view and the simulation decreases super-polynomially in the length of the input. For

our purpose, the most important fact is that any random number in a view is computationally indistinguishable from another random number drawn from the same probability distribution. More details can be found in [8].

3.1.3 Yao's Generic Circuit Solution

Yao has presented a generic solution that allows the secure evaluation of *any* two-party functionality [31, 15]. The solution is based on the encryption and secure evaluation of circuits. A detailed description of the circuit encryption scheme can be found in the excellent presentation of Lindell and Pinkas [15]. While encrypted circuits typically cannot be used directly to solve a data mining task (among others, because there are more than two participating sites), they can be very useful to securely compute some sub-functionality.

3.2 Subgroup Discovery

Subgroup discovery is a supervised descriptive rule learning task. In this paper, we only consider binary labeled data, thus a *subgroup description* can be seen as the antecedent of a rule whose consequent is the positive class.

Formally, let $\mathcal{A} = A_1, \ldots, A_m$ be a sequence of m sets we refer to as *attributes*. Beside these attributes, there is a special binary set $\{+, -\}$ called the *label*. A *data record* over $\mathcal{A}$ is an $m+1$-tuple $D = (a_1, \ldots, a_m, l) \in A_1 \times \cdots \times A_m \times \{+, -\}$. A *database* $\mathcal{D}$ over $\mathcal{A}$ is a multiset of data records over $\mathcal{A}$. We use the expression $\mathcal{D}^+$ to denote the sub-multiset of all $+$-labeled data records in $\mathcal{D}$. Formally, thus $\mathcal{D}^+ = \{(a_1, \ldots, a_m, l) \in \mathcal{D} \mid l = +\}$. Similarly, $\mathcal{D}^-$ denotes the sub-multiset of all $-$-labeled data records in $\mathcal{D}$.

The subgroup description language considered here is the language of conjunctions of attribute/value equality constraints. We formalize this as follows: a *constraint* over $\mathcal{A}$ is an expression $(A_i = v)$ with $i \in \{1, \ldots, m\}$ and $v \in A_i$. The language of *subgroup descriptions* over $\mathcal{A}$, denoted by $\mathcal{L}_\mathcal{A}$, is then the power set of constraints over $\mathcal{A}$. In the following, we drop the index $\mathcal{A}$ because it is always clear from the context. Given a subgroup description $s \in \mathcal{L}$, we call the number of constraints it is built of its *length*, denoted by *length*(s).

We will now turn to the semantics of subgroup descriptions: a data-record $(a_1, \ldots, a_m, l) \in \mathcal{D}$ satisfies a subgroup description $s \in \mathcal{L}$, if for all $(A_i = v) \in s$ it holds that $a_i = v$. The support set of s in $\mathcal{D}$ (or simply the "subgroup"), denoted by $\mathcal{D}[s]$, is then the sub-multiset of $\mathcal{D}$ containing the data records that satisfy s.

The "interestingness" of a subgroup description is measured by a *quality function*, which is a mapping from a database and a subgroup description to the reals. The idea is that high function values indicate interesting patterns. In this paper, we consider the *Piatetsky-Shapiro* quality function, which is (factor) equivalent to the *weighted relative accuracy* [13]. This function, which is arguably one of the most common subgroup quality functions, is defined as follows:

$$q(\mathcal{D}, s) = n(\mathcal{D}, s)\,(p(\mathcal{D}, s) - p_0(\mathcal{D}))\,. \tag{2}$$

Here, $n(\mathcal{D}, s) = |\mathcal{D}[s]|$ denotes the size of the subgroup, $p(\mathcal{D}, s) = |\mathcal{D}^+[s]| \,/\, |\mathcal{D}[s]|$ the fraction of records with positive label in the subgroup and $p_0(\mathcal{D}) = |\mathcal{D}^+| \,/\, |\mathcal{D}|$ the fraction of positive records in the overall population. Subgroup discovery is concerned with finding high-quality subgroups, as precised in the next section.

4 A Protocol for Secure Top-1 Subgroup Discovery

We assume that there are $S \geq 2$ sites (or parties) participating in the computation, each holding a private database $\mathcal{D}_i$ ($1 \leq i \leq S$) built over the same set of attributes. Instead of directly considering the task of top-k subgroup discovery, we first consider a specialization of this problem, namely finding a *single* subgroup of maximum quality. We will describe later (in Section 6), how given a solution to this task, we can iteratively collect a set of k subgroups.

4.1 The Task of Top-1 Subgroup Discovery

The task of finding a *single* subgroup of maximum quality can be seen as the special case of the top-k task where $k = 1$. Beside the subgroup description, we want to learn its quality, because this tells us whether the subgroup describes a significant phenomenon or merely a manifestation of noise in the data. The task we consider is thus the following:

Task 1. Top-1 Subgroup Discovery Given private databases $\mathcal{D}_1, \ldots, \mathcal{D}_S$ at Sites 1 to S (each built over the same set of attributes) together with a length limit L, calculate and distribute a maximum quality subgroup description s_{max} of length $\leq L$ together with its quality. That is, compute a pair $\langle s_{max}, q_{max} \rangle \in \mathcal{L} \times \mathcal{R}$ such that

$$q_{max} = \max_{\{s \in \mathcal{L} | \mathit{length}(s) \leq L\}} q(\mathcal{D}, s),$$

$$\mathit{length}(s_{max}) \leq L \text{ and } q(\mathcal{D}, s_{max}) = q_{max}.$$

Here, the quality function q (defined in Equation 2) is evaluated wrt. the disjoint union of the local databases, $\mathcal{D} = \bigoplus_{i=1}^{S} \mathcal{D}_i$.

