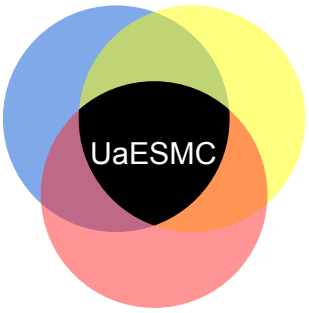

Project N°: **FP7-284731**

Project Acronym: **UaESMC**

Project Title: **Usable and Efficient Secure Multiparty Computation**

Instrument: **Specific Targeted Research Project**

Scheme: **Information & Communication Technologies**

Future and Emerging Technologies (FET-Open)

Deliverable D5.2.1
Scientific Progress Analysis and Recommendations

Due date of deliverable: 31st January 2013

Actual submission date: 31st January 2013

Start date of the project: **1st February 2012** Duration: **36 months**

Organisation name of lead contractor for this deliverable: **CYB**

Specific Targeted Research Project supported by the 7th Framework Programme of the EC		
Dissemination level		
PU	Public	✓
PP	Restricted to other programme participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

Executive Summary:
Scientific Progress Analysis and Recommendations

This document summarizes deliverable D5.2.1 of project FP7-284731 (UaESMC), a Specific Targeted Research Project supported by the 7th Framework Programme of the EC within the FET-Open (Future and Emerging Technologies) scheme. Full information on this project, including the contents of this deliverable, is available online at `http://www.usable-security.eu`.

The work in UaESMC project is guided by a set of example problems that have been extracted from the interviews with end users conducted in Task 1.2, as well as from the experience of the project participants. In this report, we describe these problems. The first of them is the privacy-preserving statistical analysis of structured data, belonging to several different data owners and being split between them both horizontally and vertically. In real-world tasks, the quality of analysed data is not perfect and the methods must be able to cope with missing values, obviously wrong values, etc. The analysis suite must be sufficiently general to handle these issues, while at the same time providing a rich set of data analysis tools.

The second set of problems concerns privacy-preserving optimization. We have selected linear programming as the centre of our focus. Here the existing approaches have been twofold — either implement an algorithm for linear programming in some SMC framework, or scramble the problem in some manner, such that from the solution to the scrambled problem it is easy to deduce the solution to the original problem. The first approach is slow and the second has so far lacked a satisfactory security analysis. In UaESMC, we would like to obtain the benefits of both approaches. Besides linear programming, we also consider other optimization tasks, including NP-hard problems and privacy-preserving (heuristic) methods for solving them.

The third set of problems stems from KTH's work and concerns the privacy-preserving operation and management of networks. Again, there are several subproblems, e.g. shortest paths between nodes, simulation of attacks, management of reputations, etc., which have to be solved in privacy-preserving manner. The last set of problems concerns the interplay between game theory and cryptography (but note that there are significant mechanism design questions also in other sets), where the non-repudiation property may contribute to new ways of designing mechanisms.

Through the selection and refinement of these problems, we have reached Milestone 1 of the project.

In the first year, we have made progress in all areas listed above. We have used the Sharemind SMC framework to implement and optimize various tools for data analysis. We have proposed a transformation for linear programs, where the preservation of privacy follows from concrete hardness assumptions. We have studied the use of privacy-preserving genetic algorithms for various optimization tasks, as well as privacy-preserving graph algorithms, usable for network management. We have proposed composable notions of truthfulness in games and shown how to obtain them.

In the second year, our work plans are centered on the same problem areas. Additionally, we will consider how our results will come together, forming the UaESMC framework. This report details those plans.

List of Authors

Dan Bogdanov (CYB) Yiannis Giannakopoulos (UoA) Roberto Guanciale (KTH)
Liina Kamm (CYB) Peeter Laud (CYB) Pille Pruulmann-Vengerfeldt (UT)
Riivo Talviste (CYB) Kadri Tõldsepp (UT) Jan Willemson (CYB)

Contents

1 Introduction **4**

2 Selected Problems **6**
2.1 Statistical Analysis of Structured Data . . . 6
2.1.1 Problem Description . . . 6
2.1.2 Measuring the performance . . . 7
2.1.3 Performance goals . . . 7
2.2 (Integer) Linear Programming . . . 7
2.2.1 Problem Description . . . 8
2.2.2 Measuring the Performance . . . 8
2.2.3 Performance Goals . . . 8
2.2.4 Shortest Paths in Graphs . . . 8
2.3 Network Management . . . 9
2.3.1 Global minimal cost . . . 9
2.3.2 QoS in multihomed gateways . . . 9
2.3.3 BGP verification . . . 10
2.3.4 Monitoring and replaying attacks . . . 10
2.4 Cryptographic Support for Mechanism Design . . . 10

3 Tasks Worked on During Year 1 **13**
3.1 Optimization . . . 13
3.1.1 Linear programming through problem transformation . . . 13
3.1.2 Privacy-Preserving Genetic algorithms . . . 14
3.2 Statistical Analysis . . . 15
3.2.1 Oblivious Sorting . . . 15
3.2.2 Implemented Statistics and Tests . . . 16
3.2.3 Data Classification . . . 16
3.3 Strong Truthfulness . . . 16

4 Work Plans for Year 2 **19**
4.1 Optimization . . . 19
4.1.1 All-Pairs Shortest Paths . . . 19
4.2 Statistics API . . . 20
4.3 Mechanism Design for Statistical Data Collection . . . 20
4.4 Mechanisms for Playing Games . . . 20
4.5 Progress towards the UaESMC Framework . . . 21

Bibliography **22**

Chapter 1

Introduction

The goal of UaESMC is to increase the use of secure multiparty computation (SMC) techniques both in numbers and in variety. The project works toward this goal by looking for real-life problems that could most benefit from SMC techniques, by determining the reasons why these techniques are not used, by proposing solutions that overcome these reasons, and by demostrating the usefulness of these solutions. This workplan requires tight coordination between the partners to make sure that the project is on track and the research directions of the partners are well-aligned.

One of the most significant steps during the first year of the project has been the conducting of interviews with the representatives of various fields of activity. The interviews, reported and analysed in Deliverable 1.2 [11], were held with the aim to identify multi-party computational problems relevant to these fields, where the security requirements of the data owners are currently hindering the actual computation, and where a solution to satisfying these requirements and allowing the computations to take place would benefit the parties. The interviews showed certain problems common to a number of fields. The project consortium has extracted and refined these problems from the interviews. The problems are reported in Chapter 2 of this deliverable. The extraction of common problems marked that milestone MS1 of the project had been reached. The problems have been further refined and extended in Deliverable 4.1 [9]. The extension of the problems and their embodyment in application scenarios has identified further privacy-preserving tasks to look at.

