



D12.2

Adversary, Trust, Communication and System Models

Project number:	609611
Project acronym:	PRACTICE
Project title:	Privacy-Preserving Computation in the Cloud
Project Start Date:	1 November, 2013
Duration:	36 months
Programme:	FP7/2007-2013
Deliverable Type:	Report
Reference Number:	ICT-609611 / D12.2 / 1.0
Activity and WP:	Activity 1 / WP 12
Due Date:	April 2015 - M18
Actual Submission Date:	30 th April, 2015
Responsible Organisation:	TUDA
Editor:	Ferdinand Brassler, Hiva Mahmoodi, Ahmad-Reza Sadeghi
Dissemination Level:	Public
Revision:	1.0 (r-2)
Abstract:	Adversary, Trust, Communication and System Model: This deliverable is the outcome of Task 1.2.2 and summarizes the assumptions on the adversaries, trust, communication and system models for the applications scenarios identified in D12.1.
Keywords:	Scenario, Requirements, Trust Model, Adersary Model, Communication Model, System Model



This project has received funding from the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 609611.

Editor

Ferdinand Brasser, Hiva Mahmoodi, Ahmad-Reza Sadeghi (TUDA)

Contributors (ordered according to beneficiary numbers)

Martin Haerterich (SAP)
Hiva Mahmoodi (TUDA)
Ferdinand Brasser (TUDA)
Ágnes Kiss (TUDA)
Michael Stausholm (ALX)
Cem Kazan (ARC)
Sander Siim (CYBER)
Manuel Barbosa (INESC PORTO)
Bernardo Portela (INESC PORTO)
Meilof Veeningen (TUE)
Niels de Vreede (TUE)
Antonio Zilli (DTA)
Stelvio Cimato (UMIL)

Disclaimer

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose subject to any liability which is mandatory due to applicable law. The users use the information at their sole risk and liability.

Executive Summary

This deliverable defines different security models (i.e., adversary, trust, communication and system models) for the real-world use-case scenarios of secure computation technologies. The security models are provided for thirteen application scenarios identified in deliverable D12.1 of the project. After identification of the participants' roles, their behaviour is evaluated in an adversary model, in order to determine which participants can be assumed to act in a malicious or benign way in each application scenario. Akin to the adversary model, the trust model indicates the level of trust given to each participating party as well as trust assumptions that are oft made implicitly. The impact of using trusted hardware is also examined in the trust model. The communication model states explicit and implicit assumptions about different communication channels among participants. This model investigates features such as online availability of the participants, simultaneous and transactional communication as well as bandwidth, latency and reliability of the communication channel. The system model deals with the capabilities and properties of the parties participating in the application scenario regarding today's heterogeneous computation landscape. It considers benchmarks such as computational power, amount of memory, parallelism of computation, reuse of services, etc. Further, a list of security goals such as correctness, verifiability, confidentiality, privacy and indistinguishability are specified for each application. Achieving these security goals guarantees that the algorithm in question executes as expected and correct results are generated.

To show the ability of the PRACTICE architecture to implement a broad range of secure computation applications, the thirteen use case scenarios examined in this deliverable are mapped to the overall architecture of the project. The potential implementation of each application scenario in the project's architecture is evaluated by identifying the tools and components of the architecture that are involved in the implementation of each scenario. This mapping shows that the PRACTICE architecture is general and suitable to implement a wide range of business applications with its comprehensive architecture.

The use of trusted hardware is an orthogonal approach when computation on private data needs to be outsourced. This document provides a summary of the various trusted hardware devices and the functionalities they provide. It then elaborates how secure multi-party computation can be performed in the cloud environment using trusted hardware (Intel SGX in particular).

The provided models play an important role in development and implementation of solutions for the problems mentioned in the studied application scenarios. The collected per-application security models are summarized and compared with each other in order to derive general conclusions. These results provide insight into the possible implementations and highlight important criteria for the development of applications which benefit secure computation techniques in the era of distributed, cloud and mobile computing.

Contents

1	Introduction	1
1.1	Scope of the Deliverable	1
1.2	Relation to Other Parts of the Project	1
1.3	Application Scenarios	3
1.3.1	From Analysis to Implementation	3
1.3.2	Categorization of Application Scenarios	3
2	Model Description	5
2.1	Participant Roles	7
2.2	Adversary Model	7
2.3	Trust Model	8
2.3.1	Trusted Hardware	9
2.4	Communication Model	10
2.5	System Model	11
2.6	Guarantees	12
3	Integration of Application Scenarios in the Project Architecture	13
3.1	Overall Architecture	13
3.2	Implementation of Application Scenarios: An Overview	14
4	Application Scenarios	18
4.1	Joint Business Applications	19
4.2	Joint Studies Applications	41
4.3	Location Sharing Applications	58
4.4	End User Applications	66
5	Formalizing Security Models	75
5.1	The Overall Security Model: Ideal versus Real	76
5.1.1	Execution Model: Trust and Communication Models	76
5.1.2	The Ideal-World Trusted Party	77
5.1.3	Security Definition and Adversary Model	77
5.2	Corruption-Dependent Trusted Parties and Verifiability	77
5.2.1	Model of Universally Verifiable Secure Function Evaluation	78
5.3	Combining Secure Computation Engines for Validation	80
5.3.1	Trusted Party for Validation	80
5.4	Secure Computation in the Random Oracle Model	81
5.4.1	Adapting Security Models	82
5.4.2	Consequences for Practical Security	82

6	Secure Multi-Party Computation with Trusted Hardware	84
6.1	Trusted Hardware	85
6.1.1	Hardware Security Modules	85
6.1.2	Trusted Execution Environments	86
6.2	Secure Multi-party Computation with SGX	88
6.3	Adversary and Trust Model for SMC with Trusted Hardware	90
7	Conclusion	92
7.1	Adversary Model	92
7.2	Trust Model	97
7.3	Communication Model	99
7.4	System Model	99
7.5	Guarantees	100
7.6	Final Statements	101
8	List of Abbreviations	102

List of Figures

1.1	Interdependency chart for Activity 1	2
3.1	Overall Architecture	17
4.1	Colours Used For Different Tools	18
6.1	SMC protocol with SGX - non-confidential operations	88
6.2	SMC protocol with SGX - operations revealed to input parties	89
6.3	SMC protocol with SGX	90

List of Tables

2.1	Application scenario template.	6
3.1	Candidate Tools for Implementation of Application Scenarios	16
7.1	Comparative List of Adversary Models	97
7.2	Application of Trusted Hardware	98

Chapter 1

Introduction

1.1 Scope of the Deliverable

There are many real-world application scenarios which would benefit from secure computation techniques and concepts. Successful and usable implementation of these use cases requires exact and detailed models for specification of adversaries, trusted parties/components, communication channels and system requirements. The goal of this document is to capture these models for various application scenarios. We specify an *adversary model* with the aim of capturing the strength of realistic adversaries. This model will help in providing an adequate level of security for a particular application. The *trust model* is defined to determine which levels of trust can be assumed. The possibility of leveraging trusted hardware in a scenario is also evaluated in the trust model. Different communication channels and their related explicit and implicit assumptions are combined in the *communication model* of each application scenario. We describe the capabilities and functional properties of different participants of every scenario in an individual *system model* for each scenario. This includes investigation of properties such as computation power, connection bandwidth, relevance of parallelism, etc.

We provide a summary of the application scenarios which are considered as the basis of this deliverable. Then we define the terms used in the document and provide a tabular template for compilation of the different models (adversary, trust, communication and system models) for every scenario. Chapter 2 is devoted to describing the meaning and specifics of any of the models discussed in this deliverable. The connection between application scenarios and the overall architecture of the project is presented in Chapter 3. The tabular representation of the individual application scenarios and their related models comprises the main part of this document (Chapter 4). A precise real/ideal-world formalisation of the security model is presented in Chapter 5. Chapter 6 describes how trusted hardware can be used as a(n) alternative/complement to SMC techniques to perform secure computation, particularly in the cloud. The deliverable ends with a summary, comparison of security models provided for different scenarios and concluding remarks in Chapter 7.

1.2 Relation to Other Parts of the Project

This deliverable has a close and tight relationship with a number of other tasks in different work packages of the project PRACTICE. The application scenarios identified in the first deliverable of WP12 serve as input to this deliverable. These scenarios are analysed in more details in this deliverable, with focus on their individual adversary, trust, communication and system models.

The analysis results of this work packages will feed into the work package about the *Analysis of Existing Techniques* (WP11). Furthermore, the results of this work package will be input along with

the results from WP11 to WP13, which is concerned with *Protocol Specification and Design*. This is according to the interdependency chart for the activity A1 from the project depicted Figure 1.1.

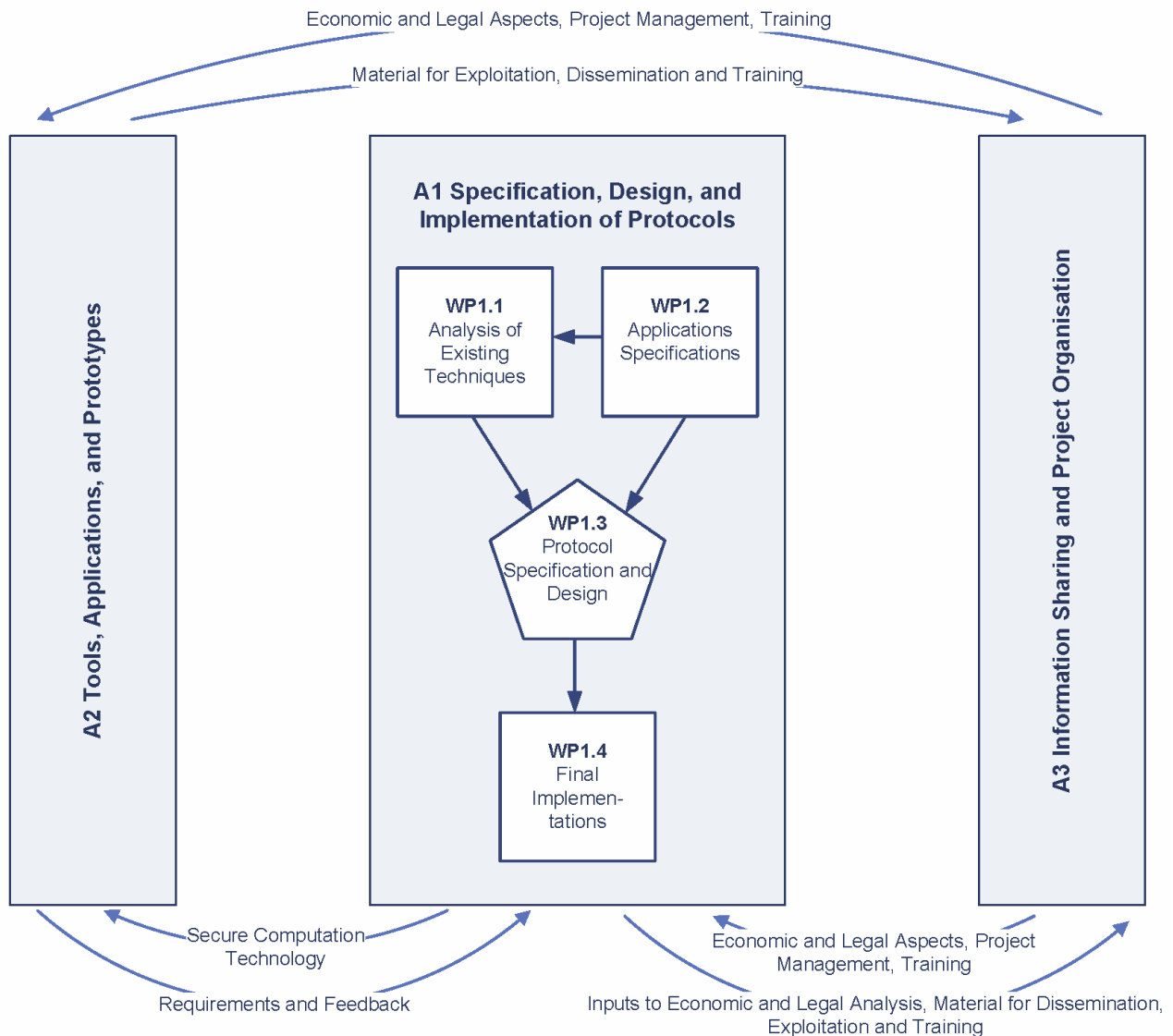


Figure 1.1: Interdependency chart for Activity 1

The application scenarios defined by WP12 and documented in the deliverables D12.1 and D12.2 of this work package are analysed in WP11. The goal of the application scenario based analysis is to evaluate the performance of existing protocols in these scenarios.

The results collected and compiled in this deliverable will be used to enable the mapping of the known techniques and results of secure computation to the targeted applications. The adversary, trust, communication and system models described in this deliverable are fed to the work package WP13. The goal of WP13 is to design protocols which close the the gaps existing between the existing protocols and protocols needed to achieve the desired solutions. The security models collected and specified in this deliverable will be considered in WP13 in order to construct protocols that fit more closely the needs of a specific application with regard to the models provided for that application. The results mentioned in this document are used in design, development and optimization of efficient protocols for applications of specific interest, like application scenarios specified in the results of

the current project deliverable. Evaluating the effect of using (trusted) hardware in an application scenario helps in designing lightweight hardware-assisted protocols or trust anchors for large-scale and distributed applications.

1.3 Application Scenarios

This deliverable contains thirteen application scenarios, which are introduced in this section. The application scenarios were collected in D12.1 and reflect a wide range of use cases for secure computing.

1.3.1 From Analysis to Implementation

The scenarios of this deliverable were collected among the project partners in the deliverable D12.1 and represent use cases which are interesting for the partners but also for the members of the project's advisory board.

Furthermore, considering a broad variety of scenarios in the analysis and design of the projects helps in building with PRACTICE a universal framework for SMC in the cloud. This is why *all* application scenarios from D12.1 were analysed in this deliverable.

Among the scenarios considered in this deliverable are the two scenarios which are evaluated in depth in the work packages *Secure Statistics* (WP23) and *Supply Chain* (WP24). The remaining scenarios, although not designed and implemented in the same detail as those in WP23 and WP24, build a valuable starting point for further exploitation of the project results.

1.3.2 Categorization of Application Scenarios

The application scenarios are grouped thematically into four different categories. These categories are *joint business applications*, *joint studies applications*, *location sharing applications* and *end user applications*.

A short overview of all scenarios and categories is given in this section. The application scenarios are analysed in detail and illustrated in Chapter 4 as well as in the referenced work packages.

The first category *joint business applications* involves companies that are interested in cooperating with each other without revealing sensitive internal data of their company. Scenarios from this category use secure computation to jointly evaluate calculations, e.g., supply chain optimization, based on sensitive company data without revealing the data itself. *Joint business applications* that are investigated in this deliverable are:

- **Aeroengine Fleet Management:** This scenario describes a system that enables the optimization of the maintenance repair and overhaul process for the engine sector of the aeronautic supply chain. Maintenance plans can be calculated without revealing the participating companies data. An in-depth analysis of this use case as well as a prototype implementation are the objectives of Work Package (WP) 24.
- **Consortium Gathering Information from Its Members:** A consortium would like to gather information from its members, e.g., benchmarking economic results. Secure computation enables competing companies to contribute their private data to the consortium without risking disclosure of the individual data.
- **Platform for Auctions:** Multiple parties negotiate in an auction without revealing their bids. Exemplary markets are spectrum and electricity auctions.
- **Platform for Benchmarking:** A privacy preserving platform for benchmarking between business partners enables a trustworthy assessment. Partners can evaluate each other regarding different

factors, i.e., credit card rating, without divulging losing sensitive company data. A prototype will be implemented in WP 23.

- **Tax Fraud Detection:** Detecting tax frauds is an important scenario in which state entities are interested in analyzing precise financial data of companies. With the help of secure computation, a precise analysis of money flows can be executed without the necessity to reveal the companies' sensitive financial data to the revenue office.

In the second area, namely *joint studies applications*, sensitive data of many individuals or entities is used for studies and statistics without exposing the individual's data at any time. In this area we discuss the following scenarios:

- **Joint Statistical Analysis Between State Entities:** In some cases the law forbids the compilation of so-called super-databases from the individual datasets of different state entities. To enable a joint study across different entities, secure computation can be used to join data bases in a privacy-preserving manner that fulfils the legal requirements.
- **Privacy-Preserving Genome Studies Between Biobanks:** Biobanks from different countries can perform a joint genome-wide association study using each other's data without breaching the donors' privacy using secure computation.
- **Privacy-Preserving Personal Genome Analyses and Studies:** Similar to the service offered by 23andMe [5], donors can submit their genome data and enter their phenotype data to receive feedback on genetic associations with specific illnesses and disorders. Secure computation can be used to prevent any mishandling of the donors' data.
- **Surveys on Sensitive Data:** A cloud system that provides a platform for privacy-preserving surveys. A survey creator submits a survey to the platform that is then filled with opinions from invited participants. Using secure cloud computing, the survey is evaluated and only the result is sent back to the creator. Thus, with the help of secure computation, the participants' input data can be protected. This scenario is elaborated further in WP 23.

Privacy-preserving *location sharing* is of relevance in the following two scenarios:

- **Location Sharing with Nearby Contacts:** Location information of smart phone users is sensitive, yet useful for social activities where contacts meet. With the help of secure computation, proximities can be calculated without revealing actual location data.
- **Privacy-Preserving Satellite Collision Detection:** Different countries wish to forecast collisions between their satellites without revealing the exact location and trajectory of their satellites.

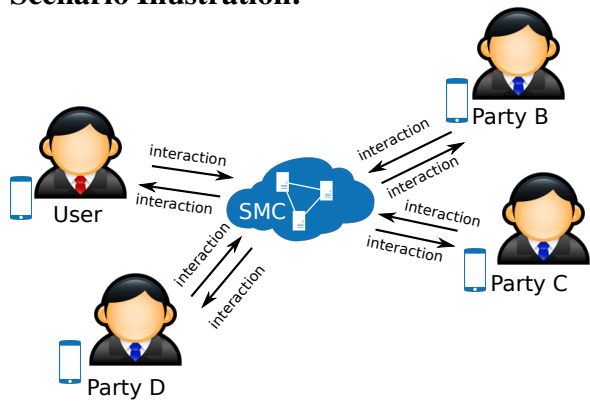
The last group of scenarios described in this document are *end user applications*. These scenarios aim towards increasing the end user's privacy when using cloud services. The described applications in this area are:

- **Key Management:** With the increasing number of devices used by an end user, cryptographic keys need to be shared between different devices more and more frequently. To avoid a centralized trusted third party, i.e., key server, a solution based on secure computation is preferable and is described in this scenario.
- **Mobile Data Sharing:** This scenario provides privacy-preserving data sharing between different mobile devices and users through the cloud. Data are stored in the cloud only encrypted (i.e., not inspectable by the cloud service provider) but still sharable between users, even if the users have their data stored on different cloud storage providers.

Chapter 2

Model Description

This document uses the tabular representation of application scenarios as already used in deliverable D12.1 of the project PRACTICE. Besides the general information about each application scenario specified in D12.1, the table includes the adversary, trust, communication and system models. Furthermore it lists the guarantees required in every application scenario, in order to achieve the desired security objectives. The models and guarantees are specified for every individual application scenario separately. The template used is presented below in Table 2.1 and structured as follows:

Scenario: Name of the scenario																												
Summary: A short description of the scenario																												
Scenario Illustration: 	Participants: The participating parties and their roles. <ul style="list-style-type: none"> • $P1$: < Party 1 > – \mathcal{I} (e.g.) • $P2$: < Party 2 > – \mathcal{IC} • $P3$: < Party 3 > – \mathcal{R} • ... 																											
Communication Channels: The communication channels between the participating parties. <ul style="list-style-type: none"> • $P1 \leftrightarrow P2$ (Description, if required) • $P2 \leftrightarrow P1$ (Description, if required) • $P3 \rightarrow P1$ (Description, if required) • ... 	Adversary Model: <table border="1"> <thead> <tr> <th colspan="2">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)		Party (collectively)		trusted	semi-honest	covert	malicious	honest majority	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>
Party (indiv. entities)		Party (collectively)																										
trusted	semi-honest	covert	malicious	honest majority																								
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																							
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>																							
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>																							
Trust Model: P1 malicious P2 covert <ul style="list-style-type: none"> • Impact of using trusted hardware • ... 																												
Communication Model: <ul style="list-style-type: none"> • Which parties (e.g. server(s)) need to be online/reachable all the time? • Do parties need to respond immediately or within time limits? • What happens if no answer is received within the specified time limit? • ... 																												
System Model: <ul style="list-style-type: none"> • Examples for architectural constraints are: Execution time (CPU), latency, bandwidth, synchronization • ... 																												
Required Guarantees: <ul style="list-style-type: none"> • A list of security goals for the participating parties. • ... 																												
Workpackage References: WP <xx.x>	Literature References: [R1][R2][R3]																											

Each scenario is motivated and explained in a short summary. Further, each scenario is illustrated by a figure that shows the interaction and communication behaviour between the different participants of the scenario. The participants are also separately listed to show their assigned roles in the Secure Multi-party Computation (SMC) model. The different roles are introduced in the following sections. Furthermore, the security, privacy and verification goals are described informally. To analyze the capabilities of the adversaries, an attacker model is defined for every participant. The attacker model is further elaborated in the next sections. Moreover, the technical requirements, e.g., hardware and network limitations, are listed. Finally, references to related work packages and literature are given, if available.

