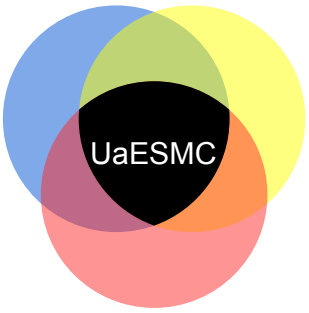

Project N°: **FP7-284731**

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 D4.1
Identification of application scenarios

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: **KTH**

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:
Identification of application scenarios

This document summarizes deliverable D4.1 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 report identifies a suitable set of application scenarios for demonstrating the usefulness of the approaches developed in the UaESMC project. The report describes the scenarios by three main point of views: algorithmic, application and deployment.

The general algorithmic problems identified in this report are: (*a*) linear programming applications; (*b*) eigenvalues; (*c*) point projection; (*d*) principal component analysis (PCA), also used to compute eigenvalues; (*e*) equation systems, both linear and non-linear; (*f*) path and flow problems in graphs; and (*g*) a set of common statistical analysis algorithms. The statistical analysis problem is given as a set of algorithms that a minimal statistical API should cover, including table sorting, filtering, dataset aggregation functions, cluster and regression analysis (both linear and non-linear).

The report also lists various real-life scenarios that can be addressed by using the previously described algorithms. Among others, it covers several mobile computing applications where a user could benefit from possibly sensitive information from other users, for example, picture based friend matching, P2P positioning and navigating. The same picture based matching is also considered in a scenario where an investigation agency is using public/CCTV cameras to search for a criminal or missing person. In this case the underlying algorithm is similar but the privacy requirements differ. Continuing with intelligence agencies, a real-life use case from Deliverable D1.2 [9] is also listed. This is an optimization problem about finding the safest path while the information about the terrain and its risks (e.g. location of landmines) is a secret and therefore not freely available. More use-cases that emerged from the interviews summarized in Deliverable D1.2 cover different types of privacy-preserving auctions, repurchasing bonds and optimizing energy consumption on a smart grid. Network management is considered as the last larger group of real-life use-cases. Detailed network traffic data is sensitive information and thus cannot be shared among all ISP-s. However, this report mentions (QoS aware) routing and intrusion detection as some of the possible areas where privacy-preserving algorithms could give closer to optimal results.

Finally, the report describes trusted security modules and hardware tokens as possible environments to deploy SMC solutions.

List of Authors

Roberto Guanciale (KTH)

Contents

Chapter 1

Introduction

This document summarizes deliverable D4.1 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 report identifies a suitable set of application scenarios for demonstrating the usefulness of the approaches developed in the UaESMC project. The application scenarios have been extracted from the interviews presented in the Deliverable 1.2 [9]. Chapter 2 identifies a set of useful "primitive" computations that should be addressed using SMC. Chapter 3 presents real applications that can take benefit from SMC. The chapter exemplifies real problems and stresses the application constraints in terms of (i) privacy (ii), network structure and (iii) computational resources. Finally, Chapter 4 presents deployment scenarios to identify the benefits of adopting specific settings in order to reduce the computational and network costs of the SMC solutions.

Chapter 2

Algorithmic problems

This Chapter presents an overview of algorithmic problems. We identify a set of useful "primitive" computations that abstract from the constraints and goals of real applications. The main goal of this chapter is to formally identify the computations that should be addressed using SMC. Hence, we do not describe the possible SMC solutions to these problems. As a general rule, each problem is first formalized and then possible privacy constraints are suggested. In some cases a reference to a widely adopted non-secure algorithm is presented.

2.1 Linear programming

Linear programming applications require determining the best solution that satisfies a list of requirements, where both the objective function and the constraints are expressed as a linear relationships. Let x be a vector representing the variables; the problem is modeled as finding the $\bar{x}$ assignment that minimizes the $c^T x$ objective function and satisfies the constraint $Ax \leq b$. Linear programming problems can be classified as

- problems whose variables are continuous
- problems whose variables are discrete integers

Linear programming problems can be subject to different privacy constraints:

1. The number of variables is public, each participant knows a subset of the constraints (rows of A and of b), each participant knows a subset of the objective function (rows of c).
2. The number of variables is public, each participant knows a subset of the constraints for a subset of the variables, each participant knows a subset of the objective function (rows of c).
3. The participants can discover the whole target assignment $\bar{x}$ or only a subset of the variables.
4. The number of variables can be public or secret.
5. Some scenarios can admit data leakage.

2.2 Eigenimages

Eigenimages [4] (also called Eigenfaces) is a technique adopted in several pattern recognition problems, ranging from face recognition to gestures interpretation and medical imaging analysis. All these problems represent the information as bi-dimensional images.

Informally, the eigenimages technique transforms the bi-dimensional information into characteristic feature vectors of a low-dimensional vector space (called "trained-space"), whose basis is composed of eigenimages. The eigenimages are determined through Principal Component Analysis (PCA) (see Section 2.4) from a set of training images:

- every original image is projected onto the trained-space to obtain a marker image
- recognition of an image is done by projecting the image to the trained-space and locating the nearest marker image

The eigenimages technique proceeds as follows. Let $\Theta_1, ..., \Theta_M$ be a set of M training images. All training-images have the same resolution c, r and can be represented as M vectors of length $N = c * r$. Initially, the training average image Ψ is computed and subtracted by all training images as follow:

$$\Psi = \frac{1}{M} \sum \Theta_i$$

$$\Phi_i = \Theta_i - \Psi$$

Then, PCA is applied to the matrix

$$C = \frac{1}{M} \sum \Phi_i \Phi_i^T$$

obtaining at most M non-zero eigenvalues. The $K < M$ eigenvectors $u_1, ..., u_K$ corresponding to the K greatest eigenvalues are the basis of the computed "trained-space". Then, the original images $\Theta_1, .., \Theta_M$ (or other images that have to be recognized) are projected (see Section 2.3) to the "trained-space" obtaining $\Omega_1, ..., \Omega_M$ marking images.

To recognize an image Γ, the image is projected to the "trained-space" (see Section 2.3) to obtain $\bar{\Omega}$. A match is reported queering the distances between the projection and the marker images:

- $min(\mid \bar{\Omega} - \Omega_i \mid)$ must be smaller than a given threshold
- the recognized image j is $argmin(\mid \bar{\Omega} - \Omega_i \mid)$

Common pattern recognition problems can be handled with the eigenimages technique. A common scenario involving two parties consists of a participant that knows the "trained space" and the other one that knows the image to be matched. Some scenarios can require that only one of the two parties obtains the result.