The interviews showed that in computationally similar problems, the providers of inputs and receivers of outputs may be partitioned quite differently leading to different requirements to protect certain data from certain parties. Hence, When stating the common problems, the generality with respect to security and privacy requirements was seen as important. This has lead to the following commonality in the problem statements. Let $y = f(x_1, \ldots, x_n)$ be the function to be computed on inputs $x_1, \ldots, x_n$. We assume that the protocol set for SMC, run by the parties, is such that some kind of *private storage* is implemented. Values held in this private storage can become public only if several of the computing parties cooperate. The private storage may be implemented through secret sharing, or through threshold homomorphic encryption, or through fully homomorphic encryption if none of the computing parties knows the decryption key. When stating the problems of private computation, we will typically require that

- at the beginning of the computation, $x_1, \ldots, x_n$ are in the private storage;
- at the end of the computation, y will be added to the private storage;
- nothing about $x_1, \ldots, x_n$ (or about the intermediate values during the computation of f) will become public.

After the computation, y may be made public, or public to some of the parties, or kept private and used as input to some other computation. Private computation tasks implemented in such manner are *universally composable*, meaning that the security of some composition of these tasks follows from the security of each task separately. The use of universally composable building blocks makes the building of complex, but secure functionalities conceptually easy.

According to the description of work of UaESMC, further theoretical work in the project should cluster around the selected problems. This has indeed taken place. Chapter 3 gives an overview of the tackled theoretical problems; the achieved results are more thoroughly described in deliverables D2.2.1, D3.2 and D4.1. In all presented protocols and techniques, one can see a strong drive towards *composability*. Chapter 3 also describes and analyses the achieved performance of the proposed protocols. The work on selected problems will continue in the next year, the plans are described in Chapter 4. The inter-partner cooperation has been greatly helped by the biweekly meetings over instant messaging, attended by all project partners.

Chapter 2

Selected Problems

2.1 Statistical Analysis of Structured Data

The analysis of sensitive data combined from several registries and databases held by different entities was the task that came up most often in the interviews [11]. A typical sequence of steps in performing such an analysis might be the following.

- Determining the customer of the study, the statistical questions to be answered, the data owners, the available SMC infrastructure.
- Obtaining the data owners' agreement to participate in the study.
- Uploading the data to the storage offered by the SMC infrastructure.
- Running the statistical analysis with the SMC techniques.
- Making the results of the analysis known to the customer of the study.
- Disposing the stored data.

Often, the interviewees did not have a clear idea of exactly which analyses they would run if such combination were possible, as they had not thought about it yet. Hence the SMC problem we are going to select must be sufficiently generic, in order to cover the actual problems we may consider in year two of the project.

A common issue with the usage of SMC for statistical analysis was reported to be the data analyst's desire to "see" the data; to get a feel of what it looks like before starting the actual analysis, in order to have some ideas on which analyses and parameter selections make sense for this dataset. We have decided to interpret this desire as the analyst's wish to see some simple statistics of the dataset, e.g. means, medians, standard deviations, histograms of single parameters. Having the data analyst "see" the data introduces more steps to the analysis, which may make it more complicated. It also introduces new opportunities in reusing the computations that have already been made.

2.1.1 Problem Description

In this problem, we are analysing data coming from several different sources, split both horizontally and vertically. Our data model consists of several different tables. The tables have to be joined and the analysis performed on the view obtained. The view maybe quite large. The quality of the data may be varying (e.g. there may be missing values). Besides the columns present in one of the tables, there may also be derived columns (sums, averages, normalized values, etc.).

On the joined view, various statistics mentioned above must be computed. Other operations of data analysis to be implemented include

- Detection of errors in the data. The list of detected errors must remain private, in order to be composable with subsequent analysis.

- Various regressions and tests, including

  - Linear and various kinds of non-linear regression;
  - T-tests;
  - Significance testing;
  - Factor and cluster analysis.

The suite of statistical tests, including the necessary privacy for the result of each test will be specified in deliverable D4.1.

The techniques we develop for privacy-preserving analysis of structured data should also use the structure of data, as it is public. The structure may be used to pick a more efficient strategy for making the database queries, for building the view and for performing the actual computations.

Depending on the application, there may be the issue of obtaining truthful data from data owners. We need locate the incentives for inputting truthful data, and to devise sufficiently general mechanisms for acting on these incentives.

2.1.2 Measuring the performance

In practice, we are interested in the size of database that can be statistically analyzed using secure multiparty computation. Therefore, the main performance metric we are interested in, is the time that it takes to securely process an input database of the given size.

To test the performance of a secure statistical algorithm, we set up an input database of the given size. Note that to test simpler algorithms, we can replace the database with an input vector or table. We will then run the algorithm with this input on a secure multiparty computation system.

Our statistical algorithms are implemented on the Sharemind platform. The performance measurement tools and methodology for Sharemind algorithms and protocols have been described in Section 4.2 of [4].

2.1.3 Performance goals

The statistical processing algorithms that UaESMC is developing, are expected to operate with input databases ranging from 100 to 100 000 values in practice. Therefore, we set the minimal goal to be to process 100 000 values in reasonable time. The optimistic goal is to process databases of one million values.

More specifically, we want to be able to complete a statistical study on a 100 000-value database in a day. We assume that in this setting, Sharemind is deployed in a near-optimal setting, with fast network links. One day is a reasonable amount of time, as statistical studies typically set time aside for interpreting the results.

Also, paying the price of running secure multiparty computation for one day is often better than not being able to process the data at all.

2.2 (Integer) Linear Programming

A number of interviews mentioned optimization problems with sensitive inputs coming from different parties as an SMC task. We anticipate that the exact nature of the optimization problems will vary widely among different applications, hence we have selected a generic optimization problem as one of the example problems of UaESMC.

2.2.1 Problem Description

A linear programming task is of the form "minimize $\mathbf{c}^T \cdot \mathbf{x}$ subject to the constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, where $\mathbf{A}$ is a $m \times n$ matrix (with integer or real entries), $\mathbf{b}$ is a vector of length m, $\mathbf{c}$ is a vector of length n, and $\mathbf{x}$ is the vector of unknowns, also of length n. The goal is to pick the values for the elements of $\mathbf{x} = (x_1, \ldots, x_n)^T$, such that the dot product of $\mathbf{c}$ and $\mathbf{x}$ is minimized, while all the constraints are satisfied. The values of the variables are real numbers.