The technical notations and definitions that are used for each scenario are described in the remainder of this section. We begin by introducing the different roles of parties participating in an application scenario. Afterwards we provide descriptions for the security models and guarantees being discussed in this document.

2.1 Participant Roles

The participants in each scenario can be assigned with a role. In the article [16], Bogdanov et al. introduce three fundamental roles to describe an SMC system—the input party \mathcal{I} , the computation party \mathcal{C} and the result party \mathcal{R} . Input parties collect and send data to the SMC system. The SMC system itself is hosted by computation parties who carry out the SMC protocols on the inputs and send results to result parties (in response of queries).

For the scenarios description in this deliverable we use the following notation. Let $\mathcal{I}^k = (\mathcal{I}_1, \dots, \mathcal{I}_k)$ be the list of input parties, $\mathcal{C}^m = (\mathcal{C}_1, \dots, \mathcal{C}_m)$ be the list of computing parties and $\mathcal{R}^n = (\mathcal{R}_1, \dots, \mathcal{R}_n)$ be the list of result parties.

A special kind of result party is the external verifier, denoted \mathcal{V} , who is not active in the system at the time of the computation. Instead, it later receives a transcript of the computation, typically sent by a result party or published on a public location. Although both result parties and external verifiers learn the result of a computation, there is an important difference in the trust assumptions they make. Whereas result parties receive a result whose correctness depends on trust assumptions on the protocol parties (e.g. the computation parties), an external verifier needs to be able to verify the result long after the computation, regardless of who performed it. Hence, the correctness of its results is required to hold even in a setting where all computation parties behave maliciously.

Real world parties can have more than one of these roles assigned to them. Thus, Bogdanov et al. argue that all deployments of SMC can be expressed using combinations of the above three roles. See Table 1 of [16] for examples of typical SMC deployment models inspired by published research on SMC applications.

In the following, \mathcal{ICR} refers to a party that fills all three roles, similarly, \mathcal{IC} refers to a party with roles \mathcal{I} and \mathcal{C} . We use superscripts ($k, m, n \geq 1$) to denote that there are several parties with the same role combination in the system.

In conclusion, the three roles (or combinations of them) are sufficient to describe each participant in the SMC model. Because of this, all participants in SMC-based scenario descriptions are annotated with a subset of the three fundamental roles $\mathcal{I}, \mathcal{C}, \mathcal{R}$.

2.2 Adversary Model

Different applications require different levels of security and thus different adversary models can be assumed for the underlying protocols so that the required security level for each application scenario

is met. The adversary model should help to identify an adequate level of security, because a higher security level usually has negative impacts on the efficiency. Hence, the capabilities of realistic adversarial participants are captured by the adversary model.

In the setting of SMC, multiple parties with private inputs wish to jointly compute a function of their inputs. Informally speaking, the security requirements of such a computation are that nothing is learned from the protocol other than the output (privacy), the output is distributed according to the prescribed functionality (correctness), and parties cannot make their inputs depend on other parties' inputs [10].

The security requirements in the setting of multi-party computation must hold even when some of the participating parties misbehave. Cryptographic tools have been proven to withstand strong adversarial behavior. However, the computational performance of computations crucially depends on the adversaries' strength. Therefore, an analysis of the attacker model is of importance when describing an application scenario.

Aumann and Lindell [10] distinguish three adversary models that are used to describe the attacker model in each scenario:

- *Malicious adversaries* are adversaries that may behave arbitrarily and are not bound in any way to follow the instructions of the specified protocol. Protocols that are secure in the malicious model provide a very strong security guarantee for the user.
- *Covert adversaries* have the property that they may deviate arbitrarily from the protocol specification in an attempt to cheat, but do not wish to be “caught” doing so. Protocols secure in the covert model guarantee that an adversary is caught cheating with at least a defined probability ϵ .
- *Semi-honest adversaries* correctly follow the specified protocol, yet they may attempt to learn additional information by analysing the transcript of messages received during the execution. Security in the presence of semi-honest adversaries provides a weaker security guarantee, yet might already be sufficient if the adversary is given limited access to the computation, e.g. through defined interface to framework executed in isolation (like trusted hardware).

We also annotate some parties as trusted parties which do not ‘attack’. A *Trusted Third Party (TTP)* is a party that is not in control of the honest party (user) but is assumed to behave according to the protocol specifications.

The adversary model *honest majority* refers to the case, where multiple participants (of the same type) are involved in a protocol and we assume that most of them (the majority) act in a benign way. This means that for a particular participant there is no guarantee that he is not malicious. However, we assume that the number of honest participants is larger than the number of malicious participants.

2.3 Trust Model

Depending on the base problem of a scenario and the solution approach chosen in the scenario, different levels of trust in the participating components and parties can be assumed. The trust model identifies trusted parties and components and the degree of trust one can place in them. A party is called trusted, if it behaves exactly as requested by the protocol. It is important to note that sometimes there are some *implicit* trust assumptions. For example, we implicitly trust the Certificate Authorities when using Public Key Infrastructure. Such implicit trust assumptions must be considered in the trust model as well.

A special case which should be considered in trust model is the use of *trusted hardware*, which can increase both security and efficiency. Using cryptographic functionalities included in dedicated hardware devices such as smartcards and Hardware Security Modules (HSM) or integrated into complex hardware components like processors can help to amplify operations performance or relax

assumptions. The trust model should also specify the assumptions made on the use of trusted hardware; for instance, it should declare which operations of a trusted hardware are used for a particular problem and what is the cost and performance gain of using those operations.

2.3.1 Trusted Hardware

Smartcards are dedicated hardware devices which can be used to protect data and code from unauthorized access. The data and code inside a smartcard is protected by the hardware-based protection of the smartcard from the rest of the system (i.e., it is dedicated hardware). Usually a smartcard is tamper resistant, protecting its data and code also from physical attacks on the smartcard. The chip inside a smartcard enables secure implementation of cryptographic protocols and algorithms. A smartcard can also be used for storing encryption keys. However smartcards can be vulnerable to side channel attacks such as power and timing attacks. The main limitation of smartcards is their low computational performance.

A **Hardware Security Module (HSM)** is a device which is plugged into a (network) computer or attached to it externally in order to provide a tamper-resistant environment for performing secure cryptographic processing. Digital keys can be securely stored, managed and processed within a hardware security module. HSMs are used in a variety of applications which perform operations on cryptographic keys including generation, storage and management of keys, en/decryption, digital signing and authentication services.

Since HSMs are required for several use cases, mostly for compliance reasons, providing HSMs in the cloud is of concern to the cloud service providers (CSP). Different CSPs already provide access to HSMs in the cloud or plan to do so in the near future, e.g., Amazon already offers a service called CloudHSM¹, and Microsoft announce a service called Azure Key Vault² which will provide HSM access within their clouds.

A **Trusted Platform Module (TPM)** [49] is a hardware security module which nowadays is present on most enterprise platforms. It provides a random number generator, asymmetric key generator, SHA-1 hash algorithm and secure storage amongst other capabilities.

One of the most common use cases for a TPM is to capture the configuration of a platform and store this information inside the TPM. The platform configuration is represented by cryptographic hash digest calculated for every software component loaded. The hash digests are stored in the TPM in register dedicated for this purpose, called Platform Configuration Registers (PCRs). These PCRs can only be manipulated by an *extend* operation. This means whenever a new value is stored in a PCR the previous value of the PCR is concatenated with the new value and the cryptographic hash of both is the new value of the PCR. By this, all values ever stored before in the PCR are represented by the current value. The PCRs are only reset when the system is restarted. This means, the TPM provides a write only log of the software loaded on the system.

The TPM further provides means to digitally sign the PCRs values as a report for attesting the software configuration of the system. An external verifier can verify the origin of the attestation report by the signature of the TPM. The verifier who got a detailed log of all loaded software along with the attestation report can recompute the PCRs values and verify that all software is indeed included in the log. This gives the verifier certainty about the configuration of the platform.

Modern CPUs include a set of instructions to enable dynamic root of trust for measurement, e.g., **Intel Trusted Execution Technology (TXT)** [36] and **AMD Secure Virtual Machine (SVM)** [7]. This allows to minimize the dependence of software on other software executed before and/or with higher privileges. DRTM allows to reset the state of a CPU at runtime before execution of a certain software.

¹<http://aws.amazon.com/cloudhsm/>

²<http://azure.microsoft.com/blog/2015/01/08/azure-is-now-bigger-faster-more-open-and-more-secure/>

This has the effect that all possible influences of software which executed before is nullified. With the CPU reset also a set of *dynamic*³ PCRs is reset and the hash of the executing code is extended to one of these PCRs. Hence, the software configuration of a platform consists only of the software executed after the CPU reset. Therefore, the attestation report and log sent to a verifier is reduced which allows a more efficient and more meaningful verification of the system configuration.

Texas Instrument's M-Shield [11] and **ARM TrustZone** [6] provide architectures for mobile devices which allow isolated code execution. To execute code in isolation the processor can be switched into a *secure mode* in which only trusted code is loaded. Software executed in the *non-secure mode* like the operating system or the user's applications cannot influence the state of the secure mode and do not need to be trusted. To ensure the integrity of the code in the secure mode it is loaded protected by *secure boot*. Hence, the secure mode executed only a small, integrity-checked code in isolation from the rest of the system.

The most recently announced secure execution technology is Intel's **Software Guard Extensions (SGX)**. As explained in detail in [22], SGX consists of a set of instructions and mechanisms for memory accesses control. These extensions will be included in future *Intel Architecture* (IA) processors, i.e., they will be available in most processors used by cloud service providers. These extensions allow an application to instantiate a protected container, called an *enclave*. An enclave is a protected area in the application's address space, which provides confidentiality and integrity even in the presence of privileged malware or a malicious system administrator. Noteworthy, SGX is designed such that an enclave is protected from all other software on the system, including the operating system and the hypervisor, and by this also from the administrators operating those. Hence, SGX allows the execution of enclaves which are inaccessible to the cloud service provider.

The SGX-enabled processors are not available on the market to date. However details on SGX technology are already published in Intel's whitepapers [40, 8, 33] and the *Software Guard Extensions Programming Reference*.⁴

2.4 Communication Model

Understanding the real-world communication model used in different scenarios is necessary to ensure that solutions meet the underlying requirements. The communication model should first of all specify the parties that communicate with each other in an application scenario. For any communication channel between parties, we then declare the assumptions made for the channel, including both explicit and implicit assumptions.

The assumptions to be made on the communication might depend on the application scenario. The following list gives a *non-complete* selection of assumption which need to be considered for the communication models of the individual application scenarios: Capabilities of the communication parties and channels. The requirements may differ for the different parties/channels, e.g., the cloud service provider is expected to be always online. Hence, it the need to be captured separately for each party/channel. The capabilities for the communication channels include, for instance:

- The requirement for *simultaneously* communication of all participants or a subset of participants.
- The requirement a party to be *always online*, i.e., reachable.
- Parties might need to respond within certain time limits or a delay might have some unwanted effects.
- The protocols might relay on the network layer for providing transactional communication. A violation of this property could potentially leak information or break the system.

³only present on v1.2 and higher TPMs

⁴<https://software.intel.com/sites/default/files/329298-001.pdf>

- The before mentioned capabilities are of special interest when some of the participants are expected to have limited capabilities, e.g., because they use mobile phones with unreliable network reception. The communication model also needs to specify assumptions that are made on the communication channels with focus on security properties.
- The authenticity of the sender and/or receiver can be relevant for the correct working of a protocol (authenticity of communication).
- Manipulations during the transmission of messages between participants can lead to unwanted effects (integrity of communication).
- When the communication is not encrypted during transport an eavesdropper on a communication channel might obtain information which he should have access to (confidentiality of communication).
- If the freshness of a message is not guaranteed, e.g., by the use of nonces, replay attacks can be used to attack the system (freshness).

2.5 System Model

The system model reflects the capabilities and properties of the parties participating in the application scenario regarding today's heterogeneous computation landscape. The model considers system benchmarks such as computational power, amount of memory, network connection properties, parallelism of computation, reuse of services, etc.

The following list elaborates on these capabilities/properties which are studied for the application scenarios in this document.

- Computational power and system memory requirements should express if there are special requirements for a scenario. In general every system benefits from higher computational power or more memory available, i.e., the system can operate faster. However, in some cases, e.g., when responses need to be provided within constrained time frames, a minimum of computational power or memory must be available to meet the deadlines. Furthermore, special cases such as smartphones or laptop computers should be considered. They are limited in their computational power but additional also in their energy, i.e., they are running on battery.
- With regard to network connection the properties such as bandwidth, latency and network reliability should be considered. One important special case are connections over mobile networks which might not be sufficient in their reliability, latency or bandwidth in a specific scenario.
- The parallelism of computations is a special variant of computational power. However, not every computation can benefit from massive parallelism provided in distributed systems or by cloud computing.⁵ Using cloud services may have positive effects such as speed-up and negative effects such as loss of control/trust. Both positive and negative impacts must be addressed in the system model. In particular it is important to evaluate, what an attacker can achieve by leveraging the massive computational power of the cloud. For instance, if the input provider stores its input in the cloud (e.g., Dropbox) the input data might not be trustworthy any more under the assumption that the cloud provider (i.e., Dropbox) is malicious.
- Concurrent usage of services can have different effects. For instance, for scaling a solution multiple instance of the same protocol/service can be used on the same device (e.g., server). However, sharing resources like a server might lead to security risks, for instance, by reusing keys.

⁵Computations might be adapted to benefit from parallelism but this usually requires significant implementation effort.

2.6 Guarantees

In this section we provide a list of security goals for the participating parties which must be guaranteed. This list includes, but is not limited to, the following:

- **Correctness and Verifiability**

The correct execution of the function/algorithm in question is required, i.e., it must produce the correct output. This seems to be an apparent goal in every application. However, it may not be achieved easily when some parts of the computations are performed in the cloud. Therefore, this requirement should be explicitly mentioned for the different application scenarios.

In [30] Genarro et al. introduce the notion of *Verifiable Computation*, which enables a computationally weak client to outsource the computation of a function on various dynamically-chosen inputs to one or more workers. The workers return the result of the function evaluation as well as a proof that the computation of the function was carried out correctly on the given value. Considering the cloud as worker, it should be specified if the computations must be verifiable and if yes, what kind of verifiability is required: *public (universal) verifiability* or *designated verifiability*. In the public verifiable scheme, any verifier can perform the verification, while in designated scheme, only a chosen verifier with specific characteristics, such as the capability of authenticated interaction, is able to perform the verification.

Implicit assumptions in the context of verifiability should be specified as well, e.g., the assumption that a majority of participants are honest.

- **In-distinguishability of inputs/parties (anonymity of participants)**

In some cases the inputs of a function and/or the parties participating in an application scenario require to be indistinguishable from one another, so that their anonymity can be preserved. Therefore, it should be declared if in-distinguishability and anonymity of inputs and participants are a requirement to be guaranteed in any of the application scenarios.

Chapter 3

Integration of Application Scenarios in the Project Architecture

This work package identified and evaluated several application scenarios. The motivation for considering a broad range of scenarios and capturing their (security) requirements is based on multiple reasons.

These reasons are

- These miscellaneous application scenarios show real-world and practical use of secure computation.
- Providing several use cases from various fields attract the attention of end users by showing the applicability of the project efforts and results in practical applications.
- Most scenarios are cloud computing based which is in line with the goal of the project. Considering several scenarios highlights the role of the cloud service provider and the need for thinking of solutions for privacy-preserving computation in the cloud.
- The evaluation of multiple scenarios shows that the final PRACTICE framework will not be limited to the two scenarios which will be implemented as part of the project. Rather the PRACTICE solution will be able to cover a wide range of applications.

In this chapter a connection between application scenarios and the overall architecture of the project is presented. For each application scenario from this work package it is shown how it fits into the PRACTICE architecture design.

Different tools and platforms for secure computation are developed in this project and are included in the overall architecture. Each platform consists of several components which are distributed over different layers of the overall architecture.

Our goal is to identify which components are required to implement every specific use case scenario. It is possible that one application can be implemented using several tools. In this case all candidate tools will be listed along with those components of each tool that need to be used to implement the application scenario.

3.1 Overall Architecture

The overall architecture of the project is depicted in Figure 3.1. This diagram shows the components of a *Platform as a Service* (PaaS) for performing secure computation in the cloud. It consists of several layers and incorporates a number of secure computation engines and tools, namely VIFF [3], SAP HANA [1], Compiled L1 [45, 46], Fresco¹, SCAPI [29], FairPlayMP [14], ABY [26] and Sharemind [2]. These tools use different technologies and provide multiple interfaces of different types with the end user. They also differ in their support of secure computation schemes (e.g.,

¹Fresco is a tool developed and used internally at the Alexandra Institute <http://alexandra.dk/>

two-party vs. multi-party) and the adversary models of participating parties. The goal of PRACTICE is to present a unified architecture which can support implementation of wide variety application scenarios in the context of secure computation in the cloud. The detailed features and capabilities of these engines and the whole architecture are described in the deliverable D21.2 of the project.

In this document a path is specified on the architecture for each application scenario, which can represent those components of the architecture that are required for the implementation(s) of an application. It represents how a wide variety of use case scenarios can be implemented in PRACTICE. Whereas no single tool is suitable for implementing all application scenarios, there is at least one tool in the architecture of the project which can be used to implement each of the presented scenarios. The tool may be used as it is or may need to be adapted with new protocols and combined with other tools and interfaces to realise the implementation of the applications.

3.2 Implementation of Application Scenarios: An Overview

Table 3.1 represents a mapping between application scenarios and secure computation tools and platforms which potentially can implement each scenario. The implementation paths on the architecture's big picture are highlighted in chapter 4 for every individual application scenario. In most cases at least one component from each layer of the architecture is used to have a complete path beginning from the end user and ending at the cloud infrastructure.

Application Scenario	Tools	Remarks
Aeroengine Fleet Management	Fresco	The current version of Fresco is not suitable for implementing this scenario, because it has only one computing party and Fresco only supports secure computation with two or more parties. However, this issue could be handled by implementing appropriate protocols in Fresco.
Platform for Auctions	Sharemind Fresco	Assuming that a number of the computing parties can behave maliciously, this application can be implemented on Sharemind using passively secure computation protocols along with SGX hardware for correctness guarantees, if the cloud service providers are trusted not to collude with each other and also other data providers.
Platform for Benchmarking	Sharemind HANA/SEED	The application could be implemented in Sharemind when computation servers are hosted by non-colluding cloud service providers and correctness is guaranteed by the use of trusted hardware (SGX).
Consortium Gathering Information From its Members	ABY Sharemind	ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two.

Tax Fraud Detection	ABY Sharemind Fresco	ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. Fresco can implement all parts of this scenario except correctness verification which is not currently supported by Fresco.
Joint Statistical Analysis Between State Entities	ABY Sharemind Fresco	ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. Fresco can implement all parts of this scenario except correctness verification which is not currently supported by Fresco.
Privacy Preserving Genome-Wide Association Studies Between Biobanks	Sharemind Fresco	Fresco can implement all parts of this scenario except correctness verification which is not currently supported by Fresco.
Privacy Preserving Personal Genome Analyses and Studies	Sharemind HANA/SEED Fresco	For implementation of this scenario in Fresco much work is required in the upper layers of the architecture to ensure against a corrupt lab.
Platform for Surveys on Sensitive Data	ABY Sharemind HANA/SEED Fresco	ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. This scenario can be implemented on Sharemind assuming honest majority and using secure hardware. This application could use techniques of SEED, but this would require the introduction of asymmetric cryptography schemes and even then things wouldn't be simplified a lot. This scenario could be implemented by Fresco in an instantiation with two computing parties, since currently Fresco supports only two-party secure computation with malicious security. However, the currently supported protocol (SPDZ) for this setting can easily be extended to multiple parties.
Location Sharing with Nearby Contacts	Sharemind	This scenario can be implemented on Sharemind assuming honest majority and using secure hardware.