2.3 Point Projection

A common geometrical problem is projecting a 3-dimensional point Γ (represented as a vector) on a 2-dimension coordinate system, identified by two linearly independent 3-dimensional vectors u_1, u_2 and the origin Ψ. The projection 2-dimensional point Ω is computed by the following equation

$$\Omega = (u_1^T(\Gamma - \Psi), u_2^T(\Gamma - \Psi))$$

In general, let Γ be a N-dimensional vector, let U be a K-dimensional subspace ($K < N$) identified by the basis $u_1, ..., u_K$ (u_i is a N-dimensional vector) and the origin Ψ. The projection Ω is computed by

$$\Omega = (\omega_1, .., \omega_K) \quad \omega_i = u_i^T(\Gamma - \Psi)$$

Possible privacy constrains

- participant A knows the point Γ, participant B knows the subspace parameters, A (or B or both) obtains the projection Ψ
- each participant A_i knows Γ_i, participant B knows the subspace parameters, all A_i obtain the projection Ψ
- participant A knows Γ, each participant B_i knows the u_i, B_i obtain the projection Ψ_i

2.4 Principal Component Analysis

Principal Component Analysis (PCA) is a technique adopted to analyze of statistical datasets. The main goal of PCA is feature reduction: reducing the number of variables that characterize the experiments onto a smaller set of uncorrelated variables. PCA transforms the data to a new coordinate system such that the first coordinate (called first principal component) holds the "greatest variance" of the data, the second coordinate holds the second greatest variance, and so on.

To ensure that the principal component identifies the maximum variance it is necessary to subtract the mean from the data-set (notice that Y has zero-mean):

$$m = \frac{\sum X}{\mid X \mid}$$

$$Y = [X_1 - m, ..., X_n - m]$$

The principal component w_1 of a data-set X can be defined as

$$w_1 = argmax_{|w|=1}(Var(w^T Y)) = argmax_{|w|=1} = E\{(w^T Y)^2\}$$

The kth principal component is computed by subtracting the first $k-1$ components from Y:

$$\bar{Y}_k = \bar{Y}_{k-1} - w_i w_i^T Y$$

$$w_k = argmax_{|w|=1} = E\{(w^T \bar{Y}_k)^2\}$$

Trusted computation The simplest method to compute principal components exploits the covariance matrix YY^T. The techniques proceeds as follows:

- compute the reverse sort of eigenvalues $v_1, ..., v_n$ of the covariance matrix YY^T
- for each eigenvalue v_i compute the corresponding eigenvector w_i
- output the matrix $V = [w_1, ..., w_n]$

2.5 Equation systems

2.5.1 Linear systems with unique solution

The problem of finding the unique assignment of n variables satisfying n linear constraints can be expressed by the matrix equation $Ax = b$.

Trusted computation Linear systems can be solved by adopting Gauss-Jordan elimination or QR-decomposition. If the size of the problem does not allow a direct approach, the iterative approaches presented in 2.5.3 can be exploited.

Privacy constraints

- The participant P_1 knows the matrix A, the participant P_2 knows b, the participant P_2 obtains x
- Each participant P_i knows a row (column) of A and b_i, each participant P_i obtains x_i
- The matrix A and the vector b are linear combination of information known by participants (e.g. $A = \sum A_i \wedge b = \sum b_i$)

2.5.2 Linear least squares

Let the system $Ax = b$ be overdetermined, namely having more equations than variables. This kind of systems usually has no solution, but is used in several scenarios where the parameters are affected by errors. The linear least squares is the problem of solving such systems by finding the assignment $\bar{x}$ which fits the equations "best", in the sense of solving the quadratic minimization problem:

$$\bar{x} = argmin_x(\mid b - Ax \mid^2)$$

Trusted computation Several approaches exist to solve the problem. The linear least squares has an unique solution if the columns of A are linearly independent. The problem can be solved by handling the "normal equations":

$$AA^T\bar{x} = A^Tb$$

in some cases, it is reasonable to directly invert the normal equation matrix, then the solution is found by using QR decomposition or solving the non-overdetermined system

$$\bar{x} = (A^TA)^{-1}A^Tb$$

2.5.3 Non-linear least squares

Non-linear least squares is used to fit a set of m observations with a model that is non-linear in n unknown parameters ($m > n$). Let β be a set of experiments $(x_1, y_1), ..., (x_m, y_n)$, and a function $y = f(x, \beta)$, where β is a vector of n elements. The non-linear least squares is the problem to find β such that it minimizes

$$S = \sum_i^m r_i^2(\beta)$$

where the residuals $r_i(\beta)$ are

$$r_i(\beta) = y_i - f(x_i, \beta)$$

Informally, non-linear least squares finds the vector β of parameters such that the function fits best the given experiments in the least squares sense.

Trusted computation One of the most adopted techniques is the Gauss-Newton method. It is an iterative method that starts from an initial guess β_0. Let J_f the Jacobian matrix of the function f, at each iteration the increment Δ is computed by solving

$$(J_f^TJ_f)\Delta_k = J_f^Tr$$

and then the iterative solution is computed as $\beta_k = \beta_{k-1} + \Delta_k$

2.6 Graph problems

2.6.1 Path problems

Let $G = (V, E)$ be a graph consisting of V vertices and $E \subseteq V \times V$ edges. Let W be a function ($W : E \to R$) that represents the weight of the edges.

A path of vertices $p = v_1, \ldots, v_n$ is a sequence of vertices such that v_i is adjacent to v_{i+1}. We use $s(p)$ and $t(s)$ to represent the "source" (first) and "target" (last) node of a path respectively. The cost of a path is computed according to the sum of edge weights. Let (v_i, v_j) be the edge connecting the vertex v_i to the vertex v_j, the "cost" of a path of n nodes is defined as the sum $w(p) = \sum_{i=1}^{n-1} W(p[i], p[i+1])$.

Single pair shortest path is the problem of finding a path p that connects a source node s to a target node t and minimizes the cost:

$$SPSP(s,t) = argmin_p\{w(p) \mid s(p) = s \wedge t(p) = t\}$$

This problem can be expressed as a linear problem (see Section 2.1). This requires to minimize $\sum_{v_i,v_j} W(v_i, v_j) * x_{i,j}$ so that

$$\sum_j x_{ij} - \sum_i x_{ji} = \begin{cases} 1, i = s \\ -1, j = t \\ 0, otherwise \end{cases}$$

and $x_{ij} = 0$ if $(v_i, v_j) \notin E$.

Single source shortest path is the problem of finding the set of paths P that connect the source s to all vertices in V and minimize the costs:

$$SSSP(s) = \{SPSP(s,t) \mid t \in V\}$$

All-pairs shortest path is the problem of finding the shortest path for app pairs of vertices:

$$APSP = \{SPSP(s,t) \mid s,t \in V\}$$

2.6.2 Flow network

A flow network is a directed graph (V, E, c, s, t) where $E \subseteq V \times V$, $s, t \in V$ and $c : E \to R$. For each edge $e = (v_1, v_2) \in E$, the function $c(e) \geq 0$ represents the corresponding capacity. The two special vertices s and t represent the source and the target of the flow respectively. A flow is a function $f : E \to R$ that defines the amount of flow traversing each edge of the network. A flow is admissible if

- $f(e) \leq c(e)$: the capacity constraints are satisfied by the amount of flow traversing the edges.
- $\sum_{\forall v_1,v_2,v_2 \notin \{s,r\}, e=(v_1,v_2)} f(e) = \sum_{\forall v_1,v_2,v_2 \notin \{s,r\}, e=(v_2,v_1)} f(e)$: the total amount of net-flow for non source/target nodes is zero.