The task may have further constraints by requiring that the values of some of the variables x_i are integers.

In the multiparty setting, we consider each party P_i to provide a matrix $\mathbf{A}_i$, and vectors $\mathbf{b}_i$ and $\mathbf{c}_i$. The actual matrix $\mathbf{A}$ and the vectors $\mathbf{b}$ and $\mathbf{c}$ are the sums of the respective matrices and vectors of the parties. This implies that the size of the problem (number of constraints and variables) is public. This setting is very general and covers both the horizontal and vertical splitting of constraints. Because of this flexibility and high privacy requirements, we expect any privacy-preserving solution to construct $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ in a private domain and then apply a solution algorithm that does not leak data out of this domain. The privacy-preserving linear programming problem in this setting has been considered before [1], but existing algorithms are either slow or have unsatisfactory security analyses.

The sizes m and n of the task are public. Regarding the integrality constraints, we assume those to be public as well.

We will also consider the case where each party wants to see a different objective function maximized, and is prepared to provide incorrect data to the computation in order to achieve this. If payments are possible between players then controlling the behaviour of players will be a classical task in mechanism design.

2.2.2 Measuring the Performance

The sufficiency of the performance of an optimization task is dependent on the user. If the optimization result must be delivered in real time, the secure algorithm must be very fast. If the search/optimization task is set up for just a few uses, it may be acceptable if the task takes longer.

An example of a real-time search requirement would be shortest path routing (imagine navigation applications for cars or public transport routing systems). An example of a non-real-time application would be a system that collects business transactions from several parties and uses them to search for certain rules or patterns that could help the parties make decisions.

2.2.3 Performance Goals

Both of the prototype implementations in UaESMC are for tasks that could ideally be solved in a few seconds to end users. However, given that both are complex tasks, we are reluctant to promise real-time results for both. The current goal is to determine how big problems can we solve in one minute, five minutes, one hour, and one day.

2.2.4 Shortest Paths in Graphs

We consider separately an optimization problem that may be the most prevalent among our interviewees, as well as elsewhere in the society. Also, its structure may allow for more efficient solutions than what can be obtained from its reduction to the linear programming problem. Finding shortest paths in graphs (either with single source and target, or the all-pairs problem) is the optimization problem that we are going to consider separately.

In this problem, we assume the graph to be public. Also, the vertices between which the shortest path must be found, are public. The edge lengths, however, are private.

The privacy requirements of the task may be different in different settings. In one setting (see Sec. 2.3), the computing parties are identified with the nodes of the graph. Each party knows the lengths of the edges incident to it. As the result of the computation, each party should learn the next step that must be made

from it towards the target node. Currently, we do not know any more efficient methods for this problem than the generic ones.

2.3 Network Management

Privacy-preserving management / operation of a cluster of networks is in some sense an application for the optimization algorithms that UaESMC will develop. On the other hand, it will present its unique challenges due to the timing and connectivity constraints. This set of problems has been inspired by the work done by KTH in this area.

Internet packets are delivered through a complex network of Internet Service Providers (ISPs). Each ISP owns an Autonomous System (AS) that participates in the Border Gateway Protocol (BGP). The Border Gateway Protocol is exploited to make inter-AS routing decisions, allowing each ISP to independently enforce its own policy rules and to locally compute its preferred intra-AS routing strategy. Several scenarios can benefit from more collaborative ASs.

2.3.1 Global minimal cost

ISPs are commercial entities that negotiate political agreements. Each ISP has its own cost for packets that traverse its network and can be taken into account in the selection of routing strategies. An undirected graph can be used to encode the inter-AS topology, in such a way that vertices correspond to the border gateways of ASs and edges correspond to adjacent gateways (either intra- or inter-AS). Each edge is labeled with the cost related to the traversing traffic.

The problem of computing the optimal routing is already solved in a world where each AS knows the running cost of all edges of the graph. The real world is more complex and ASes have only a partial knowledge of the costs:

- Vertices of the network are partitioned into ASes in order to represent multihomed domains.
- The cost of an edge connecting vertices belonging to the same AS can be computed by the ISP itself, by running a local routing protocol. This cost is a commercial secret that the ISP does not want to share with the competitors/customers.
- The cost of edges crossing AS boundaries are regulated by contracts. They are known by the ISPs operating these two ASs, but cannot be discovered by the other ISPs.

The goal of each AS is to compute, for each destination vertex d and an internal source vertex s, the vertex to which the packed has to be delivered in order to reach the best path and the corresponding total cost. For making computations based on these distributions, all-to-all connectivity is available. This might be used for setting up routing tables. When a particular packet is being handled, the routing decision must be done using local information only.

Hence we have an instance of the all-pairs shortest path problem, with a particular structure of the graph and the knowledge of its edge lengths.

2.3.2 QoS in multihomed gateways

The above scenario can also be used to represent QoS aware routing problem, where the costs represent service metrics. However it is not sufficient to deploy a full QoS aware routing. In fact, the inter-AS routing problem takes into account only the costs of delivering packets between border gateways, forcing AS to rely on selfish routing techniques. The inter-AS best path is computed regardless of the intra-AS paths, then each AS adopts its own strategy to route the packet inside its own network. This approach represents a limitation to find a global optimal solution whenever multi-homed border gateways are involved, since the computed paths represent only an optimum for outbound traffic. The uncoordinated routing decisions can produce congestion on distant inter-domain links. Statically computing global routing paths is unfeasible

due to the number of intra-nodes involved. Moreover, it is not possible to run a routing algorithm whenever a packer is being handled.

To handle these problems a QoS model based on traffic class (TC) aggregates is usually involved. ASs locally monitor service levels for each class. Whenever a service level violation or a significant quality metric change occur, the ASs are allowed to exploit all-to-all connectivity to compute and update the routing path of the specific traffic class.

2.3.3 BGP verification

The proper functioning of this infrastructure depends on the veracity of BGP routing information, forwarded among ASes. Misconfiguration and attacks can be propagated, causing instability and severe faults (e.g. the AS7007 incident [5]). In order to prevent these accidents, reputation networks that exploit the real-world trust relationships among ISPs have been proposed [8]. While BGP allows to verify the information source by using standard cryptographic techniques, it does not support the verification the truth of what ASs actually say, forcing the ASs to blindly trust the received information. The reputation network can support decisions in conflicting scenarios, since it is impossible to detect all BGP faults affecting a single AS locally at that AS.