Privacy Satellite Detection	Preserving Collision	Sharemind Fresco	<p>This scenario can be implemented on Sharemind assuming honest majority and using secure hardware.</p> <p>This scenario could be implemented by Fresco in an instantiation with two computing parties (number of hosts chosen among the satellite operators should be two), since Fresco currently only supports two-party secure computation with malicious security.</p>
Key Management		Sharemind	<p>This scenario can be implemented on Sharemind assuming the cloud service providers are non-colluding. However, since there are no actual secure computations taking place in this application, implementing it with Sharemind does not make much sense.</p>
Mobile Data Sharing		Sharemind	<p>This scenario can be implemented on Sharemind assuming the cloud service providers are non-colluding. However, since there are no actual secure computations taking place in this application, implementing it with Sharemind does not make much sense.</p>

Table 3.1: Candidate Tools for Implementation of Application Scenarios

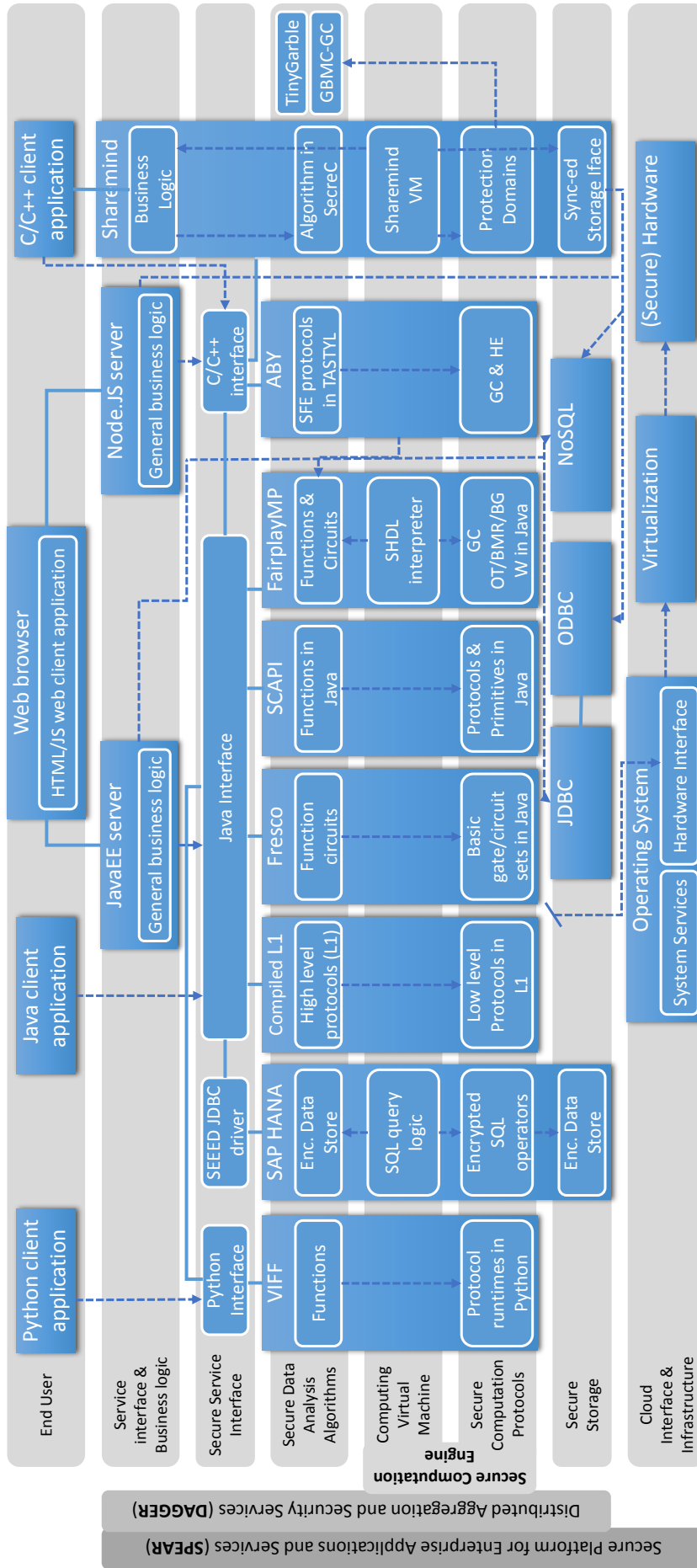


Figure 3.1: Overall Architecture

Chapter 4

Application Scenarios

In this chapter the individual application scenarios are presented and their adversaries, trust, communication and system models are described per scenario. The tabular representation in which the scenarios are presented serves to create an overview of each use case scenario and its models. It also enables the reader to compare the investigated application scenarios with one another.

The tabular representation of each scenario is followed by a figure of the overall architecture in which those tools and components of the architecture are highlighted which can be used to implement the scenario within the PRACTICE architecture. The candidate tools for implementing each application scenario are marked on the architecture with a distinguished colour to show the diversity of possible implementations. The involved components of the architecture are also coloured accordingly, so that the reader can identify which components are required for the implementation of a specific use case scenario using a particular tool. If a component is required in more than one tool, then it is marked with a combination of colours so that the reader can recognize its involvement in multiple implementations. Figure 4.1 shows the colours used for highlighting the components required for the implementation of application scenarios with one or several secure computation engine(s).









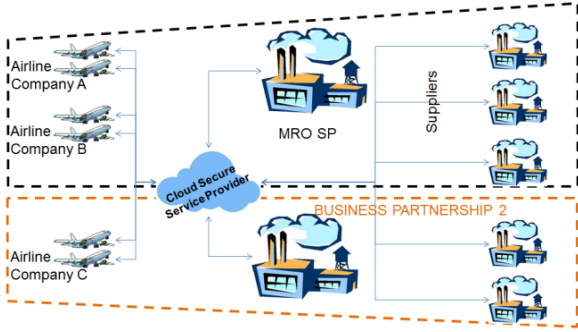
Secure Computation Tool(s)	Colour of Required Components
HANA/SEED	
Fresco	
ABY	
Sharemind	
HANA/SEED + Fresco	
Fresco + Sharemind	
ABY + Sharemind	
HANA/SEED + Fresco + Sharemind	

Figure 4.1: Colours Used For Different Tools

The application scenarios are grouped into four categories. First, *joint business applications* are described followed by *joint studies applications*. Next, different *location sharing applications* are evaluated and finally *end user applications* are described.

4.1 Joint Business Applications

<p>Scenario: Aeroengine Fleet Management</p>																																										
<p>Summary: The scenario aeroengine fleet management describes an online system enabling the optimization of the maintenance, repair, and overhaul (MRO) process for the engine sector of the aeronautic supply chain. Three parties are involved in this scenario: fleet owners (i.e., airlines or air forces), MRO service provider and suppliers. Each of the three parties provide inputs to optimize engine service works. Fleet owners provide their engine work load and status data, MRO service providers contribute their current work plan and inventory status, and the suppliers provide their production plans and inventory data. Given all data in encrypted form, the system can compute an optimal service plan for the engines. This involves computing of supply plans as well as delivery orders for the involved suppliers. Moreover, spontaneous changes in the supply plans, e.g., production delays, are fed in to the calculations to update the plans on demand.</p>																																										
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Cloud Service Provider C^l • $P2$: Airline Companies $\mathcal{I}^k \mathcal{R}^k$ • $P3$: MRO Service Provider $\mathcal{I}^m \mathcal{R}^m$ • $P4$: Suppliers $\mathcal{I}^n \mathcal{R}^n$ 																																									
<p>Communication Channels:</p> <ul style="list-style-type: none"> • Channel1: $P1 \leftrightarrow P2$ • Channel2: $P1 \leftrightarrow P3$ • Channel3: $P1 \leftrightarrow P4$ 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> <th></th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (indiv. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	honest majority		$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>
Party (indiv. entities)	Party (indiv. entities)				Party (collectively)																																					
	trusted	semi-honest	covert	malicious	honest majority																																					
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																				
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>																																				
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>																																				

Trust Model:

- The different actors in this scenario generally do not trust each other. All actors could gain advantages from accessing the confidential information of other actors. Hence, every actor might try to violate the privacy of one or all other actors.
- The cloud service provider (CSP) is generally not trusted, but it is assumed that the majority of CSP is not colluding. The CSP might be identical with one of the actors (i.e., the MRO service provider) or collaborate with one of the actors. Consequently, the CSP should not be able to reconstruct any secret information.

Communication Model:

- The participants do not need to be online for communication.
- There is no need for simultaneous communication in sending input to the cloud service provider from distinct input entities (P2, P3 and P4). However P1 must provide the computations results to P2, P3 and P4 at the same time.
- P1 must be reachable and responsive all the time.
- Transactional communication is required since incomplete input data can lead to erroneous results.
- Input and output data are provided in encrypted form, therefore, transport encryption is not required for preserving confidentiality.
- All parties must be authenticated before exchanging data.

System Model:

- Due to the large size of data being communicated a high bandwidth is required for all actors.
- The cloud service provider must have high computational power and large memory capacities in order to be able to process the huge amount of input data and produce output in an acceptable amount of time.
- This application does not require a highly reliable network connection because communication may be performed offline.
- The complexity and time-consuming nature of computations in this application scenario limits the size of groups of cooperating input and result parties which can be handled in practice.

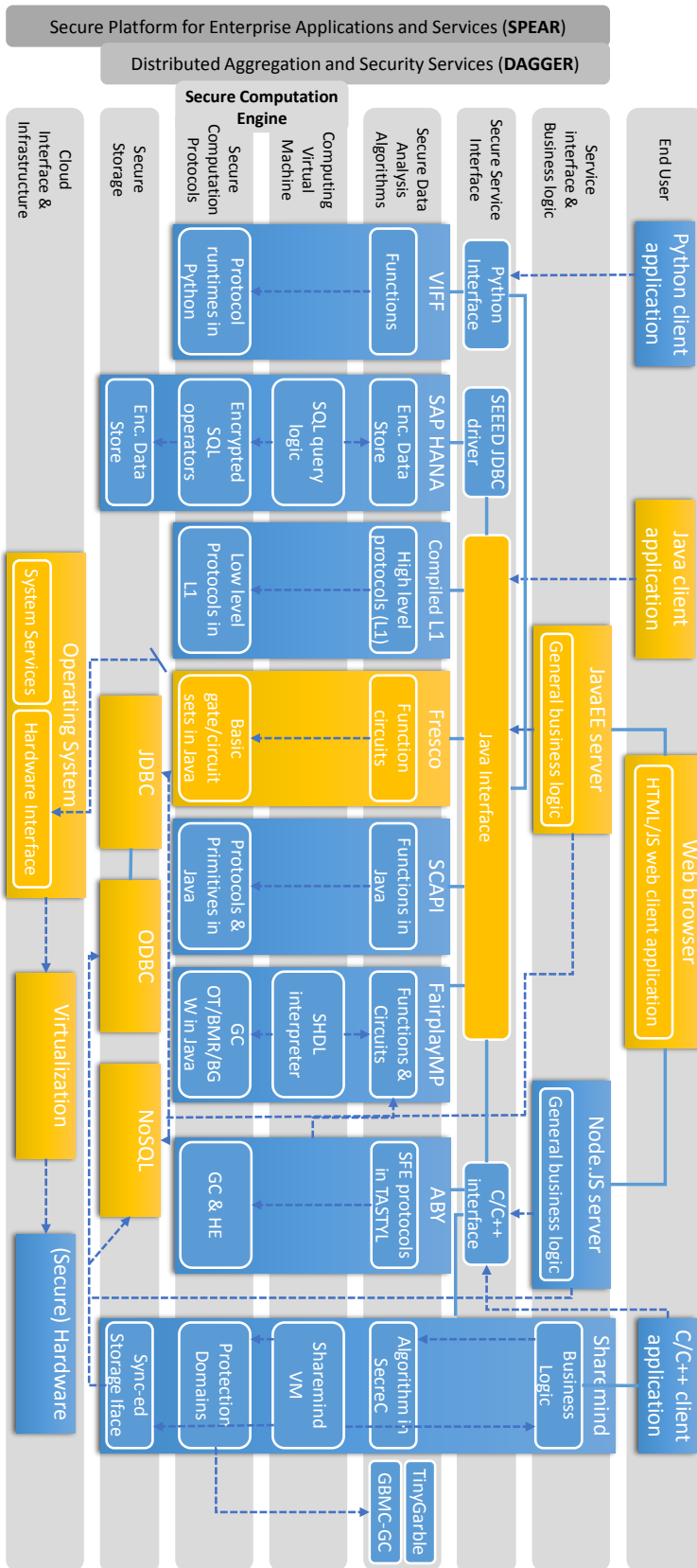
Required Guarantees:

- The system must guarantee correctness of the results. However, neither universal nor designated verifiability is required.
- A majority of participants behave honestly.
- The number of participants in the market is limited and are known by all participants. Therefore, indistinguishability is not a relevant requirement.
- The participants input data may not be decrypted by the cloud service provider, neither by any other participants.
- Computed results can only be decrypted by the designated receivers.

Workpackage References: WP 24.1, 24.2 and 24.3

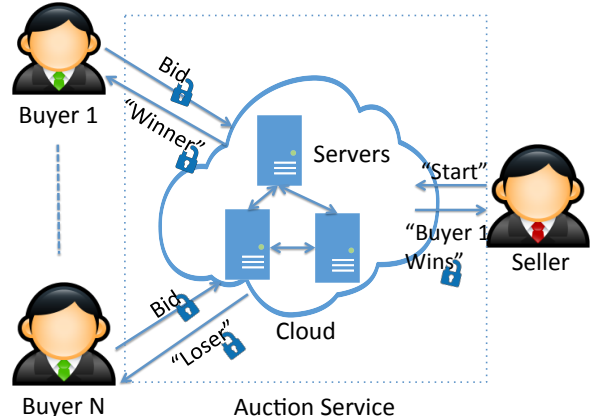
Literature References: [39]

Implementation within PRACTICE overall architecture:



Remarks:

- This scenario can potentially be implemented using Fresco, if new protocols are added to it. The scenario has only one computing party. Currently Fresco supports only secure computation with two or more parties. Therefore, some protocols could be implemented in Fresco to handle this situation as well.
- Sharemind turns out to be a tool which can implement a wide range of application scenarios. However, since the computing party is a single entity in this scenario, this application cannot be implemented on Sharemind, as this tool only provides support for two- or multi-party secure computation.
- This application can be implemented using secure outsourcing methods such as Fully Homomorphic Encryption.

<p>Scenario: Platform for Auctions</p>																																																	
<p>Summary: Auctions are means to control the ways information is coordinated on a market and most auctions have elements of sealed bidding. Apart from the submitted bids, confidential data may also concern private like information describing the commodities or services traded, e.g., a consumption profile in procurement of electricity. As such, auctions may be interlinked with secure statistics. Secure multi-party computation is used commercially for handling confidential bids in some of the most common types of auctions, the double auction known from most exchanges for financial as well as physical commodities and the classical first price sealed bid auctions used in many procurement scenarios.</p>																																																	
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Cloud service providers \mathcal{C}^k • $P2$: Auction service provider \mathcal{C} • $P3$: Buyers one or more – $\mathcal{I}^n \mathcal{R}^m \mathcal{V}^n$ • $P4$: Sellers one or more – $\mathcal{I}^n \mathcal{R}^n \mathcal{V}^n$ • $P5$: Competition regulator – \mathcal{V} 																																																
<p>Communication Channels:</p> <ul style="list-style-type: none"> • Channel1: $P1 \leftrightarrow P2$ • Channel2: $P1 \leftrightarrow P3$ • Channel3: $P1 \leftrightarrow P4$ • Channel4: $P2 \leftrightarrow P3$ • Channel5: $P2 \leftrightarrow P4$ • Channel6: $P2 \leftrightarrow P5$ 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (individ. entities)</th> <th colspan="4">Party (individ. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> <th></th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P5$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P5$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (individ. entities)	Party (individ. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	honest majority		$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>	$P5$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P5$	<input type="checkbox"/>
Party (individ. entities)	Party (individ. entities)				Party (collectively)																																												
	trusted	semi-honest	covert	malicious	honest majority																																												
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																											
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>																																											
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>																																											
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>																																											
$P5$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P5$	<input type="checkbox"/>																																											

Trust Model:

- All participants (except the competition regulator VV) might have an interest in cheating in the auction. Therefore, they are considered untrusted.
- The secure multi-party computation is carried out by the cloud service providers, hence, it has to be assumed that the majority of them is trusted.
- For the authentic and private exchange of data between the participants there is a need for managing and distributing keys among them. If this is done with a public key infrastructure (PKI) the certification authorities of the PKI need to be trusted.

Communication Model:

- P1 and P2 need to be online, reachable and responsive all the time.
- Each auction has a due date, up to this due date buyers can submit their bids. Simultaneous communication for sending input is not required. After the due date, P2 should immediately finish the auction round and results must be declared all the other parties immediately after.
- If no input is provided by a participant in set P3, then this participant automatically loses auction.
- Input should be transmitted in encrypted form for privacy of the data.

System Model:

- Execution time to evaluate the prices and determine the winner of the auction is important for the acceptance of the system.
- Synchronization between P1 and P2 must be satisfied.
- Bandwidth is not critical because shared data is not huge.

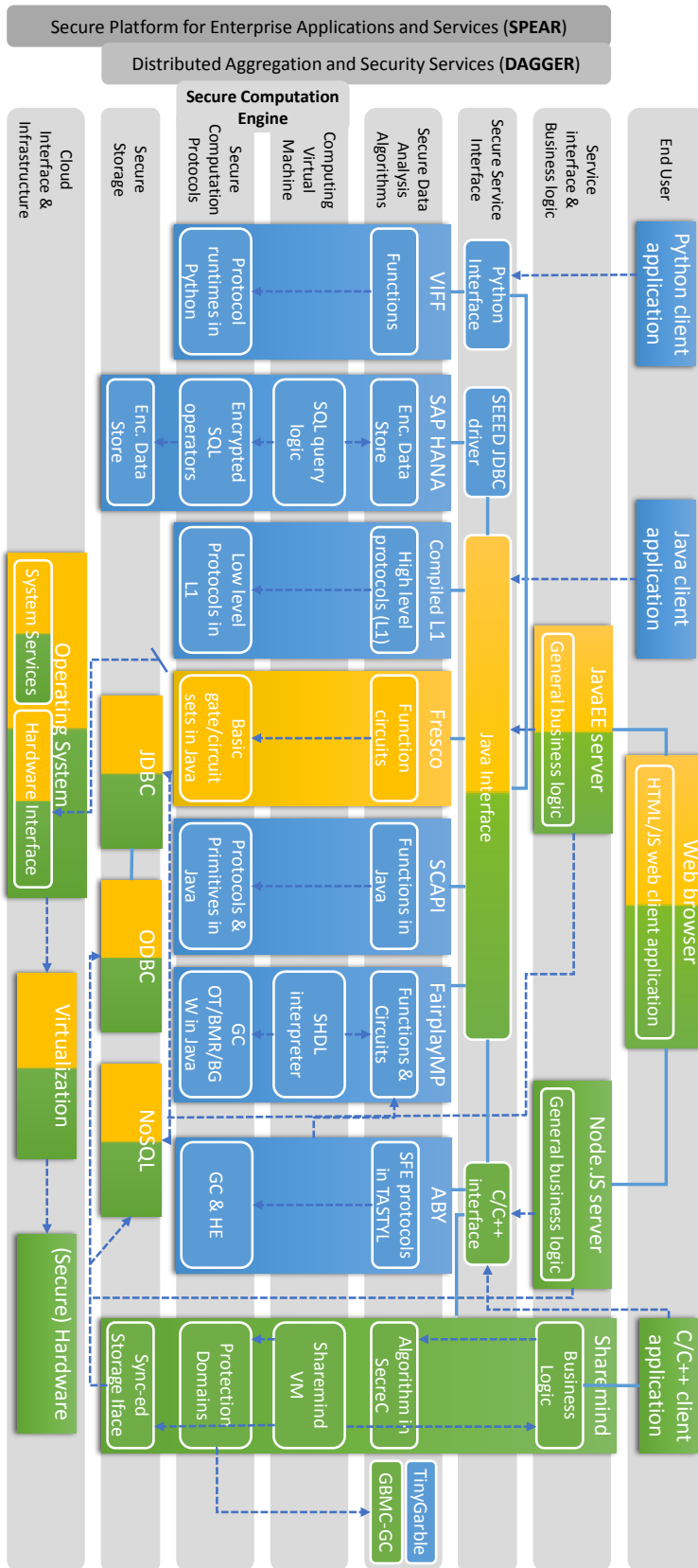
Required Guarantees:

- The inputs of the sellers and buyers must stay confidential.
- The result of the auction must be correct, i.e., the buyer with the highest bid wins.
- The correct functioning of the system must be verifiable by the Competition regulator.

Workpackage References: W24 (to some extent)

Literature References: [19]

Implementation within PRACTICE overall architecture:



Remarks:

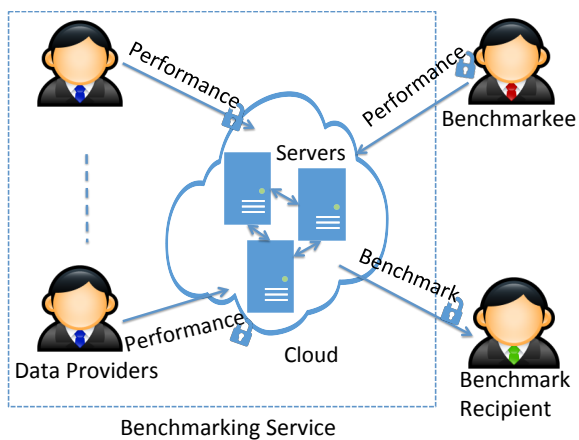
- Semi-honest secure multi-party computation is supported currently by Fresco. Therefore, this system could be implemented easily.
- Assuming that a number of the computing parties can behave maliciously, this application can be implemented on Sharemind using passively secure computation protocols along with SGX hardware for correctness guarantees, if the cloud service providers are trusted not to collude with each other and also other data providers.