It turns out that Task 1 is unexpectedly hard: First, as shown by Scholz [23], the globally best rules may perform poor at all local sites, and moreover the local quality of a subgroup can arbitrarily deviate from its global quality (no matter whether a relative or an absolute definition of support is applied). The consequence is that separately analyzing one (or all) local databases is of no help in finding the best global subgroup. This has the effect that existing distributed association rule mining protocols, which rely on the fact that every globally large itemset must be locally large at least at one of the sites [5, 11], cannot be adapted to the task of subgroup discovery. As a result, Wurst and Scholz have proposed a distributed global subgroup mining protocol [29] which computes the global quality of every subgroup essentially by polling the local support counts from all participating sites.

In the context of secure computation, we face an additional difficulty: the standard approach of non-secure top-k subgroup discovery algorithms – keeping track of the best subgroups observed so far together with their quality during the traversal of the search space [28, 9, 29] – results in a security leak. The reason is, loosely speaking, that the sequence of increments of the best quality cannot be simulated from the outcome. This sequence, however, reveals a lot of information about the data, as it induces a partial order over the quality of *all* subgroups visited during the exploration of the search space. For this reason, we compute the maximum-quality subgroup in two steps, by subsequently solving the following two sub-tasks:

- first, we compute the maximum of all subgroup qualities, that is, the value q_{max}. This is done in a way that only the maximum becomes known, but no ordering between subgroup qualities;

- second, we use this quality to securely find a maximum-quality subgroup.

We will discuss these two steps in Sections 4.2 and 4.3, respectively, before we present the overall protocol in Section 4.4. The relation of the task of *top-k* subgroup discovery will be considered later, in Section 6.

4.2 Computing the Maximum Quality

Our solution to the first sub-task works as follows: In a first phase, the sites collectively traverse the space of subgroup descriptions. For every subgroup description, the protocol ensures that Site 1 obtains a random value r_i, and Site S a second value $\tilde{q}_i + r_i$, where $\tilde{q}_i$ represents the quality of the subgroup (actually, $\tilde{q}_i$ is an integer-valued multiple of the subgroup quality, as explained below). The motivation for the distributed storage of r_i and $\tilde{q}_i + r_i$ is that none of the parties must learn the value $\tilde{q}_i$. In a second phase, Site 1 and Site S use the garbled qualities $\tilde{q}_i + r_i$ and the offsets r_i to securely calculate the maximum quality. We remark that the first phase is inspired by the garbled quality calculation in [6], while the second phase shares some similarity with the maximum computation in [14].

4.2.1 Computing the Garbled Qualities $\tilde{q}_i + r_i$.

We will now describe these two phases in more detail. We first observe that the Piatetsky-Shapiro quality (Equation 2) can be rewritten as follows [7]:

$$q(\mathcal{D}, s) = \left|\mathcal{D}^+[s]\right| (1 - p_0(\mathcal{D})) - \left|\mathcal{D}^-[s]\right| p_0(\mathcal{D}).$$

Given that $\mathcal{D} = \bigoplus_{i=1}^{S} \mathcal{D}_i$, this means that the quality can be expressed as a sum of local values:

$$q(\mathcal{D}, s) = \sum_{i=1}^{S} \left(\left|\mathcal{D}_i^+[s]\right| (1 - p_0(\mathcal{D})) - \left|\mathcal{D}_i^-[s]\right| p_0(\mathcal{D}) \right). \tag{3}$$

All that is required to compute the local summands is the value $p_0(\mathcal{D})$. Moreover, all summands are multiples of $1/|\mathcal{D}|$, because p_0 is a multiple of $1/|\mathcal{D}|$. Thus, assuming that every site has knowledge of $|\mathcal{D}|$, we can reduce the computation to arithmetics on integers. In fact, the $\tilde{q}_i$ mentioned earlier are simply the integers obtained by multiplying the quality of the i-th subgroup and $|\mathcal{D}|$. As final observation, we note that the values of the integers $\tilde{q}_i$ do not exceed $|\mathcal{D}|^2$.

Based on these observations, we realize the computation of the values r_i and $\tilde{q}_i + r_i$ at Sites 1 resp. S as follows: Site 1 generates a random number r_i uniformly distributed in $[0, \ldots, M]$, where M is a some power of 2 constant such that $M > |\mathcal{D}|^2$. Site 1 adds it local support, $(\left|\mathcal{D}_1^+[s_i]\right| (1 - p_0) - \left|\mathcal{D}_1^-[s_i]\right| p_0) \cdot |\mathcal{D}|$ to r_i, and sends the sum (modulo M) to Site 2. Sites 2 to (S-1) add their local support and send the result to the next site. Finally, Site S obtains the result, which according to Equation 3 corresponds to $\tilde{q}_i + r_i$.

4.2.2 Obtaining the Maximum

Once the first phase of the computation is completed, Site 1 has a list of values $r_1, ..., r_R$, and Site S another list $\tilde{q}_1 + r_1, ..., \tilde{q}_R + r_R$, where R denotes the number of subgroup descriptions visited during the traversal. In the second phase, only *two* sites are involved. This is an

important difference to the first phase, as it means that now we could apply a generic two-party solution like Yao's encrypted circuit scheme [31].

However, due to the potentially very large number of inputs this may result in an infeasibly large circuit, so we reduce the problem to smaller subproblems: We successively shorten the two lists by iteratively replacing *two* values by *one* at *both* Sites 1 and S. Thereby, we take care that while the length of the two lists decrease, together they still allow the reconstruction of q_{max}. This is done by replacing two values r_α, r_β at Site 1 by a *new* random value r', and the two corresponding values $r_\alpha + \tilde{q}_\alpha, r_\beta + \tilde{q}_\beta$ at Site S by $r' + \max(\tilde{q}_\alpha, \tilde{q}_\beta)$. The use of a new random number r' is important, as it will allow us to prove that the parties learn nothing new during the replacements[1]. The replacements take place until only one pair of values remains, whose difference reveals the quality q_{max}. Figure 1 illustrates the overall idea.