According to these constraints some standard problems are:

Maximum flow requires to discover an admissible flow that maximizes $\sum_{\forall v, e=(s,v)} f(e)$ (the total flow produced by the source).

Minimum cost is a cost minimization problem. A fixed amount of flow m have to be delivered from the source s, represented by the constraint $\sum_{\forall v, e=(s,v)} f(e)$. Moreover, each edge has a cost depending linearly from the amount of flow carried ($f(e) * k(e)$). The goal is to minimize the total flow cost defined as $\sum_{\forall e} f(e) * k(e)$.

Circulation problem extends the above problems with an additional constraint: the amount of flow crossing an edge is lower bound ($f(e) \geq l(e)$)

All the presented problems can be naively (but not always efficiently) solved using linear programming 2.1.

2.7 Statistical Analysis

We introduce a set of statistical problems in order to sketch the tools required to analyze, inspect and integrate datasets. Our goal is to formalize the "minimal" functionality of a statistical API.

2.7.1 Data representation

We use o, o_1, o' to range over observations. Each observation is a finite sequence of scalar values (integer, float, string). A physical table is a finite sequence of observations $[o_1, \ldots, o_n]$. We assume that all observations in a table contain the same number of elements. The size of observations of a table is also called number of axes of the table.

We also use the following notations:

- $\mid x \mid$ to represent the length of a sequence x
- $x[i]$ to represents the i-th element of the sequence x
- $min(x), max(x)$ to represent the minimum and maximum of a sequence containing scalar values
- $[f(x) \mid x \in y \wedge p(x)]$ to represent list comprehension
- $x + y$ represents the concatenation of two sequences
- $range(x)$ represents the sequence $[1 \cdots \mid x \mid]$

Logical tables (ranger over by L, R, S, T) are inductively defined; a logical table is either a physical table or a triple (S, E, m) consisting of a source "logical table" S, an extension "logical table" E and a mask vector m. Intuitively, a logical table is used to incrementally filter the observations and to extend the number of axes. Logical tables are "well defined" if the involved sequences (S, E and m) have the same length and the masks of the two logical tables S and E are equal. Let $L = (S, E, m)$ be a logical table, we use the following notations:

- $s(L) = S$, $e(L) = E$ and $m(L) = m$ represents the source, the extended axes and the mask vector of the logical table respectively
- $\mid L \mid = \sum w(L)$ is the number of observations that satisfy the mask vector
- $L[i] = S[i] + E[i]$ represents the i-th observation concatenated with the extended axes
- $range(L)$ represents the sequence $[1 \cdots \mid m(L) \mid]$

If L is a physical table, then $s(L) = L$, $E(L)$ is empty and $m(L)$ is the identity vector.

2.7.1.1 Predicates and expressions

In the following sections we assume that is possible to express predicates and functions over the observation domain (lists of scalar values). We assume that the language supports: boolean algebra operators, basic arithmetic operators and basic comparison operators.

2.7.2 Privacy constrains

The analysis can involve data combined from several parties. In this case, each table belongs to a specific party and its content can not be disclosed to the other participants. Moreover, the analysis can be requested by a participant that does not hold any table. In some cases, it is important to keep secret the executed analysis and its results.

2.7.3 Derived tables

We consider the following standard operations on logical tables.

Selection The selection yields a logical table containing the observations such that the predicate p holds:

$$\sigma(R, p) = (R, \emptyset, [m(R)[i] \wedge p(R[i]) \mid i \in range(R)]$$

Join Let R and S be two tables and p a predicate, the join function yields a new "physical" table containing all combinations (concatenation) of observations in R and S that satisfy the predicate:

$$J(R, S, p) = [R[i] + S[j] \mid i \in range(R) \wedge j \in range(S) \wedge m(S)[j] \wedge m(R)[i] \wedge p(R[i] + S[j])]$$

Sort Let R be a table, the sort function yields a new "physical" table containing the sorted observations:

$$S(R) = [R[i] \mid i \in range(R) \wedge m(R)[i]]$$

and $\forall i, j \in range(S(R))$ if $i < j$ then $S(R)[i] \leq S(R)[j]$.

Table Aggregation Let R be a table and A a list of indexes, the table aggregation function produces a new "physical" table representing the ordered sequence of all distinct value of the corresponding axes:

$$g(R, A) = [x \mid x \in \{[R[i][j] \mid j \in A] \mid i \in range(R) \wedge m(R)[i]\}]$$

To compute metrics of aggregated observations, the functions presented in Section 2.7.4 can be used to generate additional axes. For each possible distinct values of the aggregation axes, the aggregation function is applied to a logical table that filters the aggregated observations:

$$\begin{aligned} &g(R, A, f_1, \ldots, f_n) = [G[i] + [f_1(G_i), \ldots, f_n(G_i)] \mid i \in range(G)] \\ &where\ G = g(R, A) \\ &and\ G_i = \sigma(R, \lambda r. \textstyle\bigwedge_{j \in A} r[j] = G[i][j]) \end{aligned}$$

Derived attributes The following function extends the axes of a logical table by evaluating the parameter functions:

$$e(R, f_1, \ldots f_n) = (R, ([[f_1(R[i]) \ldots f_n(R[i])] \mid i \in range(R)]), m(R))$$

The parameter functions can exploit basic boolean and arithmetic operators and can be used to extend a table with derived attributes, for example:

- computing for each observation the sum of several axes
- computing for each observation the maximum of several axes

2.7.4 Common aggregations

The following functions are common statistical aggregations of the logical table R respect to the axis a. All the following functions yield a scalar value and can be used in the aggregation function (see Section 2.7.3). We also assume that standard arithmetical aggregation functions are available (e.g. $min, max, count, sum, prod$),

Average The most common averages functions used in statistics are

- arithmetic mean: $\mathcal{M}_a(R,a) = \frac{1}{|R|}\sum_{i\in range(R)\wedge m(R)[i]} R[i][a]$
- geometric mean: $\mathcal{M}_g(R,a) = \sqrt[|R|]{\prod_{i\in range(R)\wedge m(R)[i]} R[i][a]}$
- harmonic mean: $\mathcal{M}_h(R,a) = \frac{|R|}{\sum_{i\in range(R)\wedge m(R)[i]} \frac{1}{R[i][a]}}$

Standard deviation The standard deviation is defined as

$$\mathcal{SD}(R,a) = \sqrt{\frac{1}{\mid R \mid} \sum_{i\in range(R)\wedge m(R)[i]} (R[i][a] - \mathcal{M}_a(R,a))^2}$$