Scenario: Platform for Benchmarking

Summary:

Benchmarking, understood as relative performance evaluation of "alternatives" (typically decision making units), is widely used to generate insight, planning as well as motivation. Keeping private information that describes the decision making units is typically critical. One exemplary deployment is the benchmarking of commercial bank customers. Here, benchmarking economic efficiency of the commercial customers can function as a complement to traditional credit rating. The value-added may e.g. come from a richer data foundation (which may also be used for credit rating) and/or the possibility to explore how exposed a given bank is. This solution requires a third party to confidentially handle information and no natural third party institution exists.

Scenario Illustration:



Participants:

- $P1$: Cloud service providers C^k
- $P2$: Benchmarkee (entity to be benchmarked) \mathcal{I}
- $P3$: Data providers (providing data to benchmark against) \mathcal{I}^n
- $P4$: Benchmark recipient \mathcal{R}

Communication Channels:

- $P1 \leftrightarrow P1$
- $P1 \leftrightarrow P2$
- $P1 \leftrightarrow P3$
- $P1 \leftrightarrow P4$

Adversary Model:

Party (indiv. entities)					Party (collectively)	
	trusted	semi-honest	covert	malicious		honest majority
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>

Trust Model:

- Economic efficiency of the commercial customer is very important across his competitors. Because, if his economic weakness is shown by his rivals, this data can be used to take advantage in the race of gaining more market share.
- Provided data may be used by P4 to increase its prices if the customers' economic efficiency is strong. Thus, P4 try to get directly whole data provided by P3.
- P1, P2 and P3 must declare and point out their privacy preserving methods, systems and policies.

Communication Model:

- P2 and P4 do not need to be online all the time, but other participants must.
- P1 and P3 must be reachable and responsive all the time.
- There is need for simultaneous communication between P1 and P3.
- Input data must be encrypted and output data cannot include direct financial data about the P2 but it only includes P2's economic efficiency grade.
- All parties must be authenticated before exchanging data.
- Communication between all parties must be well protected.

System Model:

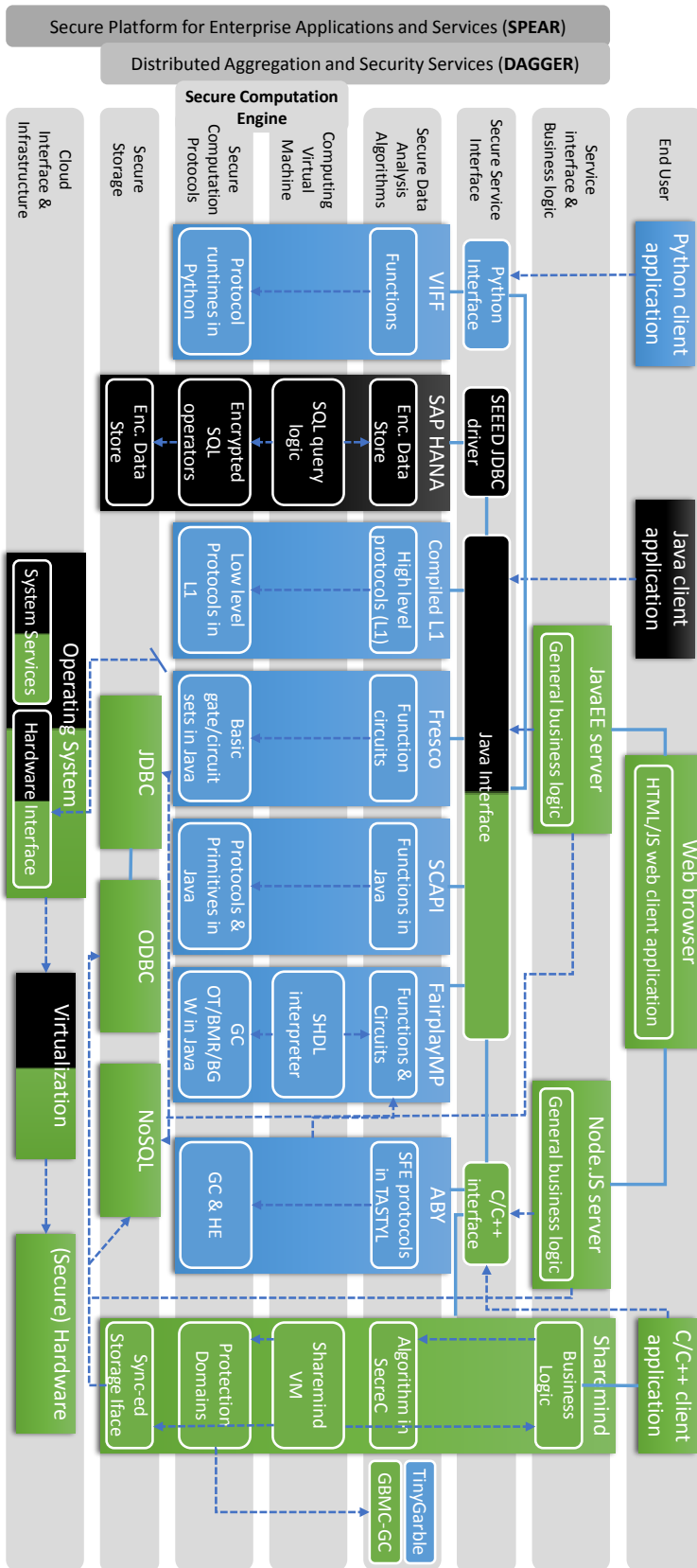
- Large amounts of data need to be shared in this scenario (e.g., analysis and reports). Hence, a high network bandwidth is required.
- System needs a highly reliable network because some of the participants must be online all the time.
- Execution time to evaluate the economic efficiency of P4 is important to take action.
- Synchronization between all parties should be satisfied, especially the synchronization between P1 and P3 is of importance.

Required Guarantees:

- P3 share correct and objective data about P2.
- Analysis and given grades can only be decrypted by P4 and never shared with any other party. On the other hand, input data cannot be decrypted by anyone under the assumption that the majority of CSP is honest.
- Any P3 member cannot see any other P3 member's data. Moreover, information that demonstrated which companies are analyzed should be kept secret.

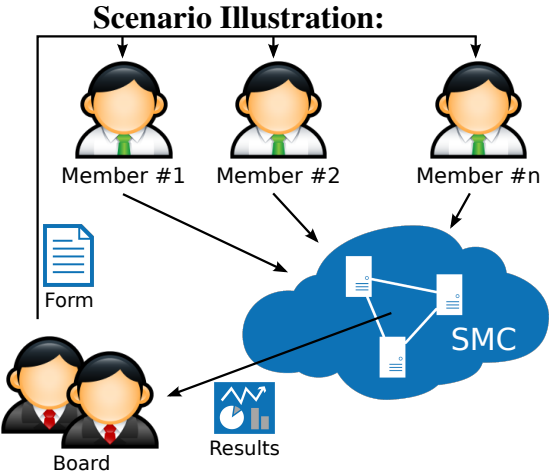
Workpackage References: WP23**Literature References:** [48]

Implementation within PRACTICE overall architecture:



Remarks:

- The application could be implemented in Sharemind when computation servers are hosted by non-colluding cloud service providers and correctness is guaranteed by the use of trusted hardware (SGX).
- SEED can be used to additionally protect data provided by the data providers while it is stored in the cloud. The key(s) must be provided to Benchmarkee only in order to be able to selectively access data for comparisons.

<p>Scenario: Consortium Gathering Information From its Members</p>																									
<p>Summary: Considering a scenario where a consortium would like gather information from its members, e.g., benchmark their joint economic results. However, consortium members might be competing companies and are, thus, reluctant to share that kind of information with the consortium board, which may consist of consortium members. Since, the consortium board should only be interested in aggregate results, a privacy breach can be alleviated by using SMC without losing functionality.</p>																									
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Consortium board – \mathcal{R}, \mathcal{V} • $P2$: Consortium members – $\mathcal{I}^n, \mathcal{V}^n$, where n is the total number of members • $P3$: Consortium members that execute the SMC for all members – $\mathcal{C}^k, k \leq n$ 																								
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P2 \rightarrow P3$ ($P2$ provide user credentials and inputs to the SMC service) • $P1 \rightarrow P3$ ($P1$ provides user credentials to access the system) • $P3 \rightarrow P1$ (SMC service provides aggregate results to $P1$) 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (collectively)				trusted	semi-honest	covert	malicious	$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Party (indiv. entities)	Party (collectively)																								
	trusted	semi-honest	covert	malicious																					
$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																					
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>																					
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>																					

Trust Model:

- P1: Consortium members might consider P1 more trustworthy if participation is not mandatory. (With voluntarily participation results will be partial, however, if a certain participation level is achieved results are still statistically valuable).
- P1: Consortium members might consider P1 more trustworthy if all of participants receive part of results (i.e., all entities of P2 will be also entities of P1), maybe personalized results (i.e., personal position in the aggregate results).
- P2: in the case results are private, trust is increased if smartcard access is required for them
- P3: is responsible for maintaining the privacy of the result data, e.g., P3 may not distribute the results.
- P1, P2 and P3: The use of trusted hardware can lead to relaxed assumption for the participants' adversary models, e.g., by giving guarantees on the correctness of the executed operations.

Communication Model:

- It is required that P1 and P2 are authenticated prior to any communication so that malicious external behaviour can be avoided.
- P2 is required to be online before the protocol execution to provide input, P1 is required to be online when the result is provided.
- If computation results are expected to be a huge quantity, specific bandwidth should be required to P1 (to download) to reduce communication time.
- P3 has to be always reachable (online) to facilitate user access.
- Time limitation (few weeks) for input providing is recommended, at the end of the time period computation starts (if a certain percentage of input were received results are statistically valuable).
- Communication between P1 and P3 must be protected, e.g., from modification of input data can lead to wrong results.
- In the case results are private, communication between P2 and P3 must be protected.

System Model:

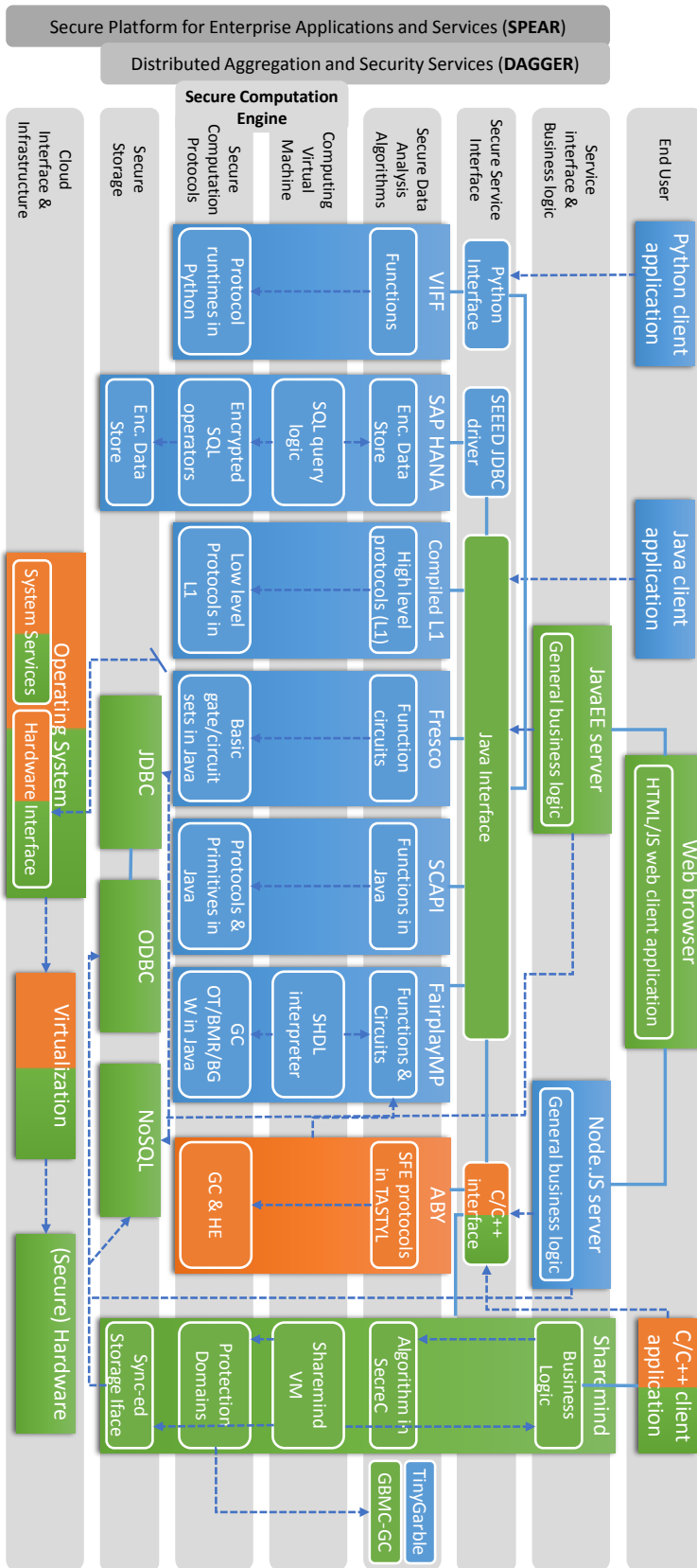
- Bandwidth is the a critical issue if input or output data are a huge amount which need to be transferred within tight time frames

Required Guarantees:

- Correct execution of the function/algorithm in order to produce correct outputs.
- Designed scheme so that a chosen verifier (with specific characteristics) can perform verification.
- Anonymity of input data is a mandatory requirement.

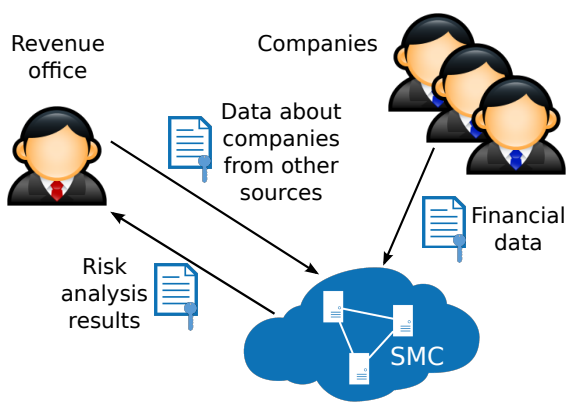
Workpackage References: WP22.1**Literature References:** [47, 18]

Implementation within PRACTICE overall architecture:



Remarks:

- ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. k trusted consortium members execute the secure computation for all consortium members, where $k \leq n$ with n being the total number of members. In the special case when $k = 2$, ABY can be used in this scenario. ABY assumes semi-honest computing parties, so the trusted consortium members can perform the computation.
- If we assume the covert model for consortium members executing SMC, the application can be implemented using Sharemind's passively secure protection domain and additionally allowing other consortium members to audit the computation servers or provide secure remote attestation of running software to the verifying consortium members. To protect against malicious adversaries, secure hardware can be used.

<p>Scenario: Tax Fraud Detection</p>																																								
<p>Summary: Detecting tax fraud is one of the cases where state entities, e.g., the revenue office, are interested in analyzing precise financial data of companies. However, such a risk analysis would require the creation of so-called super-databases, which might be prohibited by law. With the help of SMC, a precise analysis of cash flows can be executed that follows the law without the necessity to reveal the companies' sensitive financial data to the state entities. Moreover, the state entities can input data from other sources to improve the risk analysis results.</p>																																								
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Private companies – \mathcal{I}^n • $P2$: State cloud – \mathcal{C}^k • $P3$: Revenue office – \mathcal{ICR} • $P4$: Referee – \mathcal{V} 																																							
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P1 \leftrightarrow P2$ – Companies upload data to state cloud • $P3 \leftrightarrow P2$ – Revenue office inputs auxiliary data and performs risk analysis queries • $P2$ – State cloud servers communicate to perform risk analysis computations using SMC protocols • $P4 \leftrightarrow P2$ – Referee verifies computations 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4"></th> <th rowspan="2">Party (collectively)</th> <th rowspan="2">honest majority</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)					Party (collectively)	honest majority	trusted	semi-honest	covert	malicious	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>
Party (indiv. entities)					Party (collectively)	honest majority																																		
	trusted	semi-honest	covert	malicious																																				
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>																																		
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>																																		
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>																																		
$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>																																		

Trust Model:

- An implicit trust assumption is that all parties have certificates signed by a trusted authority to allow setting up secure authenticated communication channels.
- P1: The companies are naturally assumed malicious and can provide incorrect inputs. The goal of the risk analysis algorithms is to later find discrepancies in the companies' provided data as to detect tax fraud. Some initial consistency checks can be performed by comparing private aggregate statistics of the uploaded data to companies' public financial reports.
- P2: The state cloud servers are assumed to be non-colluding to guarantee the privacy of the data. This requires the cloud servers to be maintained by different organizations with clearly non-colluding relations to enhance the direct perception of security to the data owners (companies). A possible deployment model is the following. One server can be hosted by the revenue office itself or a similar state agency (e.g., the country's Department of Finance). Another server should be hosted by a non-government organization representing the interests of the private sector. For using multi-party SMC protocols, a suitable neutral party should host the third server, e.g., a country's data protection authority. Alternatively, the cloud servers could all be hosted by government agencies provided that authorized private sector representatives or designated referees are allowed to monitor and audit the server deployment and running software.
- P2: The state cloud servers are assumed to perform computations in a semi-honest manner. Semi-honest behaviour can be enforced through auditing the software and setting up strict physical and organizational security measures in addition to stating clear legal responsibilities for the hosting parties.
- P2: Using secure trusted hardware in the state cloud servers such as Intel's Trusted Execution Technology and SGX can greatly enhance the trust in the correctness of the performed computations since these technologies can provide attestation reports proving to an external verifier the executed software.
- P3: The revenue office is assumed to behave semi-honestly as its potential malicious activity is very limited. The types of risk analysis queries and the computations done with private data can be agreed on beforehand with private sector representatives to ensure that the outputs from the risk analysis do not harm the privacy of the companies' input data. A private sector representative hosting one of the cloud servers can also block excessive queries, negating the possibility of multiple-query attacks by the revenue office.
- P4: An external referee may be present and is assumed semi-honest to verify the correctness of the computations using SMC auditing techniques.

Communication Model:

- All communication between parties must use secure authenticated channels to avoid malicious external parties to interfere with the system. The state cloud servers and revenue office must use private channels so that an external malicious party cannot change the content of the exchanged messages and harm the integrity of the computations. Since companies will upload their data over the Internet, cryptographic methods to prove the integrity of the uploaded data should be used, e.g., message authentication code or digital signatures.
- The state cloud servers need to be on-line at all times to receive and aggregate companies' data and process risk analysis queries by the revenue office.
- Uploading companies' data can be done asynchronously. For companies with very large amounts of data, the company can receive a message later if the upload was unsuccessful or initial integrity checks for the data failed.
- Companies must upload their data before a fixed time in each tax declaration period, after which the revenue office can make risk analysis queries regarding that period's data. The companies' data are pre-processed and aggregated in a background process throughout the data upload phase.
- The risk analysis queries can be processed asynchronously if necessary. After the computations are complete, the results can be stored in secret-shared form in the state cloud and sent to the revenue office when it is ready to receive them.