To put the above into action, all we need is a secure solution for the following functionality: Provided Site 1's input (r_α, r_β, r') and Site S's input $(r_\alpha + \tilde{q}_\alpha, r_\beta + \tilde{q}_\beta)$, calculate the combined output $(\perp, (\max(\tilde{q}_\alpha, \tilde{q}_\beta) + r') \mod M)$, i.e. Site 1 learns nothing and Site S learns the garbled maximum. Here, all inputs and outputs are integers in $[0, ..., M]$.

We realize this functionality using an encrypted circuit [31]. A detailed description of the circuit encryption scheme is beyond the scope of this paper (see the excellent presentation in [15] for details). The bottom line is that given some boolean circuit, this scheme allows to generate an encrypted representation of that circuit, together with an encoding table for the boolean input gates. Given this representation, plus cryptographic keys representing a particular set of boolean inputs, it is possible to calculate the (plain) value of the boolean outputs - but no additional information beyond that.

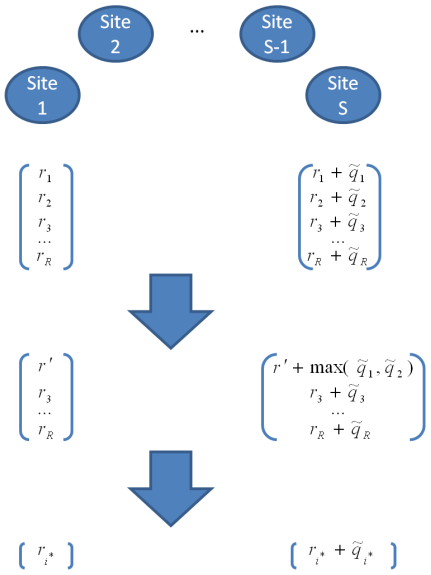

Figure 1: Maximum Quality Calculation

Using this encryption scheme, Site 1 proceeds as follows: First, it generates a boolean circuit that computes the above functionality. The circuit has input wires for the boolean representation of r_α, r_β, $\tilde{q}_\alpha + r_\alpha$, $\tilde{q}_\beta + r_\beta$ and r', and the output wires represent $(\max(\tilde{q}_\alpha, \tilde{q}_\beta) + r') \mod M$. The circuit calculates the output using some inverters and ripple-carry adders, a well-known type of digital circuit. Site 1 generates the encrypted representation of the circuit and sends the circuit (without the encoding table) to Site S, together with the keys representing its own input bits. Now all that Site S needs to calculate the output are the keys representing its own input. For this purpose, we make use of an additional ("third") party, T: Site 1 sends the encoding table for the inputs of Site S to Site T. Site S sends its (plain) input bits to Site T and ob-

[1]Simply dropping one of the pairs would not be secure, as it would reveal which subgroup has the higher quality. For similar reasons, we cannot store an encrypted representation of the subgroup description together with its garbled quality.

tains the corresponding cryptographic keys. Given this information, Site S can evaluate the encrypted circuit and thus obtain $(r' + \max(\tilde{q}_\alpha, \tilde{q}_\beta)) \mod M$. None of the parties learn anything else than the result, as we will prove in Section 5. We remark that if $S > 2$, the role of Site T can be implemented by one of the sites $2, ..., (S-1)$, e.g. Site 2, after some minor preprocessing.[2]

4.3 Computing a Maximum Quality Subgroup

Once the maximum quality q_{max} is computed, it is straightforward to solve Task 1 and compute a maximum quality subgroup. Basically, all that has to be done is to check, for every subgroup s_i, whether $q_i \geq q_{max}$, which is equivalent to $\tilde{q}_i + r_i \geq r_i + q_{max} \cdot |\mathcal{D}|$. The first subgroup satisfying this inequality is returned as outcome. The values $\tilde{q}_i + r_i$ are known at Site S after execution of Protocol 1, and moreover the values r_i, q_{max} and $|\mathcal{D}|$ are known at Site 1, so all that is needed is to securely compute the greater-or-equal test. This test, however, is equivalent to Yao' famous millionaires problem [30], and many solutions exist for this task (e.g. [25]).

4.4 The Protocol

We now have all ingredients for a secure solution for Task 1. Protocol 1 assumes the length limit L as input, together with local database $\mathcal{D}_i$, $i, 1 \leq i \leq S$.

First, the sites securely calculate $|\mathcal{D}|$ and $|\mathcal{D}^+|$. As $|\mathcal{D}|$ is the sum of the local $|\mathcal{D}_i|$, this reduces to the secure calculation of a sum of local values – a standard task in multi-party computation, for which efficient protocols exist (see e.g. [6]). Same for $|\mathcal{D}^+|$. The values $|\mathcal{D}|$ and $|\mathcal{D}^+|$ are distributed to all sites, which will enable them to calculate p_0 and hence the local values in Equation 3. Next, Site 1 and Site S initialize a local queue, $\mathcal{Q}_1$ resp. $\mathcal{Q}_S$. These will be used to store the values $r_1, ..., r_R$ (at Site 1) resp. $\tilde{q}_1 + r_1, ..., \tilde{q}_1 + r_R$ (at Site S).

Thereafter, the protocol iterates over all subgroup descriptions (Line 4.4 to 4.4). This is orchestrated by Site 1, which makes use of an iterator *iter* to generate all subgroup descriptions satisfying the length limit. The iterator traverses the space of subgroup descriptions in a canonically depth-first, left-to-right order, according to the lexicographic order of the constraints. It supports the following functionality resp. methods: (i) check if there is at least one more subgroup description left to be considered ("`hasNext`"), and (ii) retrieve the next subgroup description ("`next`") .