These reputation networks are usually built on top of e-voting protocols, enriching the existing solutions to allow each participant to obtain a different election result. The essential properties of such systems are

(i) proved mechanism to discourage malicious behavior and encourage trust;

(ii) confidentiality of the votes and the business relations; and

(iii) proved security against malicious attackers.

The data to protect here are the reputation scores different nodes have for each other.

2.3.4 Monitoring and replaying attacks

A network attack often hits multiple ASs simultaneously. To recognize the attack while it is taking place, or to analyse it afterwards, the network logs of multiple ASs have to be combined. A standard method for replaying an attack is to build a network model, typically in the form of a Petri net, and to apply each attack message to this model in the same order it occurred.

The replaying can take place in the private domain. This requires good methods for privacy-preserving executions of control-intensive applications. Alternatively, the combined log of several ASs could be sorted, made public and replayed in the public domain.

If the sorted log can be replayed in public, then the result of the private replay can probably also be made public after each step. This will considerably simplify the private execution.

A separate issue in network log analysis is the distinguishing of the log items belonging to different attacks going on at the same time. This can also be more precise if logs from several ASs can be used.

2.4 Cryptographic Support for Mechanism Design

Using SMC techniques to implement trusted mediators picking correlated equilibria has been exactly one of the two major parts of the work that has been done so far in bringing together SMC and Game Theory. This work has definitely been interesting, but even greater integration may be possible. In UaESMC, we want to come up with mechanisms that force the players of a game to report their true valuations to the mediator.

For a clearer exposition[1] let's assume we have a 2×2 (normal form) game, i.e. two players each one having two available strategies. This bimatrix game can be represented by two 2×2 matrices, $A = (a_{ij})$

[1]The discussion can be easily adapted to general n-players m-strategies games, $n, m \geq 2$.

and $B = (b_{ij})$. Player 1, also called the row player, assigns probabilities p_i to rows $i = 1, 2$, $p_1 + p_2 = 1$ and player 2, called the column player, assigns probabilities q_j to columns $j = 1, 2$, $q_1 + q_2 = 1$. An *outcome*, also called strategy profile, of the game is just a tuple (p, q) of the players' selected probability distributions $p = \{p_i\}_{i=1,2}$ and $q = \{q_j\}_{j=1,2}$. At a given outcome (p, q), player 1 receives an (expected) utility of $\sum_{i=1}^{2}\sum_{j=1}^{2} p_i q_j a_{ij}$ and player 2 a utility of $\sum_{i=1}^{2}\sum_{j=1}^{2} p_i q_j b_{ij}$.

Now, on top of this traditional game setting, consider a *mechanism design*[2] setting where the player's types are these matrices A and B and, given such (possibly mis-)reports $\tilde{A}$ and $\tilde{B}$ of these matrices by the players, a mechanism computes a Nash equilibrium (p, q) of the bimatrix game $(\tilde{A}, \tilde{B})$, the players receiving valuations equal to their *true* utilities on the outcome (p, q) of the underlying game (A, B) and also the mechanism may also give payments π_1 and π_2 back to the players for taking part at the mechanism. More formally, given as input the bimatrix game $\tilde{\mathcal{G}} = (\tilde{A}, \tilde{B})$ the mechanism computes an allocation $\alpha(\tilde{\mathcal{G}}) = (\alpha_1(\tilde{\mathcal{G}}), \alpha_2(\tilde{\mathcal{G}})) = (p, q)$, where p, q are probability distributions over support $\{1, 2\}$ constituting a Nash equilibrium of the game $\tilde{\mathcal{G}}$, and payments $\pi(\tilde{\mathcal{G}}) = (\pi_1(\tilde{\mathcal{G}}), \pi_2(\tilde{\mathcal{G}})) = (\pi_1, \pi_2) \in \mathbb{R}$. Then, the players end up having utilities

$$u_1(\tilde{\mathcal{G}}|A) = \sum_{i=1}^{2}\sum_{j=1}^{2} p_i q_j a_{ij} + \pi_1 \qquad \text{and} \qquad u_1(\tilde{\mathcal{G}}|B) = \sum_{i=1}^{2}\sum_{j=1}^{2} p_i q_j b_{ij} + \pi_2$$

respectively and, since they are rational and selfish utility maximizers, they will (mis)report the $\tilde{A}$ and $\tilde{B}$ that maximize their own utilities. Note that, while players report the bimatrix game $\tilde{\mathcal{G}}$ and the Nash equilibrium is computed for this game, the valuations the players receive are given by their true utilities for the *initial* game $\mathcal{G} = (A, B)$ and so they need to strategize and possibly lie in order to move the allocation of the mechanism to a better outcome for game $\mathcal{G}$ which is private information for the players.

As an example, to make our model even clearer, consider the following well known "Battle of Sexes" game and assume mechanisms with no payments. In this concise representation we have just merged matrices

		Boy	
		opera	*football*
Girl	*opera*	**3, 2**	1, 1
	football	0, 0	2, 3

Table 2.1: Battle of Sexes

$A = \left(\begin{smallmatrix} 3 & 1 \\ 0 & 2 \end{smallmatrix}\right)$ and $B = \left(\begin{smallmatrix} 2 & 1 \\ 0 & 3 \end{smallmatrix}\right)$. So, let's say that we have a mechanism that given this true input $\mathcal{G} = (A, B)$ will compute the (pure) Nash equilibrium $((1, 0), (1, 0))$ (see the bold top left corner of Table 2.1). At this outcome, they both end up going to the opera, the girl receiving a "happiness" of 3 and the boy receiving 2. It is indeed a Nash equilibrium: no-one wants to unilaterally (i.e. given that the other will not deviate) change her/his strategy. But, this outcome results to an imbalance at the players' gained utilities: The girl ends up with 3 and the boy with 2. So, the boy has an incentive to misreport $\tilde{B} = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 3 \end{smallmatrix}\right)$. The resulting game $\tilde{\mathcal{G}} = (A, \tilde{B})$ would have only one Nash equilibrium, the $(0, 1), (0, 1)$, so the mechanism will output this. This outcome now, at the original game $\mathcal{G}$ results to a utility of 3 for the boy which is an improvement for him.

The above exposition briefly describes in a way a rather general model of "Mechanism Design for playing games". In this mechanism design setting players' strategies-types are games and the allocations are game outcomes (in our case Nash equilibria, but we can easily use other solution concepts like e.g. correlated equilibria). Under this model one can pose some initially interesting questions like:

- For mechanisms without money that compute Nash equilibria, like the one we presented above, is there an interesting class of games $\mathcal{G}$ that induce truthfulness, i.e. the players don't have an incentive to misreport their utility matrices?