System Model:

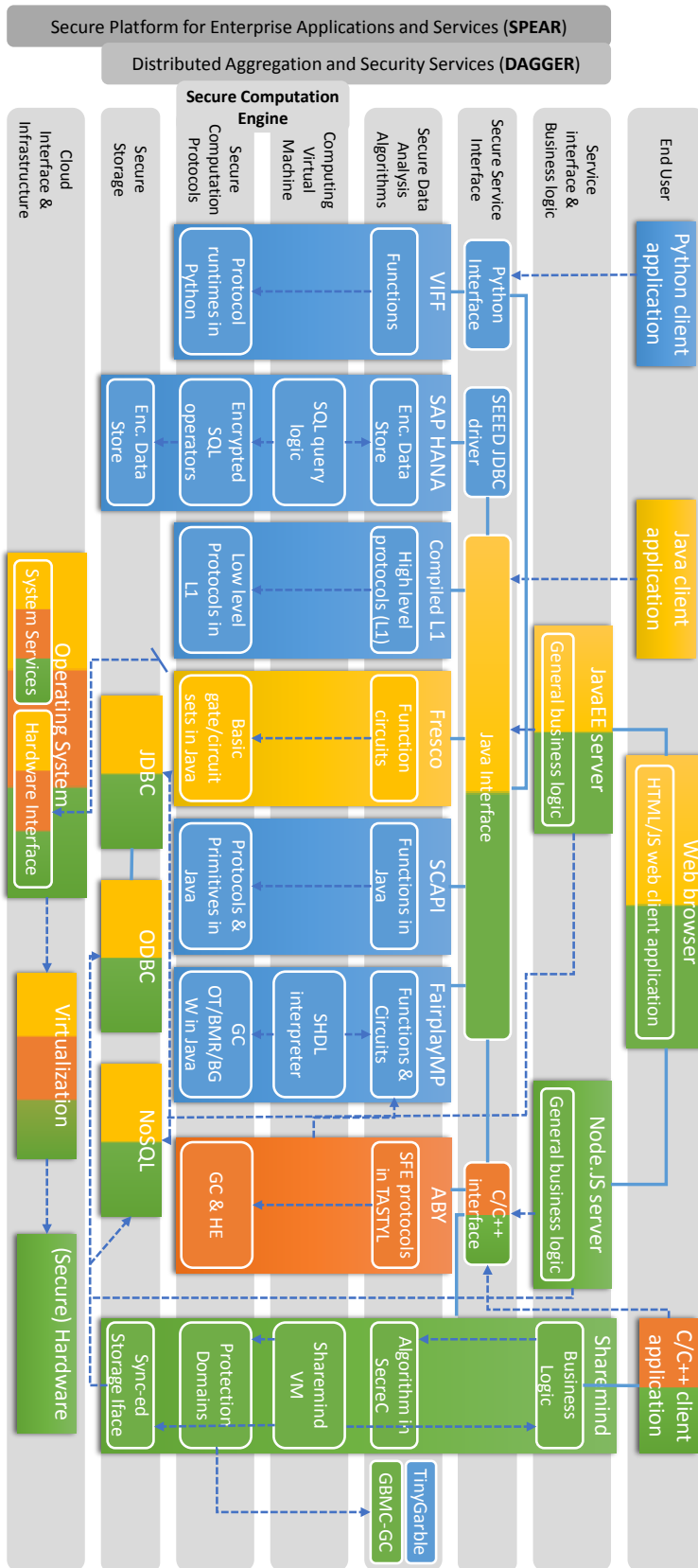
- Data collection should be performed via a web-interface provided by the revenue office.
- Data uploading must be fast even if many companies are uploading data simultaneously, since the deadlines for tax declarations are strict and it can be assumed that most of the uploads will be done in a few-day period. To allow fast data upload, the cloud servers must be equipped with powerful hardware and a large number of CPU cores to be able to handle load spikes and efficiently process and aggregate companies' data in parallel.
- Small- or medium-sized companies should get fast feedback if the upload or initial integrity checks for the data failed. For companies with the largest amount of financial data, a delay in the order of a few minutes is acceptable.
- The data aggregation must be performed seamlessly in parallel with data uploading to reduce the overall time consumption for performing risk analysis. For a taxing period of one month, the risk analysis process for a given period should not take more than 10 days to support periodical risk analysis using up to date data.
- The cloud servers must be connected with high-bandwidth network links to process all the companies' data and perform risk analysis in reasonable time as bandwidth is the bottleneck for most SMC protocols.

Required Guarantees:

- The revenue office cannot see any financial data of any of the companies.
- Only the revenue office can see the risk analysis results, which are calculated correctly and only list suspicious companies.
- The risk analysis results must maintain output privacy of the companies' data to the highest possible degree.
- Companies cannot see any data about other companies. Not even which companies are being analyzed.
- A referee or (independent) state entity may need to be involved as designated verifier for the risk analysis.

Workpackage References:**Literature References:** [15]

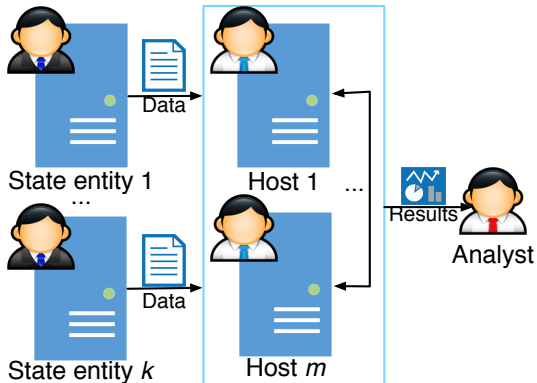
Implementation within PRACTICE overall architecture:



Remarks:

- ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. This scenario includes k semi-honest state clouds that perform secure computation for n companies and the semi-honest revenue office who upload their inputs to one of the state clouds. ABY can be used in the special case when $k = 2$.
- This scenario can be implemented using Sharemind's passively secure multi-party protection domain. It can additionally use secure hardware (Intel TXT and SGX technologies) to provide verifiability and enhance the perception of security to end users (data owners).
- Most of this system could be implemented in Fresco. Only the verification is not currently implemented. However, work is being done in PRACTICE in order to implement correctness verification on top of Fresco.

4.2 Joint Studies Applications

<p>Scenario: Joint Statistical Analysis Between State Entities</p>																																										
<p>Summary: Often, the public administration would like to have an overview of how one of its governance fields reflects on another, e.g. how does working during university studies influences the drop-out rate of university students. As the law forbids the compilation of a so-called <i>super-database</i> between the different state entities, the analysis can only be carried out by using pre-aggregated data or some other method. This, however, can reduce the quality of analysis results as more subtle nuances can be overlooked. In this scenario, the state entities combine their databases in a privacy preserving manner and allow a data analyst to perform pre-agreed queries.</p>																																										
<p>Scenario Illustration:</p>  <p>The diagram illustrates the data flow: State entity 1, ..., State entity k send Data to Host 1, ..., Host m. The Hosts then send Results to the Analyst.</p>	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Hosts – \mathcal{C}^m • $P2$: State entities – \mathcal{I}^k • $P3$: Data analyst(s) – \mathcal{R}, \mathcal{V} • $P4$: Referee – \mathcal{V} 																																									
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P2 \leftrightarrow P1$ – State entities upload data to hosts • $P1$ – Hosts perform computations using SMC protocols • $P1 \leftrightarrow P3$ – Data analyst receives result of computation. • $P4 \leftrightarrow P1$ – Referee verifies computations 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (individ. entities)</th> <th colspan="4"></th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> <th></th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (individ. entities)					Party (collectively)		trusted	semi-honest	covert	malicious	honest majority		$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>
Party (individ. entities)					Party (collectively)																																					
	trusted	semi-honest	covert	malicious	honest majority																																					
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																				
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>																																				
$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>																																				

Trust Model:

- As an implicit trust assumption, we assume the prior deployment of a PKI, which allows the public keys of all parties to be distributed in an authenticated way. This will allow the establishment of confidential and authenticated channels between all parties. We now describe the trust relations between the parties.
- P1: The servers (or hosts) performing the computations are independent from any other participants involved in the protocol (the state entities holding the data, the data analysts performing queries and the referee evaluating the computation). They are therefore assumed to be malicious. However, as mentioned in deliverable D12.1, it is plausible that in many scenarios the hosts can be controlled by state agencies and other organisations that can be assumed to be semi-honest or, in the worst case, covert adversaries who will be deterred by the possibility of malicious behaviour being detected and publicised. In this case, more efficient solutions can be obtained, as reported in [17]. In particular, one may consider a trust model where less than half of the hosts may be passively corrupted (“honest majority”), which will lead to significantly more efficient deployments.
- P1: The trust model described in the previous point can be modified in the future, if the hosts are equipped with trusted hardware, such as Intel’s Software Guard Extensions (SGX), which enables the execution of (integrity-checked) code in isolated environments. In this case, assurance as to the correctness of the computations can be built upon the assumption that the trusted hardware offers adequate protection.
- P2: The state entities are assumed to be semi-honest. In cases where state entities are also computing parties, multiple servers maintained by different entities with non-colluding relations will enhance the direct perception of security to the data owners and data privacy authorities. In particular, semi-honest behaviour can be physically enforced through software auditing and setup of organizational security measures, in addition to stating clear legal responsibilities for the hosting parties.
- P3: The data analyst is assumed to behave maliciously. In order to reduce data leakage, mechanisms that add appropriately chosen random noise to query responses can be employed to achieve differential privacy [28]. Alternatively, the analyst can be bound by contractual obligations to follow pre-specified guidelines regarding what queries are allowed by the system.
- P4: An external referee may be present and is assumed semi-honest to verify the correctness of the computations using SMC auditing techniques.

Communication Model:

- The assumption of an external PKI enables the use of standard cryptographic technology to establish secure authenticated channels between all parties, which will exclude the possibility of tampering with exchanged messages. In particular, the hosts and state entities can be assumed to communicate through secure confidential and authenticated channels.
- The setup stage of the protocol, consisting in the transmission of data from input parties, requires online presence of all hosts for a fixed period of time. The requirements for this stage are related to the amount of information contained in the collection of state entities’ databases, so the protocol should be prepared for high bandwidth usage.
- Uploading of data can be done asynchronously, i.e. state entities may upload their data at different times.
- Normal system operation involves hosts responding to queries from a data analyst, which means that state entities can be absent during the process.
- The referee can verify the correctness of computations asynchronously, and therefore is not required to be present during setup phase or query evaluation.

System Model:

- The number of state agencies participating in the analysis will typically be small, but each one of them will provide a large amount of data.
- Given the small number of input parties, it is unlikely that immediate parallel pre-processing or aggregation of the data as it is uploaded will be required, although this may happen for some specific analysis algorithms.
- The set-up period during which the input parties will be providing the data can usually be flexible enough to accommodate asynchronous uploading (e.g., one state entity at a time) so that the bandwidth and processing requirements during data upload are less demanding.
- The time-critical operation in this scenario will be providing timely answers to the data analyst.
- The hardware requirements for the hosts will be very much dependent on the nature of the analysis and data sizes. In the most typical cases, using today’s best SMC protocols, bandwidth will be a more important factor than CPU power, so the hosts should be interconnected with high-speed links.

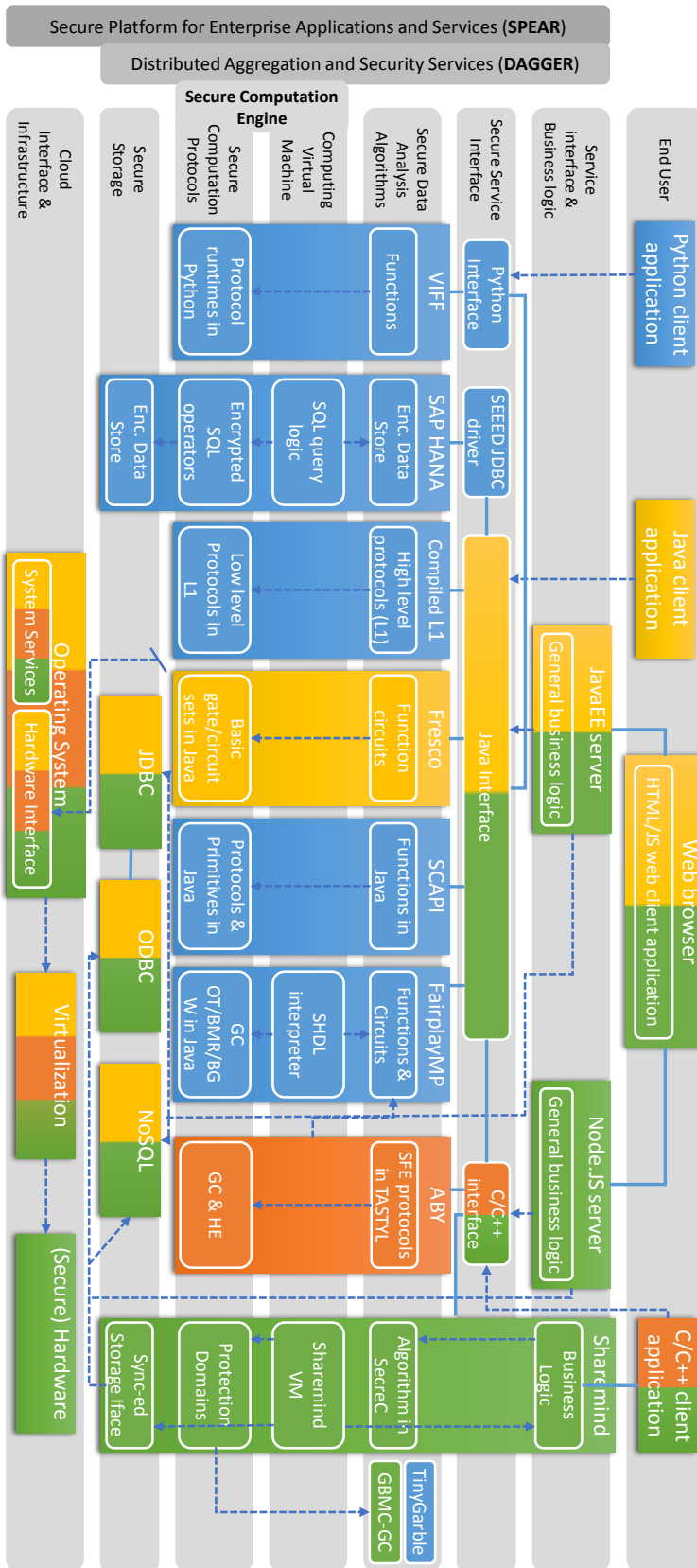
Required Guarantees:

- The state entities can only see their own data, and public analysis outputs.
- The hosts can only see public analysis outputs.
- The data analyst can only see the analysis outputs.
- The analysis outputs must be computed correctly, and must leak no more information than the pre-agreed set of queries allows.
- A referee or (independent) state entity may need to be involved as designated verifier for the analysis process.

Workpackage References: WP22, WP23
(as this scenario has common traits to secure statistics)

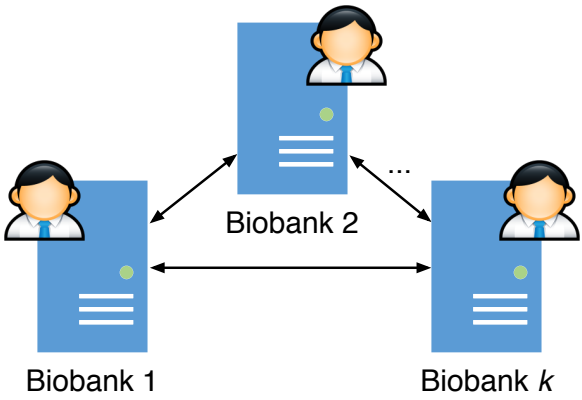
Literature References: [17, 28]

Implementation within PRACTICE overall architecture:



Remarks:

- ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. k state entities provide inputs to m hosts which then perform secure computation. It is mentioned in the description of the Trust model, that the hosts are generally assumed to be malicious but in some cases can be regarded as semi-honest. Thus, we assume them to be semi-honest and thus ABY is suitable for this scenario when $m = 2$.
- The Trust Model description describes that if suitable state agencies host the SMC servers, then they can be assumed non-colluding and semi-honest. In this setting, Sharemind is applicable using a passively secure protection domain. In the honest majority/covert model, secure hardware can be used to guarantee correctness of computations (assuming still that the hosts are non-colluding).
- Most of this system could be implemented in Fresco. Only the verification is not currently implemented. However, work is being done in PRACTICE in order to implement correctness verification on top of Fresco.

<p>Scenario: Privacy Preserving Genome-Wide Association Studies Between Biobanks</p>																						
<p>Summary: Biobanks from different countries wish to perform a joint genome-wide association study using each other's data. The biobanks already have collected the data along with signed consent forms from their donors. To collaborate, and, hence, get more accurate and interesting results, they want to share the data among each other without breaching the donors' privacy.</p>																						
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Biobanks – ICR^k • $P2$: Referee or state entity – \mathcal{V} 																					
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P1 \leftrightarrow P1$ 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th colspan="2">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)		Party (collectively)		trusted	semi-honest	covert	malicious	honest majority	$P1$	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>
Party (indiv. entities)		Party (collectively)																				
trusted	semi-honest	covert	malicious	honest majority																		
$P1$	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>																	
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																	
<p>Trust Model: Both for the trust model and for the adversary model we have a completely symmetric setup where all the biobanks are semi-honest.</p>																						
<p>Communication Model:</p> <ul style="list-style-type: none"> • All computation parties should always be online to provide the possibility to have an virtually instant query result while ensuring that the origin of individual records is not traceable). • The computation parties communicate over authenticated, secure, private channels. 																						

System Model:

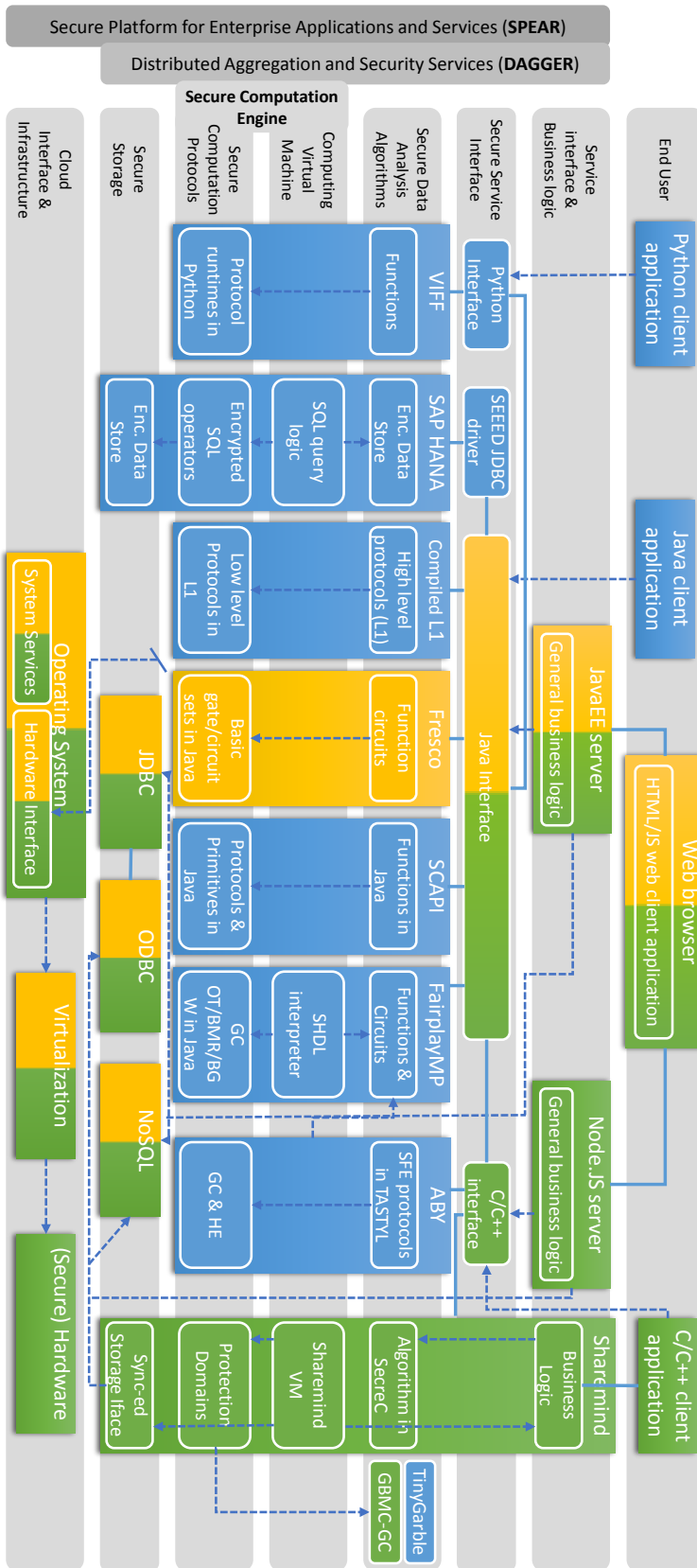
- Data analysts in biobanks interact with the service through an intuitive web-interface or through a command line tool that allows only previously agreed queries. The command line tool must resemble existing statistical analysis tools, such as GNU R, to be intuitive for the analysts.
- The biobanks must have the possibility to delete data of a donors if asked to do so by the donor.

Required Guarantees:

- The donors' input data cannot be accessed by other biobanks.
- The parties are not able to determine which records were input by which biobanks.
- The result of the analysis is guaranteed to be correctly computed.
- Query privacy is not required.
- A referee or state entity may need to be involved as designated verifier to guarantee correctness of the results on behalf of the general public.

Workpackage References: WP 22.1**Literature References:** [37]

Implementation within PRACTICE overall architecture:



Remarks:

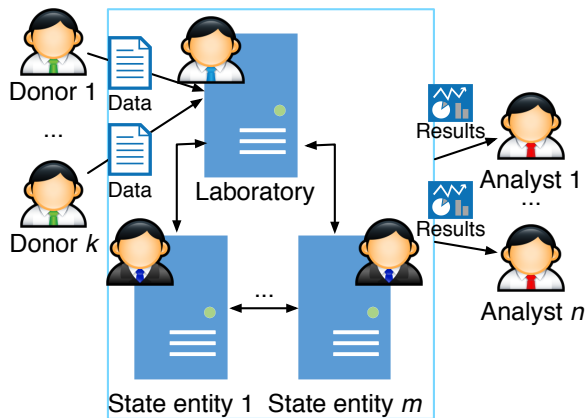
- This use case scenario can be implemented in Sharemind using a passively secure protection domain, since all parties are assumed to behave semi-honestly. No secure hardware is required.
- As semi-honest MPC is currently provided in Fresco this could be implemented. Only the verification would be missing, as Fresco does not provide this at the moment. However, work is being done in PRACTICE in order to implement correctness verification on top of Fresco.