Using this iterator, Site 1 generates the next subgroup description s_i and informs all sites that s_i is the next subgroup description to be considered. Next, the parties collectively calculate r_i and $\tilde{q}_i + r_i$. As discussed earlier, the latter value is computed by iteratively adding the local support and sending the intermediate result to the next site. The values r_i resp. $r_i + \tilde{q}_i$ are separately stored in the queues $\mathcal{Q}_1$ and $\mathcal{Q}_S$ at Sites 1 and S, respectively.

When the second loop starts at Line 4.4, all candidate subgroups have been considered. The protocol will now determine a pair of values r_{i^*}, $(r_{i^*} + \tilde{q}_{i^*})$ such that $\tilde{q}_{i^*}$ is maximal. This is done by iteratively replacing (a pair of) *pairs* at Sites 1 and S simultaneously by (a pair of) *single values*. To this end, in every iteration of Line 4.4 a new encrypted circuit is generated at Site 1, which is evaluated afterwards by Site S, resorting to a third party T (which can optionally be implemented by Site 2 as discussed earlier). The loop ends when only one pair of values remains, which allows the calculation of the quality q_{max}.

[2] All that is required is that before the circuits are generated, Site 1 and Site S replace every r_i by $r_i + r'_i$, where r'_i is a newly generated random value. This prevents the third party from drawing conclusions by relating the observed inputs from Site S with the intermediate sums observed earlier.

Protocol 1 Top-1 Subgroup Discovery

INPUT: length limit L and local databases $\mathcal{D}_1, \ldots, \mathcal{D}_S$

1: Site 1 initiates the secure calculation of $|\mathcal{D}|$ and $|\mathcal{D}^+|$ and broadcasts the result
2: Site 1 creates a local iterator *iter* and a queue $\mathcal{Q}_1$, Site S creates a local queue $\mathcal{Q}_S$

3: **while** `hasNext`(*iter*) **do**
4: Site 1 calculates and broadcasts s_i = `next`(*iter*)
5: Site 1 generates a random number r_i uniformly in $[0, \ldots, M]$, enqueues r_i in $\mathcal{Q}_1$, adds its local support $(\left|\mathcal{D}_1^+[s_i]\right|(1-p_0) - \left|\mathcal{D}_1^-[s_i]\right|p_0) \cdot |\mathcal{D}|$ to r_i and sends the result (mod M) to Site 2
6: Sites $2, ..., S-1$ add their local support to the intermediate sum and send the result (mod M) to the next site
7: Site S adds its local support to the sum and enqueues the result, $\tilde{q}_i + r_i$ (mod M), in $\mathcal{Q}_S$
8: **end while**

9: **while** $\mathcal{Q}_1$ contains more than 1 value **do**
10: Site 1 dequeues r_α and r_β from $\mathcal{Q}_1$, generates a random number r' uniformly in $[0, \ldots, M]$ and enqueues r' in $\mathcal{Q}_1$
11: Site 1 generates and encrypts a circuit that computes $(\max(\tilde{q}_\alpha, \tilde{q}_\beta) + r') \mod M)$ from r_α, r_β, $\tilde{q}_\alpha + r_\alpha$, $\tilde{q}_\beta + r_\beta$ and r'. It sends the circuit to Site S together with the cryptographic keys corresponding to the input bits for r_α, r_β and r'.
12: Site 1 sends the encoding table for the remaining inputs to Site T
13: Site S dequeues $(r_\alpha + \tilde{q}_\alpha)$ and $(r_\beta + \tilde{q}_\beta)$ from $\mathcal{Q}_S$, asks Site T for the corresponding cryptographic keys, evaluates the encrypted circuit and enqueues the result, $(r' + \max(\tilde{q}_\alpha, \tilde{q}_\beta)$ mod $M)$, in $\mathcal{Q}_S$
14: **end while**

15: Sites 1 and S calculate q_{max} by exchanging the two remaining values
16: **for** every subgroup descriptions s_i **do**
17: **if** $\tilde{q}_i + r_i \geq r_i + q_{max} \cdot |\mathcal{D}|$ **then return** $\langle s_i, q_{max} \rangle$
18: **end for**

OUTPUT: s_{max} and q_{max} (same output at all sites)

Once the figure q_{max} is calculated, the protocol re-iterates over all subgroups (in Line 4.4) until a subgroup with maximum quality is met. This subgroup is returned as result together with its quality, and the execution ends.

5 Analysis of the Protocol

5.1 Privacy

We will now show that Protocol 1 is secure, as precised by the following theorem:

Theorem 1. *Protocol 1 privately solves Task 1, revealing only $|\mathcal{D}|$ and $|\mathcal{D}^+|$ (in the semi-honest model, assuming no collusion).*

Proof. We have to specify how every site can simulate its *view* given the result, the leaked information and its own input. Recall that the simulation does not need to generate the *exact* same sequence of messages – it suffices that its output is computationally indistinguishable from the view (which involves messages based on random numbers).

The simulator generates execution traces following the algorithmic skeleton of Protocol 1, i.e. by iterating over the subgroup descriptions. This ensures that the simulations have the same overall structure as the views. We will now go over every line of the protocol involving communication and describe how the simulator can generate data computationally indistinguishable from the observed messages. To this end, we will describe how the simulator generates such data in a first paragraph "(S)", before we prove that this data is computationally indistinguishable from the actual messages in a second paragraph "(I)".

Line 4.4: (S) The protocol computes $|\mathcal{D}|$ and $|\mathcal{D}^+|$ using an existing secure-sum sub-protocol, e.g. [6]. Goldreich's *Composition Theorem* [8] says that given secure protocols for sub-tasks, they can be dealt with in a transparent way: all we need to show is that the simulator can predict the outcome of the sub-protocol. Given that $|\mathcal{D}|$ and $|\mathcal{D}^+|$ is part of the simulator's input, it can output these very values. (I) The values in the view and in the simulation are computationally indistinguishable because they are identical.