[2]For a quick and very basic introduction to Mechanism Design and Game Theory we refer to Deliverables D3.1 and D3.2.

- Using payments, can we enforce truthfulness for any class of games $\mathcal{G}$? Which are the mechanisms that minimize the payments requirements for a given class of games?
- Suppose that the mechanism is a distributed algorithm: the initial states of the processors/players are their values (utility matrix inputs), and the final state of every processor is one of its strategies. What algorithms make sense and what games are "playable" by them? A variant of this case can be that at every time step each processor has a strategy, and there are payoffs associated to every such time step.

This "mechanisms for game-playing" setting and the questions raised are interesting also in SMC perspective: the players report their true values to a trusted third party (the mechanism) which then selects an outcome for them and then the players have to comply with this. Can the notion of *non-repudiation* be brought to this setting to ensure compliance? Shall they report truthfully to this authority? How much do we have to pay them in order to ensure their "honest" participation? What are the most "efficient" protocols we can use while preserving truthfulness? How can we do this in a distributed way? The study of such problems would allow to leverage the core skills of UaESMC partners and explore the trade-offs of several dimensions of complexity of privacy-preserving applications [2].

Chapter 3

Tasks Worked on During Year 1

3.1 Optimization

Linear programming is one of the most basic optimization tasks. For many other tasks, there is a straightforward reduction to linear programming or to integer linear programming. Also, the task of finding *correlated equilibria* of a game reduces to linear programming. Hence, this task is of high interest to UaESMC.

There exists an implementation of the simplex algorithm that is composed from universally composable elementary arithmetic and logic operations working on the private domain [1]; the domain is based on secret sharing over finite fields. Due to the faithful implementation of a complex algorithm, the actual performance of the protocol is not too great.

The linear programming problem has a lot of algebraic structure. Hence there have been many proposals to base a secure linear programming protocol on the idea of *problem transformation*. Using this idea, the original problem is scrambled in the private domain, and the resulting linear programming problem is made public. The published problem is solved using conventional algorithms, and the result is again unscrambled in the private domain, using the randomness generated during scrambling. Hopefully,

- scrambling-unscrambling in private domain requires significantly less resources than the simplex algorithm,
- the public solving of the scrambled problem requires little resources, compared to private scrambling and unscrambling,
- the scrambled problem hides the original problem well.

A simple example of problem transformation is the inversion of values. Let $x \in \mathbb{Z}_q$ be a value stored in the private domain. Let the supported arithmetic operations of the domain be the computation of linear combinations of private values (with public coefficients), and the multiplication of two private values. Suppose we want to compute $x^{-1} \in \mathbb{Z}_q$ privately. We will then generate a private value $r \in \mathbb{Z}_q^*$, compute $xr \in \mathbb{Z}_q$ and make it public. In public domain, we compute $(xr)^{-1}$. Now we can compute $x^{-1} = (xr)^{-1} \cdot r$ in private [3]. Similarly, one can compute inverses of private matrices.

A common way to transform a linear programming problem has the multiplication of its constraint matrix by a random invertible matrix in the core of the method. A common shortcoming of all such proposals for linear programming so far is their lack of proper security analysis (recognized in some of the papers, ignored in others). Another common shortcoming of all such proposals is their complete failure to specify the probability distributions of the random real numbers they use for hiding the original problem.

3.1.1 Linear programming through problem transformation

In deliverable D2.2.1 we describe our approach to privacy-preserving linear programming through program transformation. In contrast to previous approaches, we have stated precise assumptions on the hardness of

certain computational tasks, serving as the basis of the security proof of our technique. These assumptions also allow us to characterize the loss of efficiency (the increase of the task) we must accept in order to get a certain level of security.

For the optimization task to make sense in the context of linear programming, the values defining the problem must be real numbers, not elements of finite fields. This significantly increases the novelty of our hardness assumption, which is also stated over real numbers (although in other aspects, it is similar to assumptions that have appeared in other papers). Hence the assumption should be subject of further study in the coming years.

We have implemented our problem transformation mechanism, though not yet on an SMC platform. Hence we cannot experimentally evaluate its performance yet. We could count the number of various operations (on private values) it uses; the operations comprise random real number generation (according to normal distribution), addition, multiplication, and square root computation. We can see that our proposed transformation is not much more expensive than existing approaches. We will not estimate the running times of our transformation on Sharemind — although the methodology and performance models are available [4], there are too many operations that have not been considered in these models and would significantly affect the performance. Instead, we will implement our transformation in early year 2 and obtain concrete running times.

3.1.2 Privacy-Preserving Genetic algorithms

The study of privacy-preserving methods for solving difficult optimization problems, and the adoption of existing heuristics for solving (NP-)hard problems to a privacy-preserving setting, is one of the main goals of UaESMC. In the context of Sharemind (this should apply also to other SMC frameworks based on secure evaluation of arithmetic circuits), genetic programming seems to be among the easiest to adapt methods due to its relatively uniform control flow. There have been previous attempts to make genetic algorithms privacy-preserving, but their security analysis has been lacking [10] or their application area has been very narrow [7].

In year 1, we have investigated the adaption of genetic algorithms to SMC frameworks, particularly to Sharemind. We have applied privacy-preserving genetic algorithms to the optimal subset cover problem (NP-hard) and to the shortest path problem in graphs. The general methods we have learned through these applications, may, however, be even more important than the results obtained for the particular problems. Our studies have brought new insights to the best methods for mutating and crossing over the candidate individuals, and for selecting a predetermined number k of individuals with top fitness.

Performance Evaluation

We implemented the genetic algorithm solving the subset problem described in Sec. 3.2 of deliverable D2.2.1 using the Sharemind system. We generated a 10×16 binary matrix A and an integer cost vector $\mathbf{c}$ inspired by the countermeasure selection scenario that motivated our study of the privacy-preserving subset cover problem. We created a data importer that was used to load A and $\mathbf{c}$ into a Sharemind installation. We then developed the described genetic algorithm in the SecreC programming language that is used to implement Sharemind applications. We implemented the oblivious top-k as a new protocol in Sharemind for optimization purposes. We then created a testing application that let Sharemind compute the output $\mathbf{x}$ on all possible inputs $\mathbf{t}$ using the genetic algorithm. We also computed all the optimal solutions using Sage and the GNU Linear Programming Kit and used the results as reference values to evaluate the correctness of the private implementation.