Variance The variance is defined as

$$\mathcal{VAR}(R,a) = \frac{1}{\mid R \mid} \sum_{i\in range(R)\wedge m(R)[i]} (R[i][a] - \mathcal{M}_a(R,a))^2$$

Quantile The quantile for the probability p is defined as

$$\mathcal{Q}(R,a,p) = inf\left\{ R[i][a] \mid i \in range(R) \wedge m(R)[i] \wedge p \leq \frac{\mid \sigma(R, \lambda r'.r'[a] < r[a]) \mid}{\mid R \mid} \right\}$$

Some common computed quantile are the quartiles:

- lower quartile: $lq(R,a) = \mathcal{Q}(R,a,\frac{1}{4})$
- median: $me(R,a) = \mathcal{Q}(R,a,\frac{1}{2})$
- upper quartile: $uq(R,a) = \mathcal{Q}(R,a,\frac{3}{4})$

One-sample t-test One-sample T-test for the mean μ based on the axis a of the table R is defined as

$$\mathcal{T}_{t_1}(R,a,\mu) = \frac{\mathcal{M}_a(R,a) - \mu}{\mathcal{SD}(R,a)/\sqrt{\mid R \mid}}$$

We use $\mathcal{P}_t(v,n)$ to represent the two-tailed p-value for the statistic v and n degree of freedom, computed using the $t-$distribution.

2.7.5 Two table aggregation functions

The following functions are common statistical aggregations of two logical tables and yield a scalar value. These functions can not be used as parameter of the table aggregation primitive (see Section 2.7.3).

Independent two-sample t-test The T-test for two samples based on the axis a and b of the tables R and S, assuming the same variance is defined as:

$$\mathcal{T}_{t_{2i}}(R,a,S,b) = \frac{\mathcal{M}_a(R,a) - \mathcal{M}_a(S,b)}{\sqrt{\frac{(|R|-1)*\mathcal{SD}(R,a)^2+(|S|-1)*\mathcal{SD}(S,b)^2}{|R|+|S|-2}} * \sqrt{\frac{1}{|R|}+\frac{1}{|S|}}}$$

X^2 test of a distribution Let R be a table, A a sequence of axis indexes and E be a sequence containing the expected distribution (number of occurrences) of each distinct value of the axes ($g(R, A)$). We require that $\sum E =\mid R \mid$ and $\mid E \mid=\mid g(R, A) \mid$. The X^2 test for the distribution is defined as:

$$\mathcal{T}_{X^2}(R, A, E) = \sum_{i \in range(E)} \frac{(g(R, A, ||) - E[i])^2}{E[i]}$$

We use $\mathcal{P}_{X^2}(v, n)$ to represent the p-value for the statistic v and n degree of freedom, computed using the X^2-distribution.

X^2 test of independence Let R be a table and A_1 and A_2 be two sequences of axis indexes. The X^2 test of independence is computed exploiting the following steps

- let E_1 be the table containing the distinct values of the axes A_1: $E_1 = g(R, A_1)$
- let E_2 be the table containing the distinct values of the axes A_2: $E_2 = g(R, A_2)$

$$\mathcal{T}_{X_i^2}(R, A_1, A_2) = \sum_{i \in range(E_1)} \sum_{j \in range(E_2)} \frac{(A_{i,j} - E_{i,j})^2}{E_{i,j}}$$

$$where\ E_{i,j} = \frac{\sum_{k \in range(A) \wedge A[k][A_1] = E_1[i]} (m(A)[k]) * \sum_{k \in range(A) \wedge A[k][A_2] = E_2[j]} (m(A)[k])}{|A|}$$
$$and\ A_{i,j} = \sum_{k \in range(A) \wedge A[k][A_1] = E_1[i] \wedge A[k][A_2] = E_2[j]} (m(A)[k])$$

2.7.6 Examples

Histograms The histogram is a function $\mathcal{H}$ that counts the number of observations that have to the same value of a specific axis. It yields a table containing the axis value and the corresponding number of observation in the original table:

$$\mathcal{H}(R, a) = g(R, [a], ||)$$

Box plot The box plot (also called five-number summary) is a function $\mathcal{BP}$ that computes, for each distinct value of an axis, the five most important percentiles of an axis. It yields a table containing sequences of six elements:

$$\mathcal{BP}(R, a, a') = g(R, [a], min(a'), lq(a'), me(a'), uq(a'), max(a'))$$

Cross table The cross table of a logical tables R respect the two axes a_1 and a_2 can be computed by aggregating the table with the size function:

$$\mathcal{CT}(R, a_1, a_2) = G(R, [a_1, a_2], ||)$$

2.7.7 Cluster analysis

Cluster analysis is the task of partitioning the observations of a table in a set of clusters. Each cluster represents a disjoint group of observations.

k-mean Let R be a table of n observations and m axes, k be the number of cluster and a be an axis index. The k-mean clustering finds a sequence C of n indexes, such that $1 \leq C[i] \leq k$ and each observation belongs to the cluster with the nearest mean:

$$\mathcal{K}_{mean}(R, k, a) = (R, C_{min}, m(R))$$

where

$$C_{min} = argmin_C \left\{ \sum_{i \in [1...k]} \sum_{j \in range(R_i) \wedge m(R_i)[j]} || \ R[j][a] - \mathcal{M}_a(R_i, a) \ ||^2 \right\}$$

and

$$R_i = \sigma((R, C, m(R)), \lambda r.r[m] = i)$$

Hierarchical clustering Hierarchical clustering is an analysis that infers a hierarchy of partitions of observations. Namely, a hierarchical clustering algorithms produces a tree $G = (V, E)$, where each leaf node represents an observation and the root node represents the set of all observations. Hierarchical cluster algorithms can be classified as

- agglomerative: the algorithms starts from a forest containing a node for each observation. At each iteration several roots of the temporary forest can be merged into a tree.
- divisive: the algorithms starts from a unique root node that contains all the observations. Each iteration splits the observations contained into one leaf node into several new leaf nodes.

The cluster algorithms depends on the metric adopted to measure dissimilarity between pair of observation and between two sets of observations. Common metrics used for pair of observations are euclidean distance, Manhattan distance and maximum coordinate distance. Metrics used to compute the distance between two set of observations are the minimum and the maximum of pairwaise distances.