Scenario: Privacy Preserving Personal Genome Analyses and Studies

Summary:

Donors can submit their DNA to a laboratory to receive feedback on genetic associations with specific illnesses and disorders. This genome data can then be added to a databases for further genome research. With the help of SMC, donors can also enter their phenotype information so that no involved organization sees their individual data, while analysts can still perform genome-wide association studies. Thus, in contrast to the 23andMe project, which is the largest genetic testing service provider, the sensitive phenotype data is protected. This can be realized by secret sharing the sensitive data between laboratories and multiple state entities. The probabilities of state entities colluding with the laboratories is much lower compared to laboratories colluding among each other which leads to a higher trust in the honest-majority assumption.

Scenario Illustration:



Participants:

- $P1$: Laboratory – \mathcal{C}
- $P2$: Donor(s) – \mathcal{I}^k
- $P3$: State entities – \mathcal{C}^m
- $P4$: Data analyst(s) – $\mathcal{R}^n, \mathcal{V}^n$
- $P5$: Referee or state entity – \mathcal{V}

Communication Channels:

- $P2 \rightarrow P1$
- $P1 \leftrightarrow P3$
- $P3 \leftrightarrow P3$
- $P1, P3 \rightarrow P4$

(With respect to communication among each other and communication to $P4$ the parties $P1$ and $P3$ behave indistinguishable.)

Adversary Model:

Party (individ. entities)					Party (collectively)	honest majority
	trusted	semi-honest	covert	malicious		
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>
$P5$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P5$	<input type="checkbox"/>

Trust Model:

- The state entities involved as computational parties are assumed to be semi-honest whereas the laboratories are considered to be malicious.

Communication Model:

- All computation parties should always be online. The donors can provide input to the system independently from each other at any time. The analysts can access the system at any time.
- The computation parties communicate over authenticated and private channels. The donors communicate with the laboratory (at least partially) non-electronically. The electronic communication (e.g. data about their phenotype) takes place over private and authenticated channels.
- The data analysts obtain the study results over authenticated and private channels.

System Model:

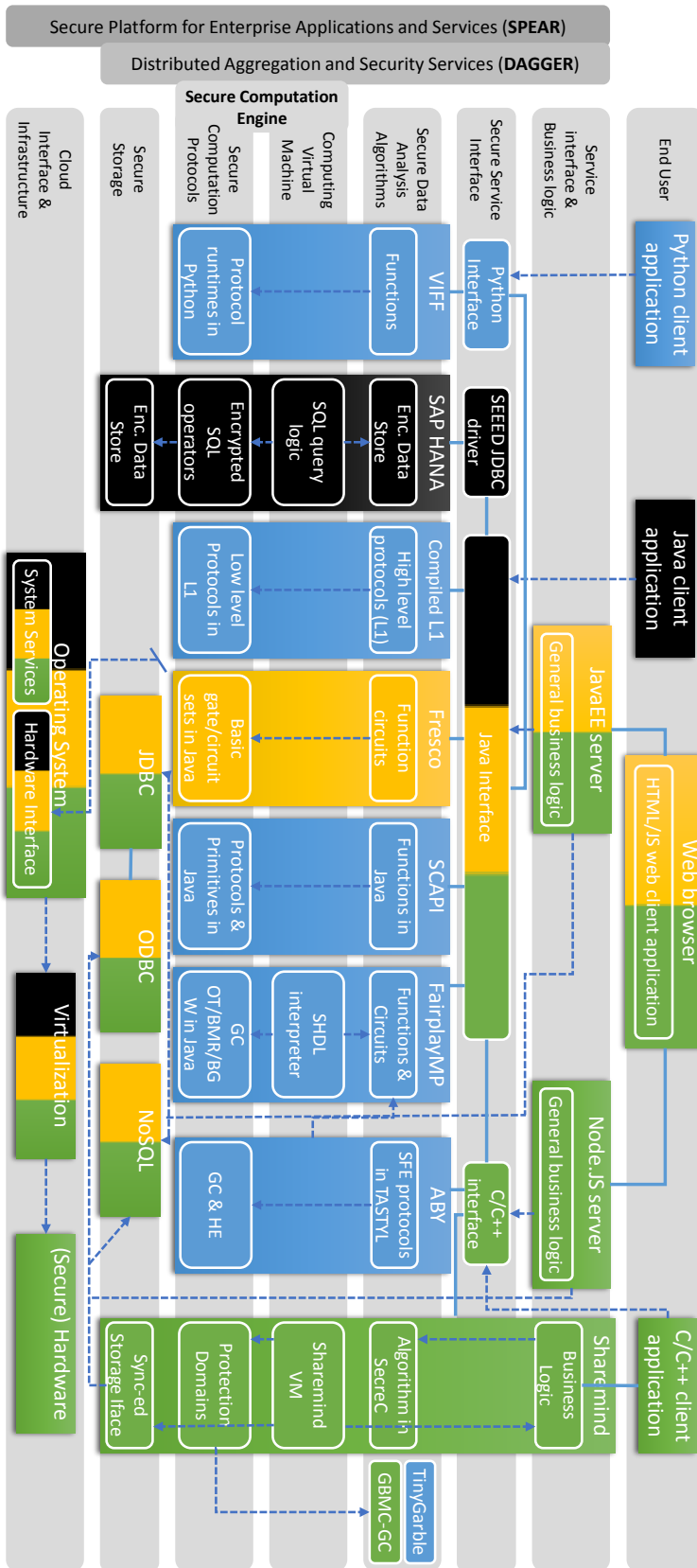
- The laboratory can host one of the servers, state entities can host others. Since the computational tasks are heavy even in the non-MPC world, both of them need strong computational power and low latency, high bandwidth communication channels between each other.
- On the other hand the opportunities of parallelization techniques can be leveraged since the computation of study results involves repeatedly applying similar operations independently on large amounts of data.
- Data donors and analysts interact through an intuitive web-interface provided by the computing hosts.
- The donors should be able to participate using commodity hardware, e.g. mobile phones, and their communication capabilities should be assumed to be limited to consumer internet connection/mobile data subscription. We may also assume the use of commodity hardware and a normal internet connection but not necessarily mobile phone/data support for the analysts.
- Data donors must have the option and possibility to delete their data from the system.

Required Guarantees:

- The donors' phenotype data cannot be decrypted by other parties.
- *Output privacy* is guaranteed for the analysis result to the highest possible degree.
- The survey result can only be accessed by the data analysts.
- Donors learn *nothing* about other donors.
- A referee or state entity may need to be involved as designated verifier to guarantee correctness of the results on behalf of the general public.

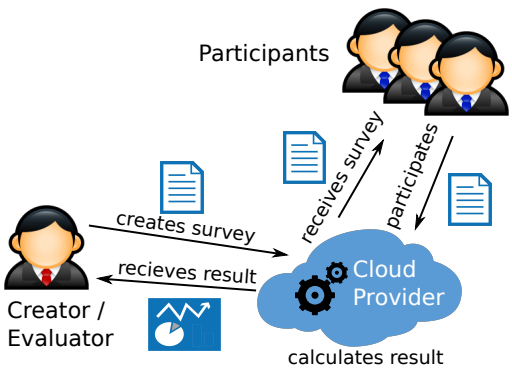
Workpackage References: WP 22.1**Literature References:** [5, 37]

Implementation within PRACTICE overall architecture:



Remarks:

- This application scenario can be implemented on Sharemind assuming honest majority and using secure hardware to protect against malicious adversaries.
- Additional phenotypical data can be stored (by P2/P1) in encrypted form in a cloud database (HANA). Data analysts (P4) can access this (encrypted) data in order to enhance their studies of the genome material, which by itself will require MPC methods.
- This use case scenario is likely implementable using Fresco for semi-honest MPC between the state entities. However, as the scenario is quite complex it would require much work in the upper layers of the architecture to ensure against a corrupt laboratory.

<p>Scenario: Platform for Surveys on Sensitive Data</p>																																										
<p>Summary: An online platform that allows users to design and run surveys while preserving the participants' privacy. Survey creators can create and upload their surveys to the platform that is also accessible to the participants. After participation a report generation system compiles a report based on the participants' encrypted data for the evaluator. The portal is designed to reduce leaks of private data.</p>																																										
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Cloud service provider – \mathcal{C}^m • $P2a$: Survey creator – \mathcal{I} • $P2b$: Survey evaluator – \mathcal{R} • $P3$: Survey participant(s) – \mathcal{I}^k <p>(One or more of the cloud service providers may be run by/on behalf of some other parties, e.g., the survey evaluator or the participants. In practice, the creator and evaluator may be the same entity.)</p>																																									
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P3 \rightarrow P1$ • $P2a \rightarrow P1$ • $P1 \rightarrow P2b$ • $P1 \leftrightarrow P1$ <p>(All parties need to communicate with each separate $P1$; in particular, all $P1$s communicate with each other.)</p>	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>honest majority</th> <th></th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2a$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2a$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2b$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2b$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (indiv. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	honest majority		$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>	$P2b$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2b$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>
Party (indiv. entities)	Party (indiv. entities)				Party (collectively)																																					
	trusted	semi-honest	covert	malicious	honest majority																																					
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																				
$P2a$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>																																				
$P2b$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2b$	<input type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>																																				
<p>Trust Model:</p> <ul style="list-style-type: none"> • We consider two deployments: one in which less than half of the cloud service providers may be passively corrupted (“honest majority”), and one in which all but one of the service providers may be actively corrupted (“malicious”). In D12.1, only protection against malicious $P1$ was considered; a semi-honest $P1$ may be preferable for performance reasons. • In deliverable D12.1, parties $P2a$, $P2b$, and $P3$ were considered to be possibly semi-honest. The present deployment offers protection even against malicious $P2a$, $P2b$, and $P3$. • Deliverable D12.1 also mentions the public as a possible result party. For efficiency reasons, the two deployments described here do not take this party into account. 																																										

Communication Model:

- All computation parties should always be on-line. The survey creator, evaluator, and participants can use the system independently from each other at any time.
- The computation parties communicate over authenticated and private channels. The participants communicate with the computation parties over private channels (with authentication if the survey is not open to everybody). The creator and evaluator communicate with the computation parties over authenticate and private channels.
- The participants need to communicate their answers while the survey is open. Otherwise, their answers are not taken into account. In the interest of privacy, if there are not enough participants, the survey result is not computed.
- The survey answers are communicated in one single transaction to prevent incomplete surveys from affecting the result

System Model:

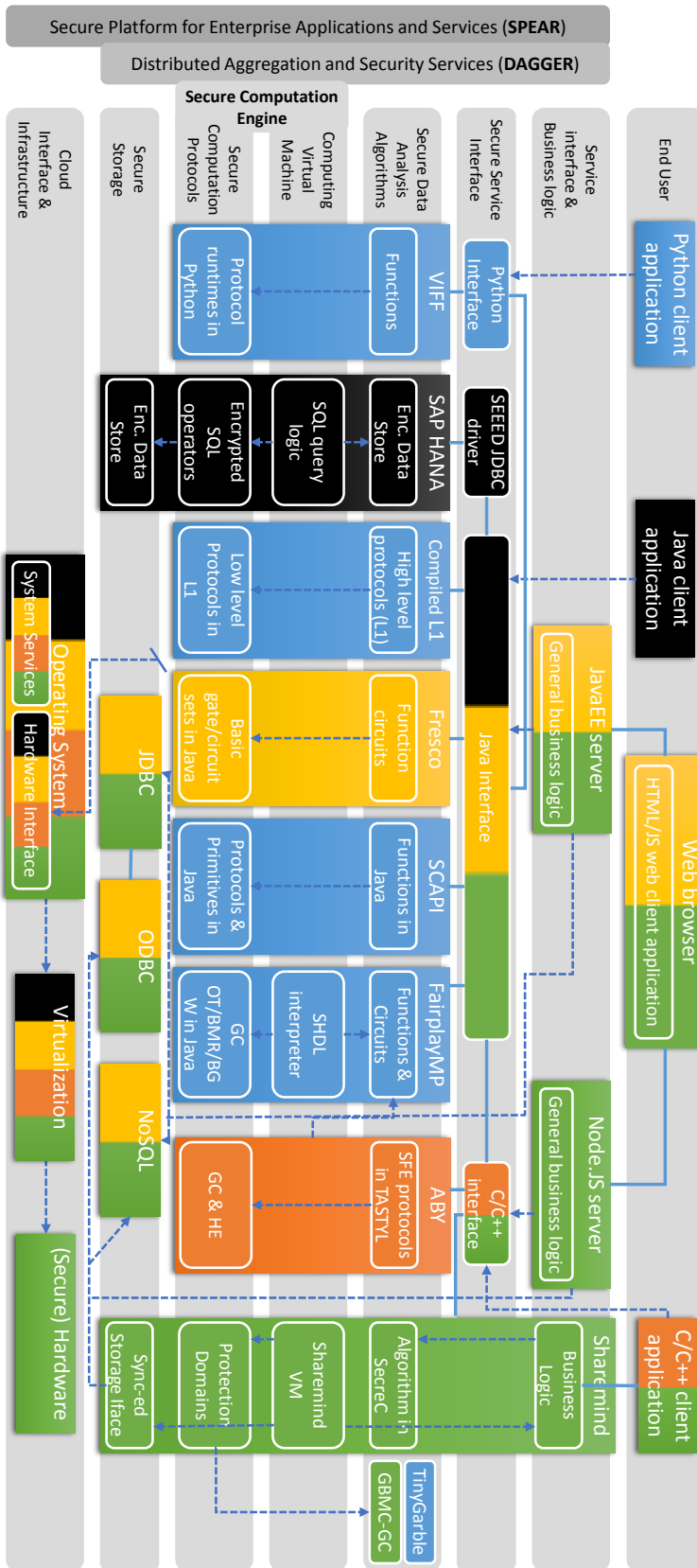
- The participants should be able to participate using commodity hardware, possibly even mobile phones. Similarly, their communication should be limited so they can participate using a consumer internet connection/mobile data subscription. For the survey creator and evaluator, the use of commodity hardware and a normal internet connection should also be assumed; mobile phone/data support is not needed.
- The cloud service providers need strong computational power and low latency, high bandwidth communication channels between each other.
- Since the computation of survey results involves repeatedly applying the same operation on large amounts of data, the cloud service providers should benefit from parallelism.

Required Guarantees:

- The participants' input data cannot be decrypted by the other participants, the cloud provider, or the survey creator/evaluator.
- The survey result can only be decrypted by the survey evaluator.
- The result of the survey is guaranteed to be correctly computed by the cloud service provider and computation engine.

Workpackage References: W23.1**Literature References:**

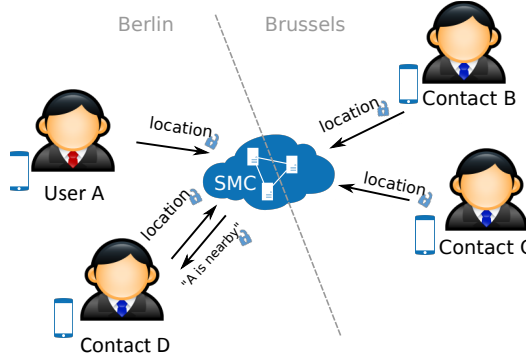
Implementation within PRACTICE overall architecture:



Remarks:

- ABY is suitable for applications that perform secure two-party computation. This scenario can be implemented by ABY, if the number of computing parties is two. In this scenario, k (malicious) participants would like to complete a secure survey using m cloud service providers that perform the secure computation. In the special case when $m = 2$ holds and the cloud service providers are assumed to be semi-honest (as described in the Trust model), ABY can be used for the secure two-party computation.
- This use case scenario can be implemented on Sharemind assuming honest majority and using secure hardware to protect against malicious adversaries.
- This application could use techniques of SEED, but this would require the introduction of asymmetric cryptography schemes and even then things wouldn't be simplified a lot.
- This application can be implemented with Fresco in an instantiation with two computing parties, since currently Fresco supports only two-party secure computation with malicious security. However, the currently supported protocol (SPDZ) for this setting can easily be extended to multiple parties.

4.3 Location Sharing Applications

<p>Scenario: Location Sharing with Nearby Contacts</p>															
<p>Summary: A smart phone app that lets users announce their location to nearby contacts, while not leaking location information to far away contacts. This is useful when users want to meet up with their contacts for various activities (dating, networking, etc.). As a user’s location can communicate a lot of private information (sexuality, religion, occupation, etc.), the users prefer to reveal their location only to relevant contacts, i.e., those nearby. The system protects the user’s privacy by computing proximity using SMC.</p>															
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • <i>PI</i>: Users (reveals her location; potentially learning other participants location) – <i>IRC^k</i> 														
<p>Communication Channels:</p> <ul style="list-style-type: none"> • <i>PI</i> ↔ <i>PI</i> <p>The communication is supposed to be peer-to-peer between the parties’ smartphones as implemented in [42]. A cloud server could act as a medium for the parties’ encrypted communication or alternatively the communication could be truly peer-to-peer.</p>	<p>Adversary Model:</p> <table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td><i>PI</i></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><i>PI</i></td> <td><input checked="" type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	<i>PI</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>PI</i>	<input checked="" type="checkbox"/>
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority									
<i>PI</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>PI</i>	<input checked="" type="checkbox"/>									
<p>Trust Model:</p> <ul style="list-style-type: none"> • All participants tend to cheat in order to find out the current location of their contacts without fulfilling the agreed requirement (physical proximity). • <i>P1</i> may attempt to find out if some of her contacts are near a specific location, by claiming that location to be her current location. • Running the proximity computations within a TEE such as ARM TrustZone is possible, but not recommended. The limited computing resources of a smartphone constrain this application. Use of a TEE introduces more resource limitations and thus makes the application non-responsive, as the number of users increases. Therefore, introducing more limitations by trusted hardware at the cost of drastic performance reduction is not a good option. 															

Communication Model:

- All participants must be online and reachable. As the location of a user is subject to continuous change, out of date location information is completely useless and online communication is required. Participants who are not reachable are left out of the protocol.
- Because the participants may change their locations every few minutes, results must be provided in a short time (few minutes). Otherwise the location information may not be valid any more.
- No high bandwidth is required for the communications between user's smartphone and the server, but reliable communication is required. However, this might be challenging given that the use case is focused on mobile devices.
- The communication does not need to be transactional.
- The freshness of the location information must be guaranteed.
- Integrity of communication must be preserved, as modified location information produces wrong and unreliable output.
- The communication between users must be authenticated, so that the participants know to whom they are sending their location information.

System Model:

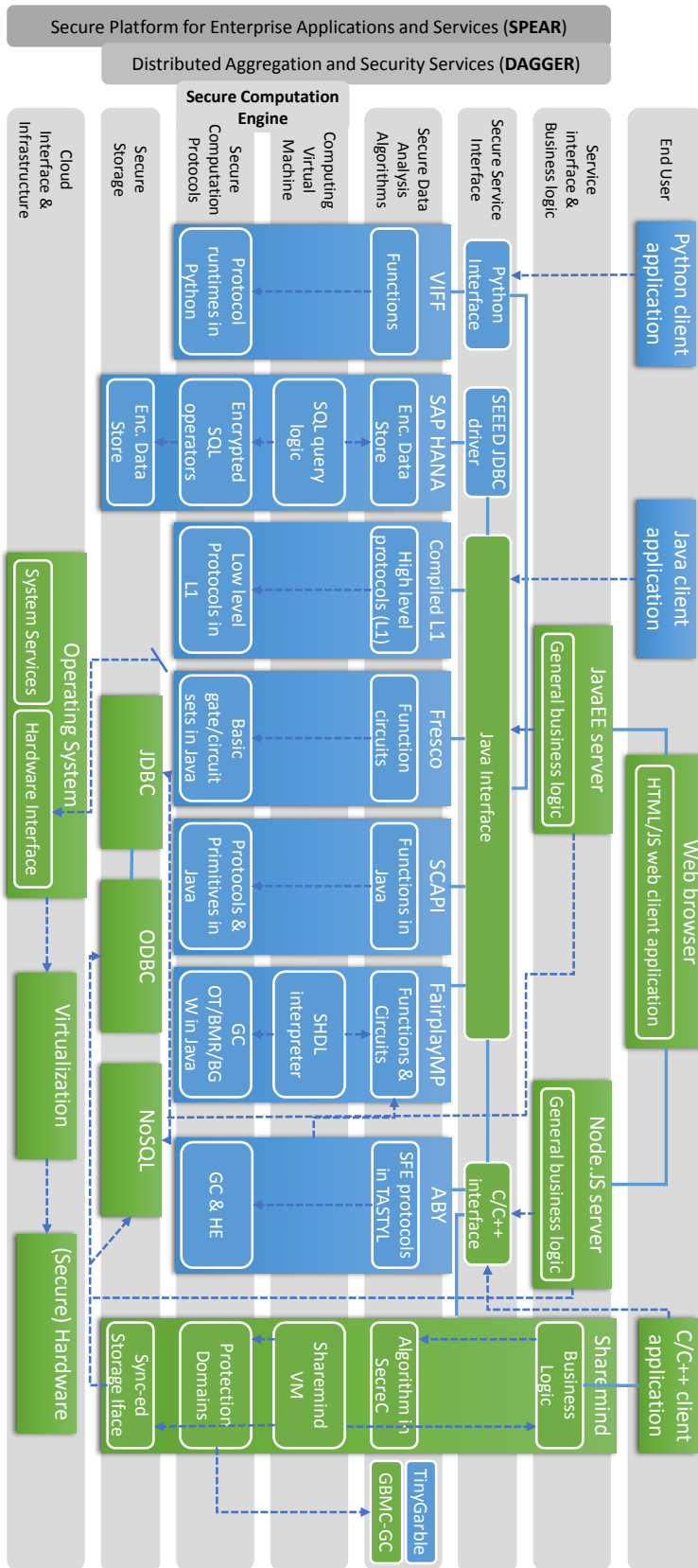
- Communication and computation is done on smartphones, with potentially quite limited resources.
- The computation should be quick for the app to appear responsive.
- The network connection must be reliable, so that users can communicate online. The communication channel is not required to provide very high bandwidth. Use of mobile network for communication can lead to the failure of the application, if users are located in places without (acceptable) network coverage, which is a common issue.
- Computational power, amount of memory and network connection play an important role for practicability of this application scenario. A shortage of each of these factors will result in the non-responsiveness of the application which will lead to bad user experience.