Line 4.4: (S) The simulator generates the next subgroup using the iterator *iter*. (I) The subgroup in the view and in the simulation coincide, because the traversal is performed in a canonical way.

Lines 4.4 to 4.4: (S) Sites 2 to S generate a random number uniformly in $[0, \ldots, M]$. (I) Given that r_i was randomly generated uniformly in $[0, \ldots, M]$, the local sum in the view is also uniformly distributed in $[0, \ldots, M]$. Hence, it is computationally indistinguishable from the random number in the simulation, because two random numbers generated from the same distribution are computationally indistinguishable [8].

Lines 4.4 to 4.4: The execution of these lines result in the following messages: (i) Site S receives an encrypted circuit representation together with a set of cryptographic keys that allow the evaluation of the circuit; (ii) Site T receives an encoding table, plus the values $(r_\alpha + \tilde{q}_\alpha)$ and $(r_\beta + \tilde{q}_\beta)$.

(S) For *Site* S, the simulator generates a new encrypted circuit, and uses its representation to simulate the circuit in the view. As simulation of the input r', it uses the encryption table to encode the bits of a newly generated random number generated uniformly in $[0, \ldots, M]$. To simulate the other inputs, it uses the encryption table to encode some arbitrary boolean values. For *Site* T, the simulator generates an encrypted circuit and uses the encoding table as simulation of the table in the view. Moreover, it generates two random numbers as simulation of the two inputs from Site S.

(I) For *Site* S, recall that all can be extracted from an encrypted circuit, given a particular set of input keys, is the outcome [15]. The outcome of the circuit in the view is a random number uniformly distributed in the domain $[0, \ldots, M]$, which is independent from all values observed so far (recall that Site 1 generates a *new* random offset in every iteration). The same is true for the circuit in the simulation, thus the view and the simulation are computationally indistinguishable. For *Site* T, the encoding table in the view is computationally indistinguishable from that in the simulation because both essentially consist of a set of cryptographic keys generates by the same cryptographic key generation mechanism. The inputs from Site S in the view are computationally indistinguishable from the random numbers in the simulation because both are uniformly distributed in $[0, \ldots, M]$, and are independent from all values observed so far.

Line 4.4: (S) The simulator generates q_{max}, which it is part of its input. (I) obvious.

Line 4.4: (S) Again, due to the composition theorem the simulator only has to generate the

result of the test (plus, optionally, s_{max}). This is straightforward, given that s_{max} is part of the input. (I) obvious. □

5.2 Complexity Analysis

We will now turn to the runtime complexity of our protocol. As usual in subgroup discovery, the critical factors are the number of attributes and the length limit for the subgroup descriptions: the number of subgroup descriptions visited is bounded by $|C|^L$, where L is the length limit and $|C|$ the number of attribute-value constraints. For every subgroup considered, the cost for the quality and optimistic estimate calculation is linear in the number of records (if conditional datasets are used). While much of the computation can be done in parallel, the transmission of the messages in the first while-loop is carried out sequentially. Each message involves numbers less than M, hence their length is bound by $O(\log(M)) = O(\log(|\mathcal{D}|))$. Summarizing, the runtime complexity of the first while-loop is thus $O(|\mathcal{D}| + S \cdot \log(|\mathcal{D}|))$ per subgroup.

Beside these calculations, the protocol involves the creation, transmission and evaluation of encrypted circuits (in the second while-loop). For every subgroup, these tasks have complexity $O(\log(M)) = O(\log(|\mathcal{D}|))$, because all input to the circuit are integers less than M, and the number of gates in ripple-carry adders and inverters is linear in the number of bits. In terms of $O(.)$ complexity, the cost for the circuits is thus dominated by the quality calculation. Altogether, the complexity of our protocol is thus $O(|C|^L \cdot (|\mathcal{D}| + S \cdot \log(|\mathcal{D}|)))$.

For comparison, let us consider the complexity of centralized approaches, like [9]. For each subgroup considered, these also have costs which are linear in the size of the dataset. Moreover, the number of subgroups to be considered is $O(|C|^L)$ in worst-case.[3] Thus, for a constant number of parties, the worst-case runtime of our protocol is the same as for a centralized, non-secure approach: $O(|C|^L \cdot |\mathcal{D}|)$.

6 Protocols for other Subgroup Discovery Tasks

In this section, we will consider the task to collect not only a single subgroup, but a set of k subgroups. Beside the classical top-k subgroup discovery task, we will consider a second approach that collects k subgroups while avoiding redundancy among those subgroups. Besides protocols that allow collecting a set of subgroups, we will present a heuristic protocol which allows finding subgroups much faster than with Protocol 1, at the price that the optimality of the result is not guaranteed.

6.1 Top-k Subgroup Discovery

It is straightforward to extend our approach to a solution for the top-k subgroup discovery task. All we need to do is to iteratively collect the next highest-quality subgroup *not* in the set of subgroups collected so far. Doing so requires only a minor modification of the iterator, ensuring that all subgroups collected so far will be ignored during subsequent search space traversals. Protocol 2 makes this precise.

This protocol leaks the same information as Protocol 1:

[3] We remark that for many datasets, the number of subgroups considered in non-secure approaches can be considerably reduced by the use of pruning techniques [28, 9]. These branch-and-bound techniques, however, cannot circumvent the worst-case complexity of the approach.