We present the "complete-linkage clustering" as an example of the possible clustering methodologies that can be provided:

1. Starts with a forest containing an isolated node for each observation
2. Selects the two nearest clusters (root nodes). Distance between two clusters is computed as the maximum pairwaise "distance" distances of their observations.
3. Merges the selected cluster trees by connecting their root to a freshly created node
4. Terminates if the forest contains only one tree, otherwise continues as in the step 2

The result of hierarchical cluster analysis can be represented using tables. Let R be a logical table of observations and $G = (V, E)$ the resulting clustering. We use $id(v)$ to represent a scalar value that uniquely identifies the node v and for each leaf of G ($v \in V \wedge \nexists v' \mid (v, v') \in E$) we use $o(v)$ to represent the index that identifies the corresponding observation ($R[o(v)]$). The cluster hierarchy can be represented by two tables C and O that satisfies the following constrains

- the table O is used to extend the logical table R (obtaining $(R, O, m(R))$) with a cluster index
- the index axis values are unique: $i \neq j \Rightarrow O[i] \neq O[j]$
- for each cluster (node $v \in V$) we define the minimum and maximum index of the contained observations:
  - for leaf nodes ($\nexists v' \mid (v, v') \in E$): $min(v) = max(v) = C[o(v)]$,

  - for non-leaf nodes: $min(v) = min\{min(v') \mid (v, v') \in E\}$, $max(v) = max\{max(v') \mid (v, v') \in E\}$

- the range of indexes of sibling nodes are disjoint: $\forall v, v', v''.(v, v') \in E \wedge (v, v'') \in E \Rightarrow [min(v'), max(v')] \cap [min(v''), max(v'')] = \emptyset$
- the table C contains the computed non-leaf nodes. It consists of three axis and the element that represent the cluster v is defined as $[id(v), min(v), max(v)]$

2.7.8 Regression analysis

2.7.8.1 Linear regression: least-squares estimation

Linear regression is a type of regression analysis. Let R be a table, X be a set of axis indexes (called regressor axes) and y be an axis index (regressand axis) such that $y \notin X$. Moreover, let β (regression coefficients) and ϵ (error terms) be two sequences of $\mid X \mid$ and $\mid R \mid$ scalar values respectively. A linear regression model assumes that the values of the axis y linearly depends on the axes X. The error terms are used to model noise to the linear relationship:

$$\forall i \in range(R) \wedge m(R)[i].R[i][y] = \epsilon[i] + \sum_{j \in range(X)} \beta[j] * R[i][X[j]]$$

An estimator of linear regression is function that infers the regression coefficients. The most common estimator is Ordinary least squares, that minimizes the sum of squared residuals (error terms) (see Section 2.5.2).

2.7.8.2 Non-linear regression: least-squares estimation

Let R be a table, $x_1, \ldots, x_n$ be a set of axis indexes (called regressor axes) and y be an axis index (regressand axis). A non-linear regression model assumes that the values of the axis y depends on the nonlinear function f of the exes $x_1, \ldots, x_n$. Namely, let β a finite sequence of scalar values:

$$\forall i \in range(R) \wedge m(R)[i].R[i][y] = f(R[i][x_1], \ldots, R[i][x_n], \beta)$$

An estimator is function that infers the regression coefficients. The most common estimator is non-linear least squares, that minimizes the sum of squared residuals (error terms) (see Section 2.5.3).

$$argmin_\beta \sum_{i \in range(R) \wedge m(R)[i]} (R[i][y] - f(R[i][x_1], \ldots, R[i][x_n], \beta))^2$$

Chapter 3

Application Scenarios

This chapter presents real applications that can be addressed using secure multiparty computation. Our goal is to present real problems and describe how they can be addressed with algorithms presented in Chapter 2. We will stress the application constraints in terms of (i) privacy (ii), network structure and (iii) computational resources.

3.1 Mobile computing

3.1.1 Picture based friend matching

Two mobile phone users (U_1 and U_2) are nearly located and they want to know their common friends by comparing the corresponding pictures. They exploit the algorithm presented in Section 2.2. We suppose that an application on the U_1 phone already trained the eigenvectors $u_1, ..., u_n$ and the mean Σ and prepared the marking images $\Omega_1, ..., \Omega_M$. We suppose that each Ω_i corresponds to a friend f_i of the user U_1. The pictures owned by the user U_2 can be complex and depicting more than one friend of U_2. The application that resides on the U_2 host can perform local computation to normalize the pictures yielding a set of images $I_1, ..., I_k$, each of them containing only one face. The privacy constraints can drive different scenarios:

- both users discover if there exists at least one common friend, without discovering the corresponding identity. Namely, they jointly compute

$$min\{\mid \bar{I}_i - \Omega_j \mid .\forall i, j\} < \epsilon \; where \; \bar{I}_i = (u_1^T(I_i - \Psi), ..., u_n^T(I_i - \Psi))$$

- both users discover the number of common friends, without discovering the corresponding identities. Namely, they jointly compute

$$\sum_C 1 \; where \; C = \{\forall i. min\{\mid \bar{I}_i - \Omega_j \mid \forall j\} < \epsilon\} \; and \; \bar{I}_i = (u_1^T(I_i - \Psi), ..., u_n^T(I_i - \Psi))$$

- both users discover the identities of common friends. Namely, U_1 discovers F_1 while U_2 discovers F_2 where

$$F_1 = \{i.\exists j.(i,j) \in F\} \quad F_2 = \{j.\exists i.(i,j) \in F\} \quad F = \{(i,j). \mid \bar{I}_i - \Omega_j \mid < \epsilon\}$$

In all cases the user U_1 is not authorized to discover the pictures $I_1, ..., I_k$ and U_2 is not authorized to discover the "trained-space" parameters.

Each user can own 1000 images depicting at most 5 friends. Moreover, each user knows at most 100 friends. Since the users are nearly located, an ad-hoc wireless (54 Mbps) connection can be used. The two smart phones can be used to perform the computation, alternatively:

- the two users involve two independently trusted services (e.g. the service hosting their pictures)
- the two users involve a single untrusted service

3.1.2 "P2P" positioning system

Location-aware mobile applications rely on two main infrastructures: the Global Positioning System (GPS) is adopted as ubiquitous available service for outdoor car navigation, WiFi coverage is used to increase precision or indoor positioning. In several scenarios, exploiting these approaches is not sufficient. The GPS infrastructure is weak in all cases where the sky is covered: indoor, tunnels, between high buildings. Moreover its error is statistically near twenty meters. The adoption of a global WiFi based positioning system has the drawback to require a huge measurement campaign. A peer to peer positioning system can enhance the existing methodologies, by providing a further measure. In this scenario several mobile peers dynamically discover their positions (e.g. by exploiting the existing techniques or the p2pps itself) and act as further information sources for other peers.

A peer Q exploits the application to discover its location $p_Q = (x_Q, y_Q, z_Q)$. We assume the availability of n other peers P_i that know their own positions $p_i = (x_i, y_i, z_i)$. We also assume that it is possible to evaluate the distance r_i between the peer Q and each other peer P_i. This task can be accomplished exploiting several electromagnetic measures, ranging by WiFi beam signals, FM signals and bluethooth.

For each peer P_i, the position p_i and the distance r_i identify a sphere surface:

$$r_i^2 = (x_Q - x_i)^2 + (y_Q - y_i)^2 + (z_Q - z_i)^2$$

If there is no measurement error, the point p_Q relies on the intersection of the sphere shapes. In presence of measurement errors, the position p_Q can be approximated by solving the non-linear least squares (see Section 2.5.3) for the n quadratic equations.