Required Guarantees:

- No location data should be revealed unless so intended (e.g., locations cannot be permanently broadcast, or registered at some third party).
- If the user decides to announce her location, only nearby contacts should learn her location.
- The user should not learn the location of her contacts unless they are also sharing their location and are nearby.

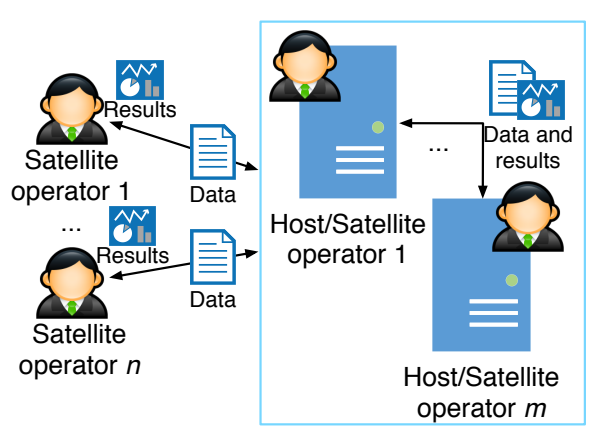
Workpackage References:**Literature References:** [42]

Implementation within PRACTICE overall architecture:



Remarks:

- This application scenario can be implemented in Sharemind using the actively secure two-party protection domain. Each pair of users would run the required computations separately. However, the attack of giving wrong input coordinates remains, since active security does not protect against false inputs. Some sort of central trusted third party would be needed to verify input coordinates of all parties.

<p>Scenario: Privacy Preserving Satellite Collision Detection</p>																															
<p>Summary: Different countries wish to detect collisions between their satellites without revealing the exact location and trajectory of their satellite.</p>																															
<p>Scenario Illustration:</p> 	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: Hosts chosen among the satellite operators – ICR^m • $P2$: Satellite operators – IR^n, \mathcal{V}^n • $P3$: Authority – \mathcal{V} 																														
<p>Communication Channels:</p> <ul style="list-style-type: none"> • $P1 \leftrightarrow P1$ • $P1 \leftrightarrow P2$ • $P1 \rightarrow P3$ 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (collectively)				trusted	semi-honest	covert	malicious	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>
Party (indiv. entities)	Party (collectively)																														
	trusted	semi-honest	covert	malicious																											
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>																									
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>																									
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>																									
<p>Trust Model:</p> <ul style="list-style-type: none"> • Satellite operators and host parties may be assumed to be malicious, since they could be interested in accessing data about the locations and the trajectories of the other satellites. • Authority is assumed to be semi-honest, since it should be a third-party international organization. 																															
<p>Communication Model:</p> <ul style="list-style-type: none"> • Hosting parties should be always online during the computation. • Satellite operators can submit data independently one from the other, i.e., asynchronously. • All parties communicate over authenticated, private channels. 																															

System Model:

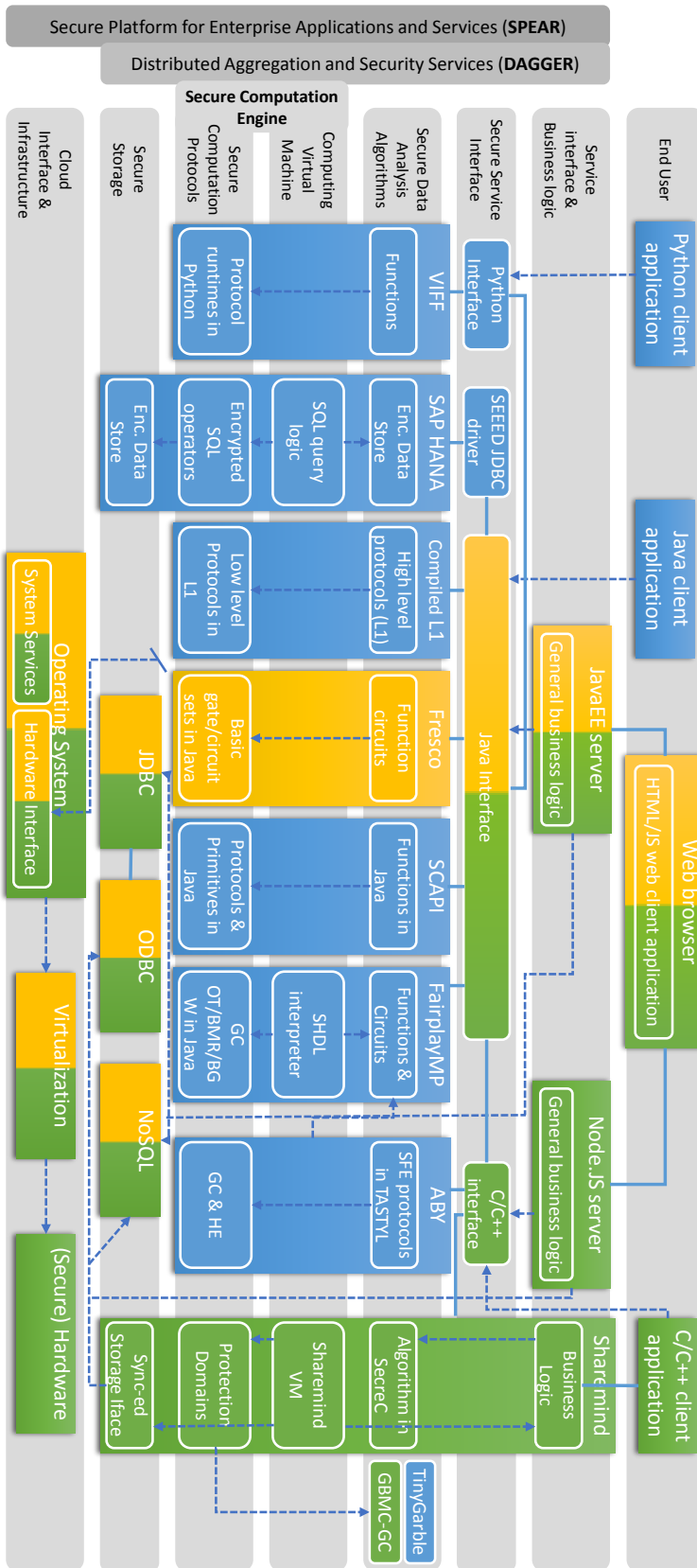
- Hosting parties are chosen among satellite operators and perform the secure computation over data collected from the other satellite operators.
- Satellite operators interact through an intuitive web-interface provided by the computing hosts. They can be requested to insert data about the location and the trajectories for computing the probability of collisions. They also get an alarm from the system if a collision is detected over a threshold of probability.
- If requested, authorities can inspect the provided data and the computations to verify the correctness.

Required Guarantees:

- Satellite operators and hosts learn nothing about satellite locations or trajectories.
- Only the collision probability is revealed if it exceeds a threshold.
- In case of a collision the operators involved may wish to prove correctness of protocol executions to a designated authority, relying on verifiable computation.

Workpackage References: WP 22.1**Literature References:** [38]

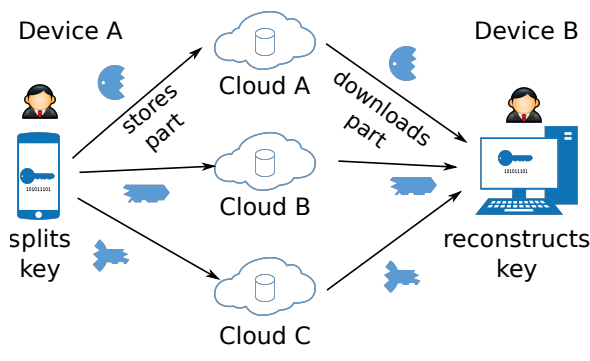
Implementation within PRACTICE overall architecture:



Remarks:

- This scenario can be implemented on Sharemind assuming honest majority and using secure hardware to protect against malicious adversaries.
- This application can be implemented with Fresco in an instantiation with two computing parties, since currently Fresco supports only two-party secure computation with malicious security. However, the currently supported protocol (SPDZ) for this setting can easily be extended to multiple parties.

4.4 End User Applications

<p>Scenario: Key Management</p>																				
<p>Summary: The increasing use of multiple devices by the same person for business and other purposes has amplified the annoyance of making cryptographic keys available across different platforms and devices. Typing cryptographic keys into a smart phone interface is at best very impractical. Copying the key to a media (e.g. USB) is not possible for many devices. Emailing the key, using sharing services or a central key server is in most cases a security liability. A solution to enable an easy access to cryptographic keys is to use SMC by delegating the trust to multiple cloud providers. In this way the required security properties can be achieved.</p>																				
<p>Scenario Illustration:</p>  <p>The diagram illustrates the process of key management. On the left, 'Device A' (represented by a smartphone icon) is shown 'splitting key' into three parts. These parts are stored in three separate cloud providers: 'Cloud A', 'Cloud B', and 'Cloud C'. On the right, 'Device B' (represented by a laptop icon) is shown 'downloading part' from each of the three clouds and then 'reconstructs key'.</p>	<p>Participants:</p> <ul style="list-style-type: none"> • $P1$: User ICR • $P2$: Cloud service providers SS^n 																			
<p>Communication Channels: The communication channels between the participating parties.</p> <ul style="list-style-type: none"> • $P1 \leftrightarrow P2$ 	<p>Adversary Model:</p> <table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (collectively)				trusted	semi-honest	covert	malicious	$P1$	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Party (indiv. entities)	Party (collectively)																			
	trusted	semi-honest	covert	malicious																
$P1$	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>																
<p>Trust Model:</p> <ul style="list-style-type: none"> • $P1$ is the owner of the key and is trusted. • The Cloud Service Provider (CSP) is considered malicious. However, it is assumed that the different CSPs are not colluding to reconstruct the key from the secret shares stored at different CSPs. 																				

Communication Model:

- A number (depending on configuration) of servers must be reachable for the client to obtain keys.
- If one or more servers are unreachable when the client attempts to upload key material, the other servers can distribute the appropriate key shares to the servers when they get back online. However, this must happen in a way that never the whole key is stored at a single CSP. For instance, the same share could be stored on multiple CSPs for reliability, say CSP A and A' . If A' is offline at the time of uploading one share of the secret is stored at A . When A' gets available again, the share from A is copied to A'

System Model:

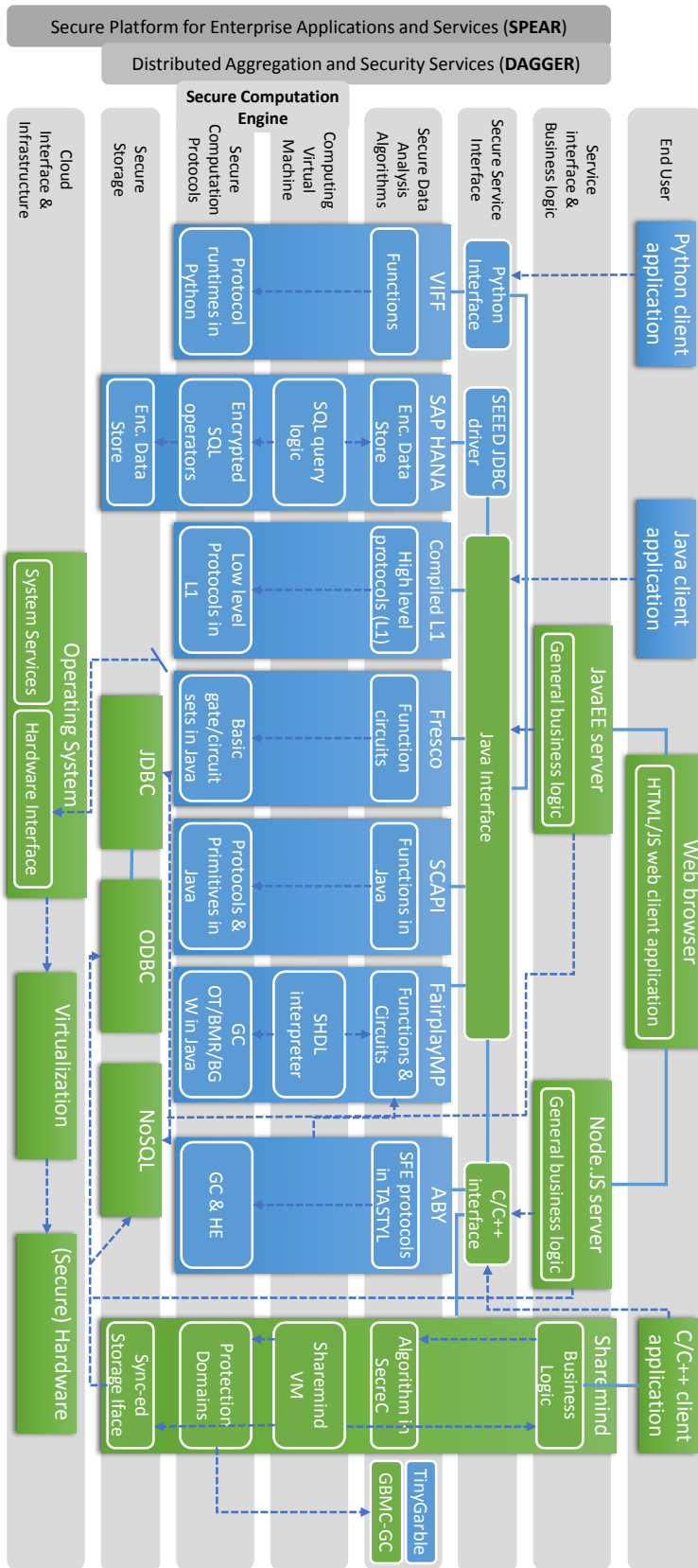
- The platform should run on commodity cloud service providers.
- The user can download and access keys from the cloud providers from a range of devices.
- The applications running on clients should be lightweight with regard to memory and cpu usage.

Required Guarantees:

- Ensure that no individual cloud provider can obtain the keys.

Workpackage References:**Literature References:**

Implementation within PRACTICE overall architecture:



Remarks:

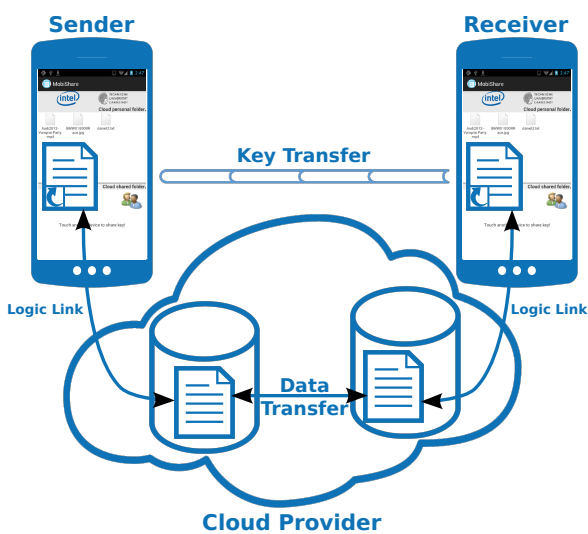
- Assuming the cloud service providers are non-colluding, this can be implemented with Sharemind easily, as Sharemind is based on secret sharing. Authenticity of the shares can be guaranteed by standard cryptographic methods. However, since there are no actual secure computations taking place in this application, an SMC platform is not needed at all.

Scenario: Mobile Data Sharing

Summary:

Mobile Data Sharing enables users of cloud storage services to encrypt their data in the cloud while still being able to share the data with other (trusted) users. All files are always encrypted while residing in the cloud; the files are encrypted when stored and also while they are transferred between users. For the receiver to be able to access the encrypted data she has to get access to the corresponding keys. These keys are transferred via a secure channel between the sender and the receiver, the secure channel is established based on a shared secret exchanged between the users smart phones, e.g. via NFC.

Scenario Illustration:



Participants:

- $P1a$: Cloud storage service provider (source storage) SS
- $P1b$: Cloud storage service provider (destination storage) SS
- $P2$: Sender IC
- $P3$: Receiver RC
- $P4$: Cloud (storage) service provider (intermediate storage) C

Communication Channels:

- Channel1: $P1a \leftrightarrow P2$
- Channel2: $P1b \leftrightarrow P3$
- Channel3: $P2 \leftrightarrow P3$
- Channel4: $P1a \leftrightarrow P4$
- Channel5: $P1b \leftrightarrow P4$

Adversary Model:

Party (indiv. entities)					Party (collectively)	honest majority
	trusted	semi-honest	covert	malicious		
$P1a$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1a$	<input type="checkbox"/>
$P1b$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1b$	<input type="checkbox"/>
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>

Trust Model:

- Both users who exchange files using this application trust each other which respect to sharing specific files. However, the users do not fully trust each other, i.e., they do not want to give the other party full access to *all* their files. Hence, the users are considered covert, since they might try to get access to files which were not explicitly shared with them but the personal relationship between the users will prevent malicious behaviour.
- Each users uses his smartphones as trust anchors. This means that the users trust their smartphones and the MobiShare application and enable it to access their personal cloud storage.
- The cloud service provider is not trusted. That is why all user files stored in the cloud are encrypted at all times.
- A security enhancement for this application is to perform critical operations such as key management inside a Trusted Execution Environment (TEE) with secure I/O feature, e.g., ARM TrustZone. The level of trust in the user device is reduced this way, because only the code in the TEE has to be trusted, not the entire smartphone.

Communication Model:

- The participants must be online during the whole process.
- There is no need for simultaneous communication between the parties, except for the key exchange.
- The cloud service provider must be online and reachable only during the communication with user's device or another cloud service provider. Delayed response is tolerated in this application scenario to a large extent. However, if the delay is too high or if no response is received at all, the process will fail and file exchange won't complete.
- Because smartphones are used in this application, the existence of a reliable network connection is required for the communication between sender/receiver and cloud service provider (channel 1,2), but not for the communication between sender and receiver (channel 3), because no Internet connection is required for the establishment of a session key between sender and receiver, but other technologies such as NFC or Bluetooth. Due to fast and reliable infrastructure of cloud service providers, there is no concern about speed and reliability of communication channel 4.
- The file transfer must happen in a transactional communication.
- No (technical) authentication is required for channel 3, because sender and receiver authenticate each other visually before starting key exchange. Users need to be authenticated prior to accessing their cloud storage via their smartphones.
- Because user files and secret keys are encrypted, no information will leak if integrity of communication is violated. The violation of communication integrity is detectable, as the received encrypted file keys cannot be decrypted with the key exchanged between sender and receiver.
- Transport encryption is not required, but recommended.
- Replay attack is not possible, because the exchanged secret key is chosen randomly during each session. Moreover, the physical proximity required in NFC prevents data interception.

System Model:

- The operations on the user side are not resource consuming and thus should run on commodity smartphones with typical computational power and memory. The mobile application should be responsive considering limited capabilities of smartphones.
- A high network bandwidth is required for transferring big files in a reasonable time. However files are transferred among cloud service providers which usually have high and reliable network bandwidth.
- Users maintain control over their data by encrypting them with random keys before uploading them to the cloud. For the large files to be uploaded in an acceptable time frame, a certain transmission rate is required. However in the course of sharing, the amount of data being communicated between users' smartphones and cloud service providers is small in size, a low network bandwidth, such as mobile network, is sufficient for this purpose. The connection must be however reliable.
- Each user has a master key with which file encryption keys are encrypted. This key is kept on user's device all the time. If user device is compromised, this master key will be accessible to the attacker.