Protocol 2 Top-k Subgroup Discovery

INPUT: integer k, length limit L and local databases $\mathcal{D}_1, \ldots, \mathcal{D}_S$

1: The sites collectively execute Protocol 1. The resulting subgroup and quality are stored in the variables s_1 and q_1
2: **for** i = 2 to k **do**
3: The sites collectively execute Protocol 1, using a modified iterator which skips the subgroup descriptions $\{s_1, \ldots, s_{i-1}\}$. The result is stored in s_i and q_i
4: **end for**

OUTPUT: $\{\langle s_1, q_1\rangle, \ldots, \langle s_k, q_k\rangle\}$ (same output at all sites)

Corollary 2. *Protocol 2 only reveals $|\mathcal{D}|$ and $|\mathcal{D}^+|$ (in the semi-honest model, assuming no collusion).*

Proof. The corollary follows directly from Goldreich's composition theorem and the fact that all s_i are part of the result. □

6.2 The Weighted Covering Scheme

An issue with the above solution is that there is a risk that the subgroups all characterize a similar portion of the data and hence, that there is some redundancy in the result. A classical approach to avoid running into this problem is the *weighted covering scheme* [13]. The idea is to iteratively search for the maximum subgroup, however using a definition of quality that accounts for *record weights*. After every iteration, the weights of the records covered by the subgroup collected so far is decreased by multiplication with some rational number. This results in a definition of quality which is equivalent to the following:

$$q^w(\mathcal{D}, s) = \sum_{i=1}^{S} \left(\left|\mathcal{D}_i^+[s]\right|^w (1 - p_0(\mathcal{D})^w) - \left|\mathcal{D}_i^-[s]\right|^w p_0(\mathcal{D})^w \right). \quad (4)$$

Here, $|\mathcal{D}|^w$ denotes the sum of the weights of the records in $\mathcal{D}$, and similarly $p_0(\mathcal{D})^w = |\mathcal{D}^+|^w / |\mathcal{D}|^w$.

All that needs to be done to implement the weighted covering scheme is thus to make use of this quality definition instead of that in Equation 3. Given that Equation 4 is a sum of local values, and that these values are rational numbers which can be computed locally given the set of subgroups collected in the previous iterations, the adaptation is thus straightforward. Protocol 3 specifies how to securely collect k subgroups according to the weighted covering scheme.

Again, this protocol leaks the same information as Protocol 1:

Corollary 3. *Protocol 3 only reveals $|\mathcal{D}|$ and $|\mathcal{D}^+|$ (in the semi-honest model, assuming no collusion).*

Proof. As before, this follows directly from Goldreich's composition theorem and the fact that all s_i are part of the result. □

Protocol 3 Weighted Covering

INPUT: integer k, length limit L and local databases $\mathcal{D}_1, \ldots, \mathcal{D}_S$

1: The sites collectively execute Protocol 1 to find s_1 and q_1
2: **for** i = 2 to k **do**
3: All sites calculate the weights of their records based on $\{s_1, \ldots, sd_{i-1}\}$, the set of subgroups collected so far
4: The sites collectively execute Protocol 1, using the modified definition of quality from Equation 4. The result is stored in s_i and q_i.
5: **end for**

OUTPUT: $\{\langle s_1, q_1\rangle, \ldots, \langle s_k, q_k\rangle\}$ (same output at all sites)

6.3 Pruning

The cryptographic techniques used in our protocol, in particular the circuit encryption and evaluation, are computationally very expensive operations. The effect is that even if according to the complexity analysis from Section 5.2 the protocols scale similarly to non-secure solutions in the worst case, their execution time is several orders of magnitude higher in practice (as we will show in Section 7.2). To cope with this issue, we will now consider the use of pruning techniques within our approach.

Unfortunately, pruning is problematic from the perspective of secure multi-party computation. In fact, the incorporation of *any* strategy which avoids visiting parts of the search space *based on intermediate computation* results in a security leak: The reason is that whether a branch is pruned or not can clearly be observed by all participants. However, this information cannot be predicted solely based only on the information about top-1 subgroup. Hence, it is not possible to simulate an execution trace involving pruning, which violates the definition of secure computation.

This consideration implies that the use of optimistic estimate pruning [28]—a standard technique in non-secure subgroup discovery—is not without problems in our secure setting. In fact, optimistic estimate pruning is particularly problematic, as it typically prunes a high number of branches. Thus, it is not easy to quantify how much information is leaked, or, considered from a different perspective, it is difficult to quantify how much information would be needed to simulate an execution trace. Beside this issue, even if optimistic estimate reduces the runtime, it will still be much higher than for a centralized approach. The same reasoning applies to pruning approaches based on the theory of closed itemsets [20, 3].

For this reason, in this section we consider a different pruning approach, namely a greedy search. This search strategy finds the best subgroup of length 1, and from here only considers specializations of the best subgroup found in the previous levels. This approach has two advantages: First, instead of considering $O(|C|^L)$ subgroups, it only considers $O(|C| \cdot L)$ subgroups (again L denotes the length limit and $|C|$ the number of attribute-value constraints). Moreover, compared to optimistic estimate pruning, this technique has the advantage that the amount of information leaked can precisely be quantified: essentially, it consists of information about $L - 1$ subgroups. Of course, the price is that the result is no longer guaranteed to be optimal.

The greedy search is specified in Protocol 4. As already mentioned, at level i the protocol only considers specializations of the best subgroup found at level $i - 1$. The result of the protocol is the subgroup with the highest quality thereby visited (the longest subgroup

description needs not have the highest quality, which means that the result needs not be the subgroup found at the last level).