A different approach involves the linearization of the equation system. We add and subtract the position p_n to all $n-1$ equations, obtaining:

$$\begin{aligned} r_i^2 = \;& (x_Q - x_n + x_n - x_i)^2 + (y_Q - y_n + y_n - y_i)^2 + (z_Q - z_n + z_n - z_i)^2 \\ r_i^2 = \;& (x_Q - x_n)^2 + 2(x_Q - x_n)(x_n - x_i) + (x_n - x_i)^2 + \\ & (y_Q - y_n)^2 + 2(y_Q - y_n)(y_n - y_i) + (y_n - y_i)^2 + \\ & (z_Q - z_n)^2 + 2(z_Q - z_n)(z_n - z_i) + (z_n - z_i)^2 \end{aligned}$$

Then we remove the equation n to all others $n-1$ equations:

$$\begin{aligned} r_i^2 - r_n^2 = \;& (x_Q - x_n)^2 + 2(x_Q - x_n)(x_n - x_i) + (x_n - x_i)^2 + \\ & (y_Q - y_n)^2 + 2(y_Q - y_n)(y_n - y_i) + (y_n - y_i)^2 + \\ & (z_Q - z_n)^2 + 2(z_Q - z_n)(z_n - z_i) + (z_n - z_i)^2 + \\ & -(x_Q - x_n)^2 - (y_Q - y_n)^2 - (z_Q - z_n)^2 \end{aligned}$$

$$\begin{aligned} r_i^2 - r_n^2 = \;& 2(x_Q - x_n)(x_n - x_i) + (x_n - x_i)^2 + \\ & 2(y_Q - y_n)(y_n - y_i) + (y_n - y_i)^2 + \\ & 2(z_Q - z_n)(z_n - z_i) + (z_n - z_i)^2 \end{aligned}$$

Each equation can be written as

$$b_i = (x_Q - x_n)(x_i - x_n) + (y_Q - y_n)(y_i - y_n) + (z_Q - z_n)(z_i - z_n)$$

where

$$b_i = \frac{r_n^2 - r_i^2 + (x_n - x_i)^2 + (y_n - y_i)^2 + (z_n - z_i)^2}{2}$$

The position p_Q is then $p_q = s + p_n$ where s is the solution of the linear system of equations $As = b$:

$$A = \begin{pmatrix} x_1 - x_n & y_1 - y_n & z_1 - z_n \\ \dots & \dots & \dots \\ x_{n-1} - x_n & y_{n-1} - y_n & z_{n-1} - z_n \end{pmatrix} \quad \vec{s} = \begin{pmatrix} x_Q - x_n \\ y_Q - y_n \\ z_Q - z_n \end{pmatrix} \quad \vec{b} = \begin{pmatrix} b_1 \\ \dots \\ b_{n-1} \end{pmatrix}$$

If the resulting system has three linear independent equations (namely there are four peers that know their own positions) and we assume that there are no measurement errors, then trilateration is reduced to solving the resulting linear system of equations (see Section 2.5.1). If the linear system is overdetermined or the measurements are affected by errors, the nearest solution can be discovered by solving the Linear least squares problem (see Section 2.5.2). Several privacy settings are possible:

- Some peers disclose their positions p_i (e.g. public hot-spots, infrastructure points etc.), while other ones maintain this information private.
- The peer Q knows all distances r_i and wants to keep this information private.
- Each peer P_i knows the distance r_i and wants to keep this information private.

Point-to-point networks can be assumed by exploiting ad hoc wifi networks. However, this approach allows "side-channel attacks", possibly allowing the peers P_i to discover the distance r_i. A more conservative approach forces all point-to-point communication to be mediated by one or more public Internet services. In this case we assume UMTS connectivity between peers and the services and high speed network (e.g. 100 Mbs) among the services.

3.1.3 Private navigation system

A user U with a mobile phone requests from a mapping service S the shortest path from his own position to a destination. The mapping service stores the street network as a graph. For copyright reason the user is not be able to discover the street graph. Moreover, for privacy reason the service is not able to discover the position of the user, its destination and the computed path.

This scenario is a specific instance of 2.6.1. The edges of the graph and the weight function are known by the map service, while the origin, destination and the computed path are known by the user.

The mobile phone and the service can be assumed interconnected by a UMTS network (e.g. 1mbs bandwidth). Moreover the service can assume the availability of a 100Mb of phone storage that can be used as cache.

3.2 Investigation agencies

3.2.1 Computer controlled identification

The High-definition CCTV cameras and facial recognition can be combined to provide "a form of mass surveillance" [6]. The investigation agencies are interested to find the last locations visited by a criminal. For this reason they can involve several public office/CCTV owners to collaborate, searching for recorded images that depict the wanted criminal (or missing person).

To accomplish face recognition the algorithm presented in Section 2.2 can be exploited. The first scenario assume that the investigation agency previously computed the eigenvectors $u_1, ..., u_n$ and the mean Σ and prepared the marking images $\Omega_1, ..., \Omega_M$. We suppose that the images $\Omega_i, i \in W$ correspond to images of the wanted criminal and images $\Omega_i, i \in \{1 \dots M\} \setminus W$ are randomly selected. Each CCTV owner can locally process the video stream to extract images and to transform them, obtaining the normalized pictures $I_1, ..., I_k$, each of them containing only one face.

The intelligence agency can repeat the procedure for each CCTV owner. Searching a positive match for a CCTV owner means computing for a given threshold ϵ

$$min\{\mid \bar{I}_i - \Omega_j \mid .\forall i \in \{1 \dots k\}, j \in W\} < \epsilon \; where \; \bar{I}_i = (u_1^T(I_i - \Psi), ..., u_n^T(I_i - \Psi))$$

The CCTV owners can not discover the "trained-space" parameters (Ω_i, u_j, Σ) and are not allowed to know the result of the computation (if a positive match is found). The agency can not discover the pictures hosted by the CCTV owners. Standard DSL connections (10MBps downlink, 0.5MBps uplink) connect the

CCTV owners to the agency. Each CCTV owner stores 2000 images daily. Several variants or extension to the application can be investigated:

- The implementation of a surveillance agency that certificates the pictures that can be matched to avoid the intelligence agency to search for arbitrary person.
- The capability of the investigation agency to obtain the matched images to enable further analysis.
- An symmetric approach, where the CCTV owners compute their own training spaces and the agency sends a mugshot.
- The capability of the investigation agency to search for more than one criminal.