The Sharemind system was deployed on three computers running on a local network. The computers contain 12 CPU cores and 48 gigabytes of RAM. However, during experiments we saw that at most two cores per machine were being fully used and the memory usage of Sharemind did not grow over 150 megabytes.

We ran the tests for generation sizes $k = 8, 12, 16, 23, 32$ and number of generations $g = 5, 10, 20$. For each of the pairs of these values, we determined the percentage of correctly computed optimal costs out of

	$g = 5$	$g = 10$	$g = 20$
$k = 8$	3.71% (3711 ms)	45.21% (7187 ms)	78.22% (14167 ms)
$k = 12$	18.75% (4220 ms)	79.39% (8186 ms)	92.87% (16120 ms)
$k = 16$	55.66% (4733 ms)	95.61% (9247 ms)	99.51% (18291 ms)
$k = 23$	89.55% (5420 ms)	99.80% (10546 ms)	99.90% (21008 ms)
$k = 32$	99.32% (6440 ms)	100.00% (12702 ms)	100.00% (25164 ms)

Table 3.1: Accuracy and running time of the privacy-preserving genetic algorithm for g generations of size k.

$2^{10} = 1024$ possible input vectors. We also measured the average execution time. The results are displayed in Table 3.1.

We see that in already under 6.5 seconds it is possible to achieve near-perfect performance of the algorithm, and that increasing the size of the generation helps to obtain better precision with much lower cost in time compared to increasing the number of generations.

Our implementation was optimized by using vector operations. Our profiling showed that in one six-second run, the secure computation platform performed tens of thousands of additions, multiplications and comparisons, showing the high performance of the underlying platform.

Similarly to the secure subset cover problem, we used genetic algorithm to solve the single pair shortest path problem for graphs detailed in Sec. 3.3 of deliverable D2.2.1. The underlaying basic genetic algorithm is the same, but the data structures are larger (adjacency matrices of directed graphs take $\mathcal{O}(n^2)$ space) and the fitness function is different. We used vectorization to apply most functions to the whole generation in one go rather than applying them to each individual separately. As part of the fitness function we must check if the individual actually contains a valid path. Unfortunately, using the equation shown in deliverable D2.2.1 for this gives suboptimal results as it often rejects individuals that only slightly deviate from the optimal path and only accepts individuals with a very specific pattern. As a result, the genetic algorithm rarely finds a solution that is sufficiently close to the optimal solution. This spoils the idea of incrementally improving the results and is therefore not suitable to be used in genetic algorithms.

3.2 Statistical Analysis

In Chapter 2 we mention privacy-preserving statistical analysis of structured data as one of the problems that UaESMC will provide tools for. We have started the consideration of this task by specifying the operations we want to see implemented and relationships between them. Altogether, these operations comprise an abstract data type, allowing the manipulation of the analyzed data by applying different filters and projections on it, as well as running different statistical tests on the current data. See Deliverable 4.1 [9] for the complete specification.

In year 1, we have started designing and implementing the functionality for statistical analysis.

3.2.1 Oblivious Sorting

Sorting a table may be a goal in itself. It is also a useful primitive when selecting certain extremal rows of the table, when hiding the order of the rows, or in preparation of merging the table with a different one.

An oblivious sorting procedure has its control flow independent of the sorted data. In our approach, such a procedure is based on sorting networks. Our sorting protocol the main component of the secure subset cover solution and profiling showed that the majority of the running time of the genetic algorithm was spent by the sorting algorithm. Therefore, the performance timings of the secure subset cover algorithm illustrate the running time of the sorting algorithm with the described sorting parameters.

3.2.2 Implemented Statistics and Tests

Not all statistical functions can be precisely computed on integers. To achieve more precise results, our work-in-progress implementation of the statistical functions and algorithms uses the secure floating point features in Sharemind 3. However, the Sharemind 3 system is currently being developed under a confidentiality contract that requires that materials regarding Sharemind are published only after they have been cleared for publication. This process could not be completed by the time of delivering this document.

3.2.3 Data Classification

The division of data into several different classes is another important means of understanding it, and tools to do it in a privacy-preserving manner will be a part of our statistical analysis suite. To integrate with other parts of the suite, we have implemented the Kernel Perceptron algorithm on top of the Sharemind platform, optimized it for that platform, and measured its performance.

We have tested the implemented Kernel Perceptron algorithm on the Iris dataset. Some modifications have been done to it before testing:

- Although the original dataset has three distinct classes, the first and the third classes have been merged together so that the classifier could be tested on separating just two classes.
- All the values have been scaled to integers. This is withoutlessening of generality — we can just imagine that all the measures are scaled up, such that their ratios remain the same.
- For simplicity, the classes are marked just "1" and "-1". If we want to introduce a third class, we may label them for example with two bits: "1 1", "1 -1", and "-1 -1".

The set has been tested for different fractions of the iterations computed in parallel. The graphs in Fig. 3.1 show how the time and true positive/true negative (TP/TN) rate grow with the number of "global iterations" (how many times the entire for-cycle is repeated). Each test has been performed 10 times, and the average has been taken.

The horizontal axis on the graphs repesents the number of iterations, and the vertical axis is the percentage of correct answers.

The initial version of the algorithm is supposed to work until convergence. This implementation uses an upper bound for the number of iterations that is given as a parameter. The main problem is actually that checking if the vector has changed would in turn require additional communication, although not too much since the additional comparison needs to be done only once per iteration. Working until convergence would also be much slower (depending on the given threshold - how much of the training set is classified correctly). Overlearning is also not too good.

The graph in Fig. 3.2 shows how the time grows with the number of iterations. Four different lines show the time (in seconds) for different fractions of the iterations computed in parallel.

3.3 Strong Truthfulness

A fundamental assumption throughout Game Theory and Mechanism Design is that all participating players (agents) are fully rational and act selfishly: they have their own well-defined utility function and they only care about optimizing this function. In particular, if truth-telling (about one's preferences) is the dominant strategy equilibrium in some game, then the players are assumed to do exactly that. This assumption might indeed be true if one looks at the "big picture", but when considering a subprotocol of some large system, the other parts of that system may introduce externalities with respect to this subprotocol, invalidating that assumption.