Required Guarantees:

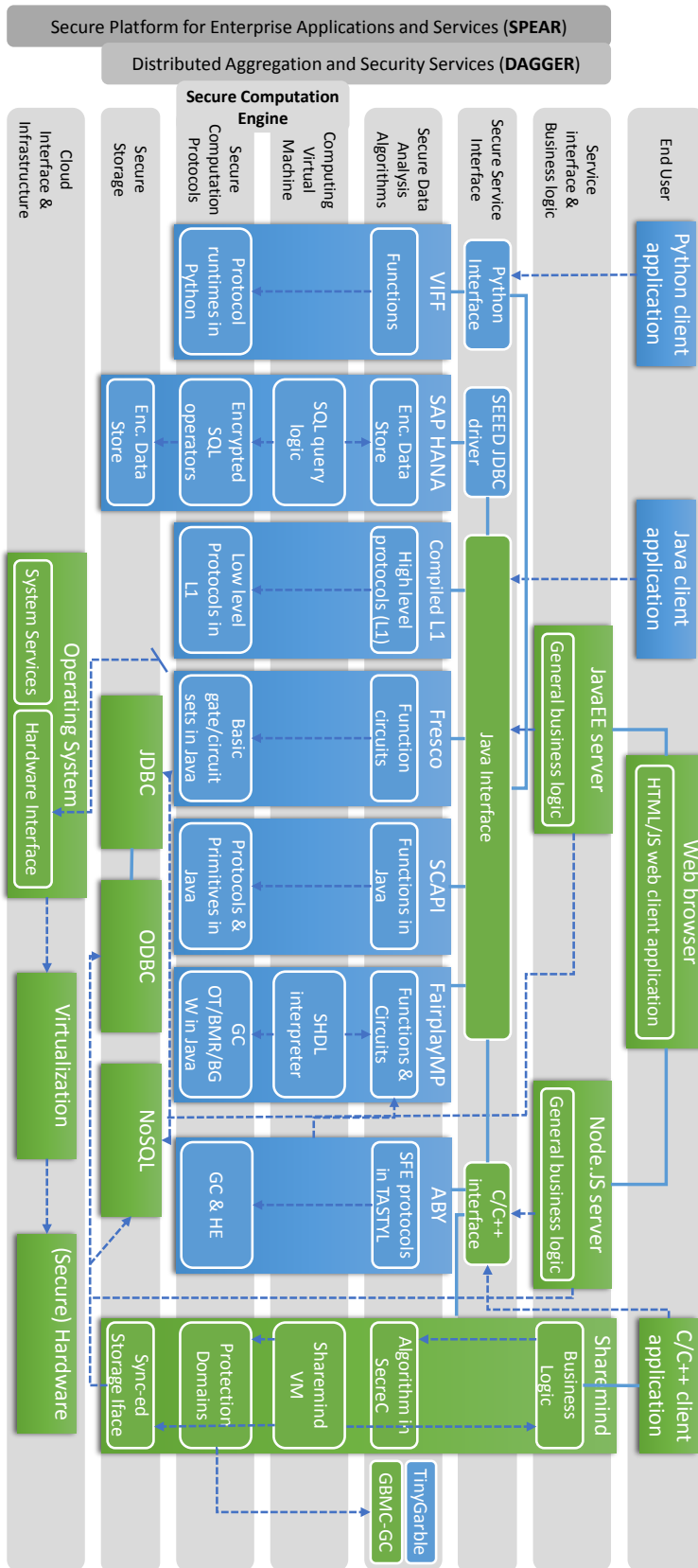
- Provide confidentiality for data in the cloud.
- Data is only shared with others when intended by the data owner (sender).
- The sender's and receiver's master keys must be kept confidential to the user.
- The received encrypted file key must be decryptable using receiver's master key.

Workpackage References:
WP21.1

WP22.1,

Literature References:

Implementation within PRACTICE overall architecture:



Remarks:

- Assuming the cloud service providers are non-colluding, this can be implemented with Sharemind easily, as Sharemind is based on secret sharing. Authenticity of the shares can be guaranteed by standard cryptographic methods. However, since there are no actual secure computations taking place in this application, an SMC platform is not needed at all.

Chapter 5

Formalizing Security Models

In Chapter 2, we have presented a high-level overview of the participants in secure outsourcing scenarios, and the way in which security in such scenarios can be described. Indeed, describing security consists of defining an *adversary model* (specifying from which of the participants of the system we have to assume malicious behaviour, and what kind of malicious behaviour should be assumed); a *trust model* (specifying trust assumptions that go beyond the adversary, e.g., the use of a public key infrastructure); a *communication model* (specifying how and when the participants communicate); and a *system model* (specifying general constraints that do not fall into the above categories).

While the above model is appropriate for describing potential scenarios for secure outsourcing, it is not sufficiently precise for analysing cryptographic implementations of the underlying secure computation techniques. In particular, definitions in the literature on what encompasses secure computation do this using frameworks based on the real/ideal-world paradigm. As discussed in D11.1 [43], this model demands that the outputs of the participants in a protocol distribution are similar to those outputs in an ideal world where the function is computed by an incorruptible trusted party. Because in the ideal world, the result party obtains the correct result and the adversary does not learn anything it should not learn, the same must be true in the real world. This paradigm captures, in a very precise way, the adversary model (by stating under which assumptions the outputs should be distributed similarly); the trust model (by assuming access to external “functionalities” such as a public key infrastructure); and the communication model (by defining in which way the participants are supposed to interact).

In this chapter, we aim to provide an “implementation” of the informal security model from the previous chapters into a precise real/ideal-world formalisation. This way, it is possible to formally prove that secure computation techniques we apply to implement the scenarios, do indeed provide the appropriate security guarantees. Compared to the mainstream literature on defining secure computation, we need adaptations to reflect our focus on outsourcing. In particular, the mainstream literature typically does not distinguish between different participants: all participants provide inputs, participate in the computation, and receive outputs. Moreover, security guarantees are typically defined symmetrically, e.g., the protocol works if at most two participants misbehave (regardless of which two parties it are). In contrast, we have distinguished between input, computation, result, and verifying parties that each have different roles in the computation; and as our analysis shows, different assumptions on misbehaviour for these different kinds of parties are appropriate. Hence, we need to redefine mainstream models to be able to fully define security of outsourcing scenarios.

5.1 The Overall Security Model: Ideal versus Real

We first present a high-level overview of the ideal/real paradigm used to define security of secure computation. By instantiating this paradigm in a careful way, it is possible to capture different adversary, trust, and communication models.

Many technical definition frameworks based on the ideal/real paradigm exist. The seminal work in this area is Canetti's model for secure function evaluation [20], that for the first time gave a complete picture of secure function evaluation using the ideal/real paradigm. This model is the foundation of most of the practical secure computation techniques in use today, and allows us to express all aspects of the adversary, trust, and communication models that we are interested in. However, it has one main limitation: it technically only guarantees security when all protocol participants are only active in one secure computation at a time. In practice, it needs to be assumed that different secure computations do not share key material, and that outputs of an ongoing secure computation are not used as inputs to another secure computation. Canetti later extended his model to overcome this limitation by defining the concept of *universal composability* [21]; other more general models also exist, e.g., [35]. Unfortunately, security in these more general models typically comes at the price of much less efficient implementations. In this chapter, we will consider Canetti's original model. Hence, our security model is an adaptation of the model of [20] to the setting of secure outsourcing. We first explain the general execution model, which basically follows [20]; we then explain how to model security guarantees in this execution model as the behaviour of the ideal-world trusted party.

5.1.1 Execution Model: Trust and Communication Models

The general execution model compares protocol executions in the real and ideal world. In the real world, a protocol Π between m input parties \mathcal{I}^m , n computation parties \mathcal{P}^n , and k result parties \mathcal{R}^k is executed in the presence of an adversary \mathcal{A} corrupting parties $C \subset \mathcal{I} \cup \mathcal{C} \cup \mathcal{R}$. (We discuss security in the presence of external verifiers in the next section.) Each input party $i \in \mathcal{I}$ gets input x_i ; the other parties get no input. The adversary gets the inputs $\{x_i\}_{i \in \mathcal{I} \cap C}$ of the corrupted parties, and has an auxiliary input a . At the end of the protocol, each honest party outputs a value according to the protocol; the corrupted parties output \perp ; and the adversary outputs a value at will. Define $\text{EXEC}_{\pi, \mathcal{A}}(k, (x_1, \dots, x_m), C, a)$ to be the random variable, given security parameter k , consisting of the outputs of all parties (including the adversary).

By working out the details of the real-world execution model, the trust and communication models are specified (see, e.g., [20, 23]). The communication model can be captured by imposing restrictions on the type of communication allowed in the real-world execution: for instance, the parties may or may not have access to a broadcast channel; or some of the parties may or may not be active at some point in time. The trust model is typically captured by giving parties access to so-called "functionalities". These are parts of the protocol that are only available to the protocol parties as "black boxes", and that hence cannot be corrupted. For instance, a public-key infrastructure, or in general, the secure set-up of keys, are commonly captured by giving parties access to a "set-up functionality"; because adversaries cannot obtain the keys except by invoking these functionalities, this captures the assumption that the set-up was performed correctly. (In more recent works based on universal composability, the broadcast channel is also modelled as a functionality instead of as an integral part of the execution model.)

The ideal-world execution similarly involves m input parties $i \in \mathcal{I}$, n computation parties $i \in \mathcal{C}$, n result parties $i \in \mathcal{R}$, and an adversary \mathcal{S} corrupting parties $C \subset \mathcal{I} \cup \mathcal{C} \cup \mathcal{R}$; but now, there is also an incorruptible trusted party \mathcal{T} . As before, the input parties receive x_i as input; the trusted party receives a list C of corrupted parties and the public key pk . Then, it runs the some code that captures

the computation to be carried out, as we discuss next. The adversary gets inputs $(C, \{x_i\}_{i \in \mathcal{I} \cap C})$, and outputs a value at will. As in the real-world case, $\text{IDEAL}_{\mathcal{S}}(k, (x_1, \dots, x_m), C, a)$ is the random variable, given security parameter k , consisting of all parties' outputs.

5.1.2 The Ideal-World Trusted Party

The code run by the ideal-world trusted party captures which secure computation takes place. Given input C , the trusted party performs the following steps:

- Receive inputs x_i from all uncorrupted input parties in $\mathcal{I} \setminus C$
- Receive inputs x_i on behalf of the corrupted inputs from \mathcal{S}
- Compute $(y_1, \dots, y_n) = f(x_1, \dots, x_m)$
- Send y_i to all non-corrupted result parties $i \in \mathcal{R}$, or to the adversary if they are corrupted.

As discussed in D11.1 [43], this algorithm for the trusted party indeed implies privacy (the only thing the adversary learns is the result of corrupted result parties); correctness (the trusted party determines the result); independence of inputs (the trusted party asks for all inputs before producing any output); and fairness (the trusted party sends the result to all parties).

5.1.3 Security Definition and Adversary Model

Given the two above mentioned distributions $\text{EXEC}_{\pi, \mathcal{A}}(k, (x_1, \dots, x_m), C, a)$ and $\text{IDEAL}_{\mathcal{S}}(k, (x_1, \dots, x_m), C, a)$ that depend on the trust and communication models, the adversary model is traditionally captured by specifying the conditions under which these distributions correspond. For instance, the adversary may state that the input parties may be malicious; the majority of the computation parties is honest; and the result party is trusted. In this case, security is stated by demanding that, for any adversary \mathcal{A} in the real world that corrupts any number of input parties and a minority of computation parties, there exists an adversary \mathcal{S} in the ideal world such that $\text{EXEC}_{\pi, \mathcal{A}}(k, (x_1, \dots, x_m), C, a)$ and $\text{IDEAL}_{\mathcal{S}}(k, (x_1, \dots, x_m), C, a)$ are indistinguishable.

indistinguishability may be perfect (i.e., as probability distributions, REAL and IDEAL are the same), statistical (i.e., the difference between REAL and IDEAL shrinks quickly as k increases), or computational (i.e., the ability of an adversary to see the difference between REAL and IDEAL shrinks quickly as k increases). Proving security in such a model is usually done by, given access to \mathcal{A} as a “black box”, explicitly constructing an adversary \mathcal{S} that interacts with the ideal-world trusted party yet simulates the actions of the real-world computation parties to \mathcal{A} . Also, as discussed in deliverable D11.1 [43], the adversary model in a technical sense should also define the computational complexity of the attacker (polynomial-time or computationally unbounded) and the corruption strategy (corrupted parties are known beforehand or not); although the more general case is of course always better, which variant to choose does not directly follow from the scenario at hand.

5.2 Corruption-Dependent Trusted Parties and Verifiability

While the model presented above is a straightforward translation of existing models to the outsourcing setting, it is not so easy to define verifiability by performing such a translation exercise. For normal secure computation, we demand privacy, correctness, independence of inputs, and fairness, all under the same conditions for the adversary. For verifiability, in effect different properties hold under different assumptions about the honesty of the protocol participants: privacy and fairness only under some conditions (e.g., half of the computation parties are honest); but correctness and independence of inputs should hold under much weaker conditions (even if *none* of the computation parties is

honest).¹ To be able to capture this more complicated combination of guarantees, we allow the adversary to let its behaviour depend on which parties are corrupted (cf. [12, 44]). For instance, if all computation parties are corrupted, then privacy is not guaranteed, which we capture by sending all inputs to the adversary; but correctness *is* still guaranteed, which we capture by still letting the trusted party compute the function at hand.

A second issue arises when modelling *universal* verifiability [32], where the verifier does not necessarily learn the output of the computation that it is verifying. The problem here is how to model that a verifier is sure about correctness of a result, without learning its result. A naive solution would be to simply let the ideal-world trusted party send a “success” flag to the verifier that captures whether or not the secure computation was successfully performed. However, this would not enforce a link between the proof that the verifier sees and the result that result parties obtain. For instance, suppose that, at the end of a secure computation, the result party obtains the result, as well as a proof of correctness that it can then forward to a verifier. In this case, the ideal-world success flag for the verifier would correspond to whether or not the verifier obtains a valid proof, hence such a protocol would indeed implement the “success flag” ideal functionality. However, a little bit of thought reveals that also a protocol in which the result party simply decides on the success flag and forwards it to the verifier would also work: indeed, if the result party is not corrupted then it does this correctly, and if the result party is corrupted, then it can choose whether or not to convince the verifier any way. The fundamental problem here is that the verifier’s output in the IDEAL distribution, “yes” or “no”, does not say anything about what computation the verifier accepts.

To solve this issue, we have chosen to explicitly model the concrete representation of the output that the verifier obtains. Here, we assume that the representation of the result that a verifier checks may be an encryption of which it does not have the private key. As a consequence, we need to modify the execution model slightly so that the secure set-up of the used encryption scheme is performed at the beginning, and the trusted party obtains the public key of this scheme as one of its inputs.

5.2.1 Model of Universally Verifiable Secure Function Evaluation

Finally, Algorithm 1 shows a possible trusted party for universal verifiability [44]. As discussed above, adding universal verifiability to a protocol changes the correctness and independence of inputs properties, but not privacy and fairness. Hence, security of universally verifiable protocols is modelled by taking an existing trusted party that works under certain conditions, and then ensuring that if these conditions do not hold, correctness is still guaranteed. In Algorithm 1, we assume abstract “corruption conditions” under which privacy no longer holds, and under which the computation parties have the power to block sending the result to the result party. For simplicity, we model here the case when there is only one result party and one verifier. Security in this model is defined as follows:

Definition 1 ([44]) *Protocol π implements verifiable secure function evaluation if, for every probabilistic polynomial time real-world adversary \mathcal{A} , there exists a probabilistic polynomial time ideal-world adversary $\mathcal{S}_{\mathcal{A}}$ such that, for all inputs x_1, \dots, x_m ; all sets of corrupted parties C ; and all auxiliary input a : $\text{EXEC}_{\pi, \mathcal{A}}(k; x_1, \dots, x_m; C; a)$ and $\text{IDEAL}_{\mathcal{T}_{\text{ver}}, \mathcal{S}_{\mathcal{A}}}(k; x_1, \dots, x_m; C; a)$ are computationally indistinguishable in security parameter k .*

Note that, while previous definitions captured the adversary model by specifying *for which* \mathcal{A} computational indistinguishability holds, this model requires computation indistinguishability *for all* \mathcal{A} . Instead, the adversary model is now part of the the algorithm of the trusted party.

¹One can question whether or not unconditionally demanding independence of inputs should be part of verifiability. In this chapter, we choose to demand this: essentially, we see independence of inputs as a correctness requirement. It turns out that indeed, enforcing this property is realistic for practical protocols.

Algorithm 1 \mathcal{T}_{vsfe} : trusted party for verifiable secure function evaluation

```

1. // compute  $f$  on  $\{x_i\}_{i \in \mathcal{I}}$  for  $\mathcal{R}$  with corrupted parties  $C$ ;  $\mathcal{V}$  learns encryption
2.  $\mathcal{T}_{vsfe}(C, pk) :=$ 
3.   // input phase
4.   foreach  $i \in \mathcal{I} \setminus C$  do  $x_i := \text{recv}(\mathcal{I}_i)$  // honest inputs
5.    $\{x_i\}_{i \in \mathcal{I} \cap C} := \text{recv}(\mathcal{S})$  // corrupted inputs
6.   if  $\langle \text{corruption condition} \rangle$  then  $\text{send}(\{x_i\}_{i \in \mathcal{I} \setminus C}, \mathcal{S})$  // send to adversary
7.   // computation phase
8.    $r := f(x_1, \dots, x_m)$ 
9.   // output phase
10.  if  $\mathcal{R} \notin C$  then // honest  $\mathcal{R}$ : adversary learns encryption, may block result
11.     $s \in_R \langle \text{randomness} \rangle$ ;  $R := \text{Enc}_{pk}(r; s)$ ;  $\text{res} := (r, s)$ ;  $\text{send}(R, \mathcal{S})$ 
12.    if  $\langle \text{corruption condition} \rangle$  and  $\text{recv}(\mathcal{S}) = \perp$  then  $\text{res} := \perp$ ;  $R := \perp$ 
13.     $\text{send}(\text{res}, \mathcal{R})$ 
14.  else // corrupted  $\mathcal{R}$ : adversary learns output, may block result to  $\mathcal{V}$ 
15.     $\text{send}(r, \mathcal{S})$ ;  $s := \text{recv}(\mathcal{S})$ 
16.    if  $s = \perp$  then  $R := \perp$  else  $R := \text{Enc}_{pk}(r; s)$ 
17.  // proof phase
18.  if  $\mathcal{V} \notin C$  then  $\text{send}(R, \mathcal{V})$ 

```

We now discuss the trusted party \mathcal{T}_{vsfe} for verifiable secure function evaluation. Whenever the computation succeeds, \mathcal{T}_{vsfe} guarantees that the results are correct. Namely, \mathcal{T}_{vsfe} sends the result r of the computation and randomness s to \mathcal{R} (line 13), and it sends encryption $\text{Enc}(r; s)$ of the result with randomness s to \mathcal{V} (line 18); if the computation failed, \mathcal{R} gets (\perp, \perp) and \mathcal{V} gets \perp .² Whether \mathcal{T}_{vsfe} guarantees privacy (i.e., only \mathcal{R} can learn the result) and robustness (i.e., the computation does not fail) depends on which parties are corrupted. Privacy and robustness with respect to \mathcal{R} are guaranteed as long as a certain, protocol-dependent, “corruption condition” is met. If not, then in line 6, \mathcal{T}_{vsfe} sends the honest parties’ inputs to the adversary; and in line 12, it gives the adversary the option to block the computation by sending \perp . Note that the adversary receives the inputs of the honest parties after it provides the inputs of the corrupted parties, so even if privacy is broken, the adversary cannot choose the corrupted parties’ inputs based on the honest parties’ inputs. For robustness with respect to \mathcal{V} , another “corruption condition” needs to be met, e.g., the result party needs to be honest. If not, then in line 15, \mathcal{T}_{vsfe} gives the adversary the option to block \mathcal{V} ’s result by sending \perp ; in any case, it can choose the randomness.

Note that this model does not cover the “universality” aspect of universally verifiable MPC. This is because the security model for secure function evaluation only covers the input/output behaviour of protocols, not the fact that “the verifier can be anybody”. Hence, we design universally verifiable protocols by proving that they are verifiable, and then arguing based on the characteristics of the protocol (e.g., the verifier does not have any secret values) that this verifiability is “universal”.

² Although we only guarantee computational indistinguishability and the verifier does not know what value is encrypted, this definition *does* guarantee that \mathcal{V} receives the correct result. This is because the ideal-world output of the protocol execution contains \mathcal{R} ’s r and s and \mathcal{V} ’s $\text{Enc}(r; s)$, so a distinguisher between the ideal and real world can check correctness of \mathcal{V} ’s result. (If s were not in \mathcal{R} ’s result, this would not be the case, and correctness of \mathcal{V} ’s result would *not* be guaranteed.) Also, note that although privacy depends on the security of the encryption scheme, correctness *does not* rely on any knowledge assumption.

5.3 Combining Secure Computation Engines for Validation

In this section, we show that it is also possible to formalise security for secure computations in which different computation engines with different security guarantees are combined. Indeed, combining different computation engines occurs naturally when making computations verifiable. As discussed in D11.1 [43], making a full computation verifiable asks for large computational effort for the verifier. To avoid this, we can exploit [24, 25] the fact that for many practical problems (e.g. integer division, computation of matrix inverse/eigenvalues), it is easier to check a given solution than to compute it. Hence, the proposal is to make a computation verifiable by computing the solution using a normal, fast, secure computation engine; and the computing the validation circuit using a (probably slower) computation engine that offers verifiability. However, modelling security in this case becomes challenging: we now have two different secure computation engines, each with different adversary, trust, and system models; and we need to combine their security guarantees into one overall model. In more detail, the idea of validation is to combine fast computation of a solution and “certificate” of its correctness with verifiable validation that the solution is correct with respect to the certificate. Consider the example of computing an eigenvalue λ of a matrix M . It is not