3.3 Intelligence agencies

3.3.1 Shortest path

Coordinating army of different nations requires sharing military secrets. We provide an example suggested by the interview I12 used to extract the use case 2.3.2.3 of Deliverable 1.2 [9]. Two army want to compute the "safest" path to travel in a war-zone. The war-zone can be represented as a weighted graph: each edge represents a road and its cost represents the corresponding risk. A naive model of the risk of a path between two nodes is represented by the sum of the risks of the traversed roads (alternative models include maximum and product). Computing the "safest" path is a single pair shortest path problem (see Section 2.6.1). Several scenarios are possibles:

- The army A knows the risk graph, the army B requests the path from the source to the target. Only B discovers the "safest" path. The risk graph, the source and the target must remain secret.
- Both army A and B know the source and the destination, but each army knows a subset of the edges of the risk graph. Both army obtain the computed path.
- Both army A and B knows the source and the destination, but each army has different estimation of the edge costs. The costs of the risk graph are computed as the maximum of the costs estimated by the participants (alternative estimators are minimum, sum and product). Both army obtain the computed path, but the computed risk graph remains secret.

3.4 Finance and banks

Several interviews summarized in Section 2.3.6 of Deliverable 1.2 [9] stress the requirement of privately execute auctions. The presented problem requires business processes that stand for several days and allows the interaction of hundreds of companies.

3.4.1 Dutch auctions

A Dutch auction is a type of auction in which the auctioneer begins with a high asking price which is lowered until some participants are willing to accept the auctioneer's price.

Let O be a set of offers $o_1 = (q_1, p_1), ..., o_n = (q_n, p_n)$, where each offer o_i guarantees the quantity q_i at price p_i. It is possible to define the "supply curve" $s(p)$ for the auction by computing

$$s[p] = \sum_{(p_i,q_i)\ if\ p_i \leq p} q_i$$

3.4.1.1 Bonds repurchases

The use case 2.3.6.3 describes a scenario involving dutch auctions. A company has issued bonds on the market that are called "corporate bonds". The company wants to buy back its own bonds, for example to avoid interest on unpaid portion of outstanding debts. The company sends to the noteholders a buying range $[min, max]$ that represents the acceptable price range of bonds. Each noteholder selects a selling price and the corresponding acquirable bonds, participating to define the supply curve s. The company owns the goal function g that computes the goal price starting from the supply curve. Several goal functions are possible, for example:

- select the smaller available price in order to minimize the price/bound ratio

$$g(s) = min\{p.s[p] > 0\}$$

- select the price that allows to use less than MAX

$$g(s) = max\{p.s[p] * p \leq MAX\}$$

- select the price that guarantees to buy back at least MIN bounds

$$g(s) = min\{p.s[p] \geq MIN\}$$

In the presented scenario there are n bidders and one auctioner. Each bidder knows his own price and quantity, while the auctioner knows the strategy. The bidders and the auctioner discover the final price, while the auctioner discovers only the bidders that offer a price smaller than the selected buy-back price. No one discovers the supply curve, the bidders do not discover the strategy, and the bidder offers remain private. Since the auction can require several days, the bidders and the auctioner can accept to disclose the current repurchase price to plan further business activities.

3.4.1.2 OpenIPO

OpenIPO [2] is an approach to initial public offerings. OpenIPO uses an auction process to find the intersection of supply and demand in pricing Initial Public Offerings. An OpenIPO auction is open for bids for some weeks prior to the effective date of the offering. Once the bidding concludes, the auction assembles the bids by computing the supply curve s and finds the greatest bid price that will sell all the offering shares n:

$$p = max\{p.s[p] \geq n\} \text{ where } s[p] = \sum_{(p_i,q_i)\ if\ p_i \geq p} q_i$$

OpenIPO works with a Pro-Rata Allocation policy. Selected the selling price p, if the number of successful bid shares is equal or smaller to the number of shares offered ($s[p] \leq n$) then each winning bidder ($p_i \geq p$) receives the requested shares (q_i). If the number of successful bid shares exceeds the number of available shares in the offering ($s[p] > n$), allocations are on a pro-rata basis and rounded to a predefined multiples of shares (m). Namely, each winning bidder receives the number of shares $\left\lfloor \frac{n}{s[p]} * \frac{q_i}{m} \right\rfloor * m$.

3.5 Energy market

Use cases 2.3.2.1 and 2.3.2.2 of Deliverable 1.2 [9] highlight the usefulness of privacy preserving optimization of production and allocation. We present an application tailored to the smart grid environment.

3.5.1 Energy consumption scheduling

The increasing demand for energy and the adoption of heterogeneous and variable energy sources like solar power and "local/distributed" generation require a smart management of energy consumption. For example, it is necessary to schedule high-load appliances to reduce the difference between the maximum and the average energy requirements. The adoption of "smart meters" and distributed network can be exploited to provide a global scheduling of energy consumption.

The paper [5] exemplifies a scenario involving a smart power system with several load subscribers and only one energy providers. Let N be the set of subscribers. Each subscriber $n \in N$ own a set of electric appliances $a \in A_n$. For each appliance we define the consumption scheduling vector

$$x_a = [x_a^1, \ldots, x_a^{24}]$$

Each real value x_a^h represents the energy consumption scheduled for the device a during the hour h. $E_a = \sum_{h=1}^{24} x_a^h$ is the total energy allocation during one day for the appliance a. E_a is pre-determined and represents one of the constraints of the system. For example an electric vehicle consumes a predefined amount of energy to recharge its battery and supply the required driving range). For each appliance we define also the maximum consumption constraint vector

$$M_a = [M_a^1, \ldots, M_a^{24}]$$

and the minimum consumption constraint vector

$$m_a = [m_a^1, \ldots, m_a^{24}]$$

The constraint vectors define if a scheduling is acceptable. $M_a^h = 0$ can model an electric vehicle that must complete its recharge before the user wake-up or a washing machine that can be used only when the user is inside the apartment. Moreover, the maximum constraint models the appliances that have a maximum power level of usage (e.g. recharge rate). Similarly, the minimum power level can be used to represent the minimum amount of energy required by the stand-by device. These vectors constraint the system as follows:

$$m_a^h \leq x_a^h \leq M_a^h$$

We can define the total load per hour as

$$L_h = \sum_{n \in N} \sum_{a \in A_n} x_a^h$$

Let $C_h(L)$ be a function representing the total cost of producing/delivering the energy required to support the load L at the hour h. Notice that the cost of producing the same energy can be different during the day. The optimal scheduling is the one satisfying all constraint and minimizing

$$\sum_{h=1}^{24} C_h(\sum_{n \in N} \sum_{a \in A_n} x_a^h)$$

If C_h are linear functions of the total load ($C_h(L) = c_h * L$) then the problem can be addressed using linear programming 2.1.

All participants can be involved in the computation. However we require that the power provider is not able to discover the constraints and the power allocation of each customer, to avoid commercial profiling. Similarly, the customers are not able to discover any information about other participants (possibly they can discover the number of participants). Similarly, the total load per hour can be discovered only by the power supplier.