A mechanism is *truthful* if telling one's true preferences is a dominant strategy for players. However, this does not exclude a player getting exactly the same benefit in some situations by lying to some extent. If there are externalities in the form of other, related games the same set of agents is playing, then this

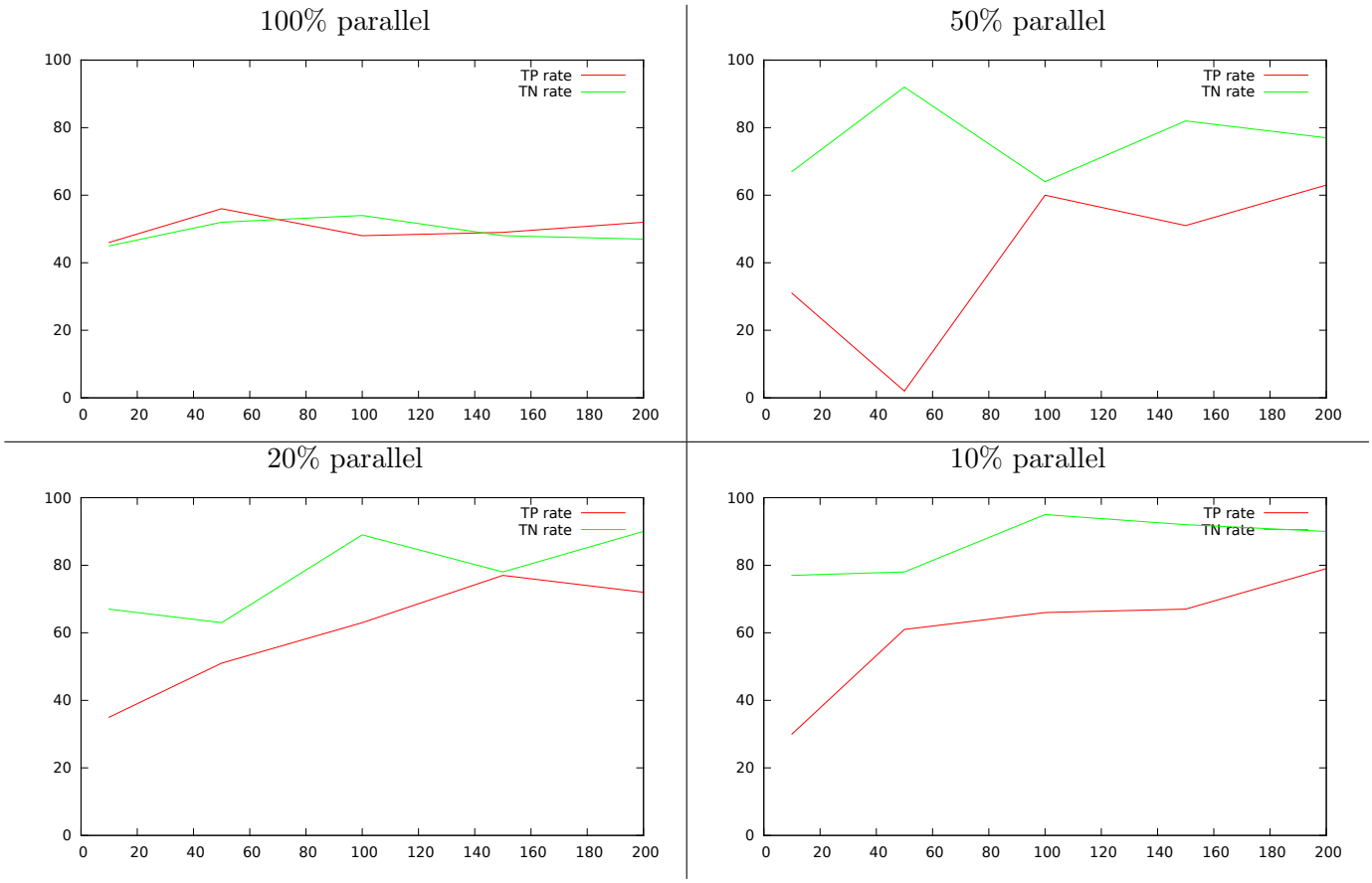

Figure 3.1: Correct answer rate of the kernel perceptron algorithm

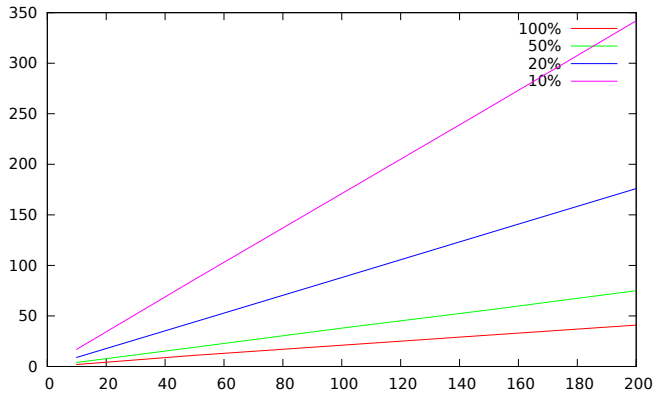

Figure 3.2: Running times of the privacy-preserving kernel perceptron algorithm

lying may bring it some benefit in some other place. A mechanism is *strongly truthful*, if any deviation from truth-telling is guaranteed to bring harm to the player. Bigger deviations bring more harm; the notion of strong truthfulness is quantified by the conversion factor. In deliverable D3.2 we show that strongly truthful mechanisms (with sufficiently large conversion factors) for sub-problems are sufficient to obtain strongly truthful mechanisms for the whole problem. We also present a general technique of constructing strongly truthful mechanisms from truthful mechanisms.

Chapter 4

Work Plans for Year 2

In the second year of UaESMC we will continue to improve our solutions for the identified problems, described in Chapter 2. We will increase the efficiency and robustness of the existing solutions, as well as extend them to cover aspects we have not considered yet. At the same time, we will also start consolidating our techniques into a framework that will be delivered at the end of the project.

4.1 Optimization

We will continue the work on both linear programming and on the privacy-preserving algorithms for NP-hard optimization problems. For linear programming, we will study further the computational assumptions we've made, trying to identify any shortcomings they may have, and possible fixes for them. For NP-hard problems, we will study how to implement other heuristic optimization methods (e.g. ant colony optimization, simulated annealing, etc.) on Sharemind and other typical platforms for SMC.

Our secure linear programming approach has currently not been implemented on Sharemind, nor have the details of implementing its private computations on a SMC framework been fully worked out. This work also has to be done during year 2.

In year 2, we will also concentrate more on the shortest path problem in graphs, especially in the context of network management. This will allow us to look at SMC techniques in the setting of limited connectivity.