Protocol 4 Greedy Top-1 Subgroup Discovery

INPUT: length limit L and local databases $\mathcal{D}_1, \ldots, \mathcal{D}_S$

1: The sites collectively execute Protocol 1 with length limit 1 to find $\tilde{s}_1$ and $\tilde{q}_1$
2: **for** i = 2 to L **do**
3: The sites collectively execute Protocol 1 with length limit i, using a modified iterator which only considers specializations of $\tilde{s}_{i-1}$. The resulting subgroup and quality are stored in the variables $\tilde{s}_i$ resp. $\tilde{q}_i$.
4: **end for**

OUTPUT: $\langle \tilde{s}_{i*}, \tilde{q}_{i*} \rangle$, such that $i* = \arg\max_{i=1,\ldots,L} \tilde{q}_i$ (same output at all sites)

Corollary 4. *Protocol 4 only reveals $|\mathcal{D}|$, $|\mathcal{D}^+|$, a set of L subgroups $\tilde{s}_1, \ldots, \tilde{s}_L$ such that $\tilde{s}_1 = \arg\max_{s \in \mathcal{L} | length(s)=1} q(\mathcal{D}, s)$ and $\tilde{s}_i = \arg\max_{s \in \mathcal{L} | s \supset \tilde{s}_{i-1} \wedge length(s)=i} q(\mathcal{D}, s)$ (for $i = 2, \ldots, L$), and the qualities of the subgroups $\tilde{s}_1, \ldots, \tilde{s}_L$ (in the semi-honest model, assuming no collusion).*

Proof. Due to the composition theorem, the simulator only needs to simulate the messages involving the $\tilde{s}_i$ and their quality. This is precisely the leaked information listed in the corollary. □

Of course, it is possible to combine the greedy protocol with the weighted covering approach to obtain a set of subgroups.

7 Implementation and Experimental Evaluation

In this section, we will briefly describe our prototypical implementation and report on some experiments.

7.1 Prototypical Implementation

Our prototype was implemented in Java. For the encryption of the circuits, we used the AES cipher with 128 bit keys implemented in the lightweight API of the crypto-library Bouncycastle (http://www.bouncycastle.org). To compute secure sums, we used the secure sum protocol described in [6]. For secure comparisons, we used the efficient solution for Yao' millionaires problem described in [25].

7.2 Experimental Evaluation

We investigated two questions: first, how does the runtime of our protocol compare to the runtime of non-secure (centralized) subgroup discovery algorithms? And second, which part of the protocol is critical for the runtime. To this end, we evaluated the performance of our implementation on different datasets from the well-known UCI repository [1]. The datasets and their characteristics are listed in Table 1. All datasets where randomly split into three parts, which were used as local datasets. The experiments were performed on three Core 2 Duo E8400 PCs with 2GB RAM, connected by an Ethernet LAN. The length limit, L, was set to 3.

dataset	credit-g	nursery	sick	soybean
number of constraints	58	27	66	133
number of records	1000	12960	3772	683

Table 1: Datasets

7.2.1 Comparison of the Different Approaches

Figure 2 shows the result of the comparison of our two secure protocols with a non-secure algorithm. As representative for the latter class of approaches, we chose the algorithm DpSubgroup [9]. The figure shows that compared with this algorithm, both secure protocols are slow. Protocol 1 (labeled as "SecureSD" in the figure) is actually slower by more than four orders of magnitude. Protocol 4 ("GreedySD") is considerably faster, but still about two orders of magnitude slower than the non-secure algorithm DpSubgroup.

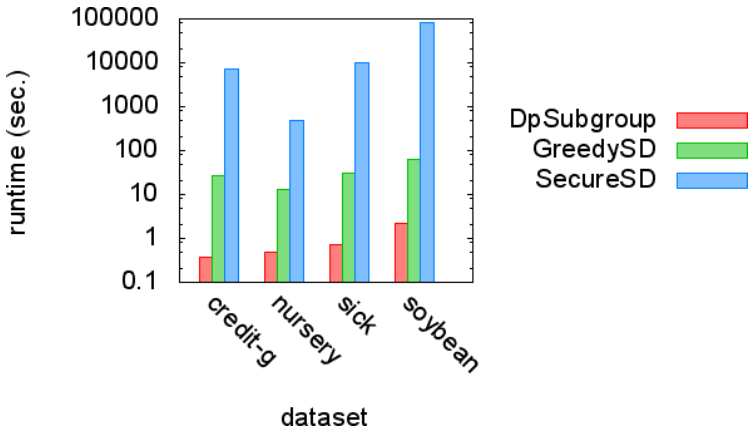

Figure 2: Runtime of the prototype

A different yet interesting issue is how good the subgroups found by the protocols are, compared with a non-secure solution. For Protocol 1, the result is exactly the same, as shown in Theorem 1. So the question is how well the greedy algorithm performs. Clearly, a representative investigation of the qualitative performance of greedy vs. exhaustive search is not the focus of this paper. Nevertheless, we note that in all our experiments, the greedy top-1 protocol found one of the top-2 subgroups (as found by the exhaustive top-k protocol). This shows that even though the greedy approach typically won't find the exact solution, it often comes up with interesting subgroups.

7.2.2 The Costs of the Encrypted Circuits

Finally, we investigate the costs of the different parts of our secure protocols. Figure 3 compares the runtime of the following two components of Protocol 1: the while loop involving the encryption and evaluation of the circuits is labeled as "Circuits". All the rest, in particular the quality computation and comparison, is labeled as "Qual. calc.". The figure reveals that the loop involving the encrypted circuits is by far the most expensive part of

the computation, consuming more than 95% of the total runtime.