The number of participants and constraints strictly depends on the size of the smart grid. For example (i) the management of one building consist of the collaboration of 50 customers, (ii) the management of one district requires the collaboration of 1000 customers, (iii) the management of one city requires the collaboration of 100000 customers. Moreover, several extension to the proposed application can be investigated:

- The power supplier can be allowed to discover the scheduling vectors $\sum_a x_a$ of each user. This enables the power provider to bill the service consumption and to enforce the scheduling policies.
- The power level domain can be continuous or discrete.
- A subset of the constraints dynamically change, requiring a new allocation.
- The energy provider is able to provide different power peaks during the day (e.g. solar power stations)
- A reward model can provide trust incentives. Each peer is charged b_n dollars. Suppose that the billing scheme is independent of the user and represented as a function $b_n = b(\sum_{a \in A_n} x_a^1, \ldots, \sum_{a \in A_n} x_a^{24})$. The function should guarantee fair behavior of customers.
- The cost function can be non-linear e.g.
  - increasing cost functions: $L_1 \leq L_2 \Rightarrow C_h(L_1) \leq C_h(L_2)$
  - convex cost functions
- The scenario can involve more energy providers and power lines, thus requiring to handle the problems presented in Section 2.6.2.

3.6 Network management

3.6.1 Inter-autonomous system routing

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 take benefits by more collaborative ASs. The number of current (2012) ASs is estimated as 40000 [1].

3.6.1.1 Routing

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

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

- Vertexes of the network are partitioned into ASes in order to represent multihomed domains.
- The cost of an edge connecting vertexes belonging to the same AP can be computed by the AP itself, by running a local routing protocol. This cost is a commercial secret that the AP does not want to share with the competitors/customers
- The cost of edges crossing AS boundaries are regulated by contracts and can not be discovered by other ASs.

The goal of each AS is to compute, for each destination vertex (d) and internal source vertex (s), the vertex to which the packet has to be delivered in order to reach the best path and the corresponding total cost (see Section 2.6.1). For making computations based on these distributions, all-to-all connectivity is available. This might be used for setting up routing tables, but when a particular packet is being handled, the routing decision must be handled by using connections between neighbouring nodes only.

3.6.1.2 QoS aware routing for 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 among border gateways, forcing AS to rely on selfish routing techniques. The inter-AS best path is computed regardless 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 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 packet is being handled.

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

3.6.2 Intrusion detection

A key enterprise interoperability issues is collaborative security. The increasing inter-connection among partner allows attackers to coordinate distributed behavior to achieve a malicious goal. In this field a collaborative intrusion detection can provide a global view of the attacker activities. A typical example of coordinated attack is the IP-spoofing technique, that is used by an intruder that wants to appear like a third party while interacting with the target host.

A collaborative security infrastructure should be able to correlate events gathered by independent Intrusion Detection Systems (IDSs) and to analyze the intruders attack strategy. This task requires the IDSs to share a common knowledge of the possible intruder behaviors (e.g. expressed as Petri Nets or Goal Trees). The goal of the participants is to replay the attacker activities on the predefined behavioral models, in order to understand the attacker state and to enable pro-active updates to the local IDSs.

In general it is not possible to replicate all observable events raised by an IDS in a collaborating environment: the event rates gathered by local IDS is so high that the bandwidth requirement and processing power are not satisfiable. Moreover, information leakage must be prevent during the collaboration.

3.7 Statistical applications

Several interviews (e.g. use cases 2.3.1, 2.3.4 and 2.3.5) of Deliverable 1.2 [9] pointed out the requirement of executing statistical analyses and aggregations of information owned bu different parties. Instead of focusing on a specific use case, we introduced in Section 2.7 a set of general statistical problems, which can be composed to satisfy the requirements of real applications.

Chapter 4

Deployment scenarios

The applications described in Chapter 3 can be deployed in several environments. This section identifies the benefit of adopting specific settings to reduce the computational and network costs of the SMC solutions. One of the major drawbacks of secure multiparty computation is efficiency. Several protocols require high communication complexity due to extensive use of secret sharing. Recent works (e.g. [3, 7, 8]) investigated the adoption of hardware support to increase efficiency of computations. All these approaches share a key requirement: the availability of a tamper-proof hardware that allows to trust the execution of the algorithm. In the following section we briefly describe different deployment scenarios.

4.1 Fully trusted security modules

The system consists of a set of hosts. Hosts are untrusted and intercommunicate exploiting point-to-point secure and reliable channels. A fully connected communication topology is not mandatory; the application specific constraints determine the host connection topology.

Each host owns a "security module", implemented using a tamper-proof hardware. Each security module is "trusted" by all involved peers (hosts and security modules). Security modules are directly connected to the corresponding host. In order to communicate to another security module, a "trusted" device must involve the "untrusted" peers in the path to accomplish the remote communication. This means that messages delivered by "trusted" agents can be dropped (or delivered to the wrong trusted agent).

All security modules are able to compute a finite set of functions F over inputs provided by the corresponding host. In general the security module can be able to directly perform the algorithm required by the multiparty computation, however:

- The security modules are subject to severe resource constraints. Solutions that involve such devices must describe the computational power, the persistent data storage and the memory available for the security modules (e.g. smart cards). It is crucial to compare the performance of the system with respect to the "untrusted" scenario and to verify that the resource constraints are satisfied.

- Since all security modules are trusted by all participants, the set of functions they can compute represents an agreement among parties. This is a key issue in real world scenarios.

4.2 Hardware Tokens

A set of "hosts" represents the "untrusted" peers that intercommunicate exploiting secure channels. Host S can send an "hardware token" T to a receiver C. The hardware token allows the receiver to execute a secret program specified by the sender tokens a fixed amount of times, by exploiting a tamper-proof device. The adoption of hardware support allows S to trust T and vice-versa, but does not require C to trust the sender or the token. The host C can "locally" communicate with T and remotely with S. There is no

direct communication channel between the token T and the issuer S, but their communications involve the receiver C that can interfere.

The main goal of this architecture is to replace most of the expensive inter-host communications with cheap communications between C and T. The resource constraints of the hardware tokens must be clearly identified and the resulting performance compared with respect to the distributed environment. Similarly, the expressiveness of the underlying hardware and the "shared" machine model must be defined.

Bibliography

[1] Cidr report. `http://www.cidr-report.org/as2.0/`, 2012.

[2] WR Hambrecht + Co. Openipo: How it works. `http://www.wrhambrecht.com/ind/auctions/openipo/`.

[3] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Embedded sfe: Offloading server and network using hardware tokens. In Radu Sion, editor, *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 207–221. Springer, 2010.

[4] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):103–108, 1990.

[5] A.-H. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, and R. Schober. Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid. In *Innovative Smart Grid Technologies (ISGT), 2010*, pages 1 –6, jan. 2010.

[6] Federal Bureau of Investigation. Next generation identification. `http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/ngi/ngi2`, 2006.

[7] Ahmad-Reza Sadeghi, Thomas Schneider, and Marcel Winandy. Token-based cloud computing. In Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi, editors, *TRUST*, volume 6101 of *Lecture Notes in Computer Science*, pages 417–429. Springer, 2010.

[8] Steve R. Tate and Roopa Vishwanathan. General secure function evaluation using standard trusted computing hardware. In *PST*, pages 221–228. IEEE, 2011.

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