4.1.1 All-Pairs Shortest Paths

In the all-pairs shortest path (APSP) problem, the goal is to find the distances and/or the shortest paths between all vertex pairs in a graph $G = (V, E, w)$. Here V is the set of vertices, $E \subseteq V \times V$ is the set of (directed) edges and $w : E \to \mathbb{R}$ gives the length of each edge. We are particularly interested in the version of the problem stemming from the network management area. In this problem, the set of vertices is partitioned into $V = V_1 \dot\cup \cdots \dot\cup V_k$. The set V_i corresponds to the border gateways in the network managed by the i-th ISP. The edge lengths between nodes in V_i are known by the i-th ISP (and considered a business secret by it). The edge lengths between a node $u \in V_i$ and a node $v \in V_j$ are known at least by the i-th and j-th ISPs. We assume that two gateways in the same network are best connected through this network. Hence, if two nodes belong to the same part V_i, then the shortest path between them will only pass through nodes in V_i. We can assume that before starting multiparty computation, each ISP has already found the distances between the gateways in its network. Hence the graph G is a complete graph when restricted to a particular V_i. Also, if $u, v \in V_i$, then the distance from u to v equals the length of the edge from u to v.

In this setting, we will study the use of adopting generic APSP algorithms to SMC frameworks (taking into account that distances between two nodes in the same part have already been computed). We will also consider the adaptation of privacy-preserving graph algorithms by Brickell and Shmatikov [6] from two-party to multi-party setting.

4.2 Statistics API

In deliverable D4.1 we have described a numerous set of statistics and data analysis primitives that a SMC engine should support in order to be usable for the analysis of statistical data. Certain primitives have already been designed and implemented, as described in Sec. 3.2. In year 2, we will complete the implementation of the API. By the end of year 2, Task 1.3 will produce an example prototype application. It is quite likely that the developed statistics API will find significant use in it.

4.3 Mechanism Design for Statistical Data Collection

The DoW identified the lack of motivation by the data owners to submit truthful data as one issue hindering the use of SMC in statistical data collection. To overcome this, we have to quantify the data owners' motivation to lie, their desire to learn the correct result of the analysis, the effect of lying to their reputation, etc. These quantities could serve as inputs to mechanism design, designing a game and payment modalities to obtain truthful results of analyses. We cannot hope to give precise values to these quantities in UaESMC, but we can study how the relationships between these quantities affect, whether parties are motivated to tell the truth.

Similarly, we can study the motivation of the participants of a secure multiparty computation (at the same time, they are also data owners) to collude with some other party and recover the data of third parties. Here we must characterize parties' desire to learn other parties' data, their fear of leaking their own data, and the computation mechanism's characteristic to prevent the singling out of data belonging to only some of the parties. We believe that shuffling the data input by the parties at the beginning of the protocol will make it much harder to detect which record was provided by which party.

4.4 Mechanisms for Playing Games

In Sec. 2.4 we outlined a general problem making the players of a game to make the move that the (trusted) mediator has selected for them. In year 2, we intend to investigate SMC mechanisms for implementing such mediators, and for using non-repudiation usually offered by digital signatures to bind the players to their allocated moves in an accountable manner. This task will integrate some of the solutions we've developed in year 1, as well as pose some new interesting problems:

- Finding a set of moves that maximizes some sort of social welfare is an optimization task. It is well-known that correlated equilibria may be found through linear programming. As far as we know, noone has yet tried this in practice in a privacy-preserving manner, because of high computational complexity of the task. With parties being bound to make the moves assigned to them, other choices of moves may be possible, the finding of which translates into an optimization task of some different kind. We will investigate the ways to make the choices of moves and privacy-preserving methods to compute them.

- Non-repudiation requires that the certificates and secret signing keys of parties are used in the computation in some manner. It is likely that the necessary public-key operations can be performed under SMC in a more efficient manner than directly implementing the arithmetic circuits for these operations in some SMC framework. In year 2, we will come up with privacy-preserving methods for these operations and for their integration with other SMC methods and frameworks.

- Public-key infrastructure has been deployed in many countries and organizations, using more or less standard protocols, software and hardware. A particular example are the electronic identity cards and mechanisms deployed by several European countries. In these mechanisms, the secret key of a person is stored on a smartcard that is capable of executing certain operations with it (including the issuance of signatures conforming to certain standards). The set of operations a card executes is

limited for security purposes. In UaESMC, we will study whether the available electronic identity and non-repudiation infrastructures can be directly used in our mechanisms, or do we need to access the secret signing keys in a manner that is not made available by tokens storing them.

4.5 Progress towards the UaESMC Framework

The UaESMC framework will be a set of recipes for choosing the techniques and implementations of SMC for a particular secure computation task. It will be organized along the dimensions identified in the Description of Work [2, Sec. B1.1.3]. During the first year of the project, as well as in activities that the partners have run in parallel with this project, we have already studied the effects of variation along certain dimensions characterizing the computation (*Data Types*, *Complexity of Computation*, *Benefit of Misbehaving*). More work remains to be done for other dimensions; particularly interesting results can be expected from studying the cost of information leakage / benefit of obtaining the result, and the availability of additional infrastructure.

Bibliography

[1] SecureSCM. Technical report D3.2: Protocol Description V2. `http://www.securescm.org`, Jan 2010.

[2] UaESMC Grant Agreement Annex I — Description of Work, 2011.

[3] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *PODC*, pages 201–209, 1989.

[4] Dan Bogdanov. *Sharemind: programmable secure computations with practical applications.* PhD thesis, University of Tartu, February 2013.

[5] Vincent J. Bono. 7007 explanation and apology. Message to NANOG mailing list, April 26th 1997. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html`.

[6] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 236–252. Springer, 2005.

[7] Daniel Funke and Florian Kerschbaum. Privacy-preserving multi-objective evolutionary algorithms. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *PPSN (2)*, volume 6239 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2010.

[8] Jennifer Golbeck and James A. Hendler. Reputation network analysis for email filtering. In *CEAS*, 2004.

[9] Roberto Guanciale. Identification of application scenarios, January 2013. UaESMC Deliverable 4.1.

[10] Jun Sakuma and Shigenobu Kobayashi. A genetic algorithm for privacy preserving combinatorial optimization. In Hod Lipson, editor, *GECCO*, pages 1372–1379. ACM, 2007.

[11] Kadri Tõldsepp, Pille Pruulmann-Vengerfeldt, and Peeter Laud. Requirements specification based on the interviews, July 2012. UaESMC Deliverable 1.2.