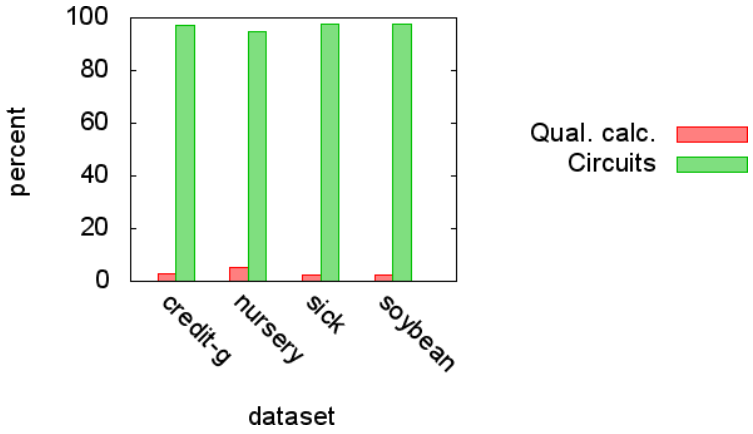

Figure 3: Share of the runtime spent for circuit encryption and evaluation

7.2.3 Summary of the Results

Summarizing, the experiments show that our approach is applicable in practice. The runtime is sufficient to process data sets of realistic size in a few hours (or minutes, if the greedy approach is adopted). This is quite often sufficiently fast for practical use. In scenarios where a failure to guarantee privacy means that data mining can not be applied at all, the users may very well be willing to invest more time if that allows finding valuable patterns which could not be obtained otherwise.

8 Conclusions

In many pattern mining settings, the data to be analyzed is sensitive, which makes the use of privacy-preserving techniques mandatory. Although approaches exist to securely find association rules in distributed data, these cannot be adapted to supervised descriptive mining tasks like subgroup discovery. The source for the difficulties are precisely the features that distinguish subgroup discovery from classical association rule mining: (i) the different quality function, and (ii) the aim to collect only k patterns.

In this paper, we have presented new secure protocols that allow secure subgroups discovery on horizontally partitioned data. While the basic protocol only solves the top-1 subgroup discovery task, it can be iterated to collect a set of k subgroups. We have analyzed the properties of our protocol and have shown that it leaks only little information, namely the size of the database and the share of positive records. Moreover, we have reported on a prototypical implementation and runtime experiments with that implementation.

In the experiments it has become clear that the improvements in security and privacy come at the price of a high runtime. While the worst-case complexity of our algorithm is the same as for a non-secure solution (i.e. exponential in the length limit), in practice it is much slower than the latter. One key reason is the high computational overhead for the generation and evaluation of the encrypted circuits. As resort, we have considered a

modified protocol which applies a greedy search. While this approach comes at a price (the solution is not guaranteed to be optimal and the information leakage is somewhat higher), it allows to reduce the runtime by several orders of magnitude. This can allow the use of privacy-preserving subgroup discovery in settings where the first protocol would be too slow.

One question worth further investigation is the effective severity of information leaks caused by pruning techniques. Clearly, knowledge about e.g. optimistic estimates tells something about the private data, but it is not really clear how much. In particular, does it allow to reconstruct (part of) the data? This question is closely related to the so-called task of *inverse frequent set mining* [16]. Another interesting question is whether the protocols presented in this paper can be adapted to other quality functions used in supervised descriptive rule discovery. Yet another question is whether it is possible to devise a custom-made protocol for the secure maximum computation as replacement for our (costly) encrypted circuit solution. Finally, it would be desirable to extend the security guarantees to colluding parties. One standard approach in the cryptographic community to deal with collusion issues is to divide the information into different parts, and to use different routes for the different calculations (e.g. [11]). We leave the investigation of these issues to future work.

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *The VLDB Journal*, 17(4):703–727, 2008.

[3] M. Boley and H. Grosskreutz. Non-redundant subgroup discovery using a closure system. In *ECML/PKDD*, 2009.

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. Peckham, editor, *SIGMOD Conference*, pages 265–276. ACM Press, 1997.

[5] D. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. *Parallel and Distributed Information Systems, International Conference*, 0:0031, 1996.

[6] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2), 2002.

[7] J. Fürnkranz and P. A. Flach. Roc 'n' rule learning-towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.

[8] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.

[9] H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *ECML/PKDD (1)*. Springer, 2008.

[10] W. Hämäläinen and M. Nykänen. Efficient discovery of statistically significant association rules. In *ICDM*, pages 203–212. IEEE Computer Society, 2008.

[11] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[12] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*. 1996.

[13] N. Lavrac, B. Kavsek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5(Feb):153–188, February 2004.

[14] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*. Springer, 2000.

[15] Y. Lindell and B. Pinkas. A proof of yao's protocol for secure two-party computation. Technical report, 2004.

[16] T. Mielikäinen. On inverse frequent set mining. In *Workshop on Privacy Preserving Data Mining*, 2003.

[17] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, pages 226–236. ACM, 2000.

[18] S. Nijssen, T. Guns, and L. D. Raedt. Correlated itemset mining in roc space: a constraint programming approach. In *KDD*, pages 647–656, 2009.

[19] P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10, 2009.

[20] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25 – 46, 1999.

[21] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explor. Newsl.*, 4(2):12–19, 2002.

[22] P. Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6), Nov/Dec 2001.

[23] M. Scholz. On the tractability of rule discovery from distributed data. In *ICDM*, pages 761–764. IEEE Computer Society, 2005.

[24] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. In *ICDM Workshops*, pages 541–545, 2006.

[25] L. Shundong, D. Yiqi, W. Daoshun, and L. Ping. Symmetric encryption solutions to millionaire's problem and its extension. In *1st International Conference on Digital Information Management*, 2006.

[26] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644. ACM, 2002.

[27] G. I. Webb. Discovering significant rules. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 434–443, New York, NY, USA, 2006. ACM.

[28] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In J. Komorowski and J. Zytkow, editors, *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87. Springer, 1997.

[29] M. Wurst and M. Scholz. Distributed subgroup mining. In *PKDD*, 2006.

[30] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*. IEEE, 1982.

[31] A. C.-C. Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1985., 27th Annual Symposium on*, pages 162–167, Oct. 1986.

[32] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, 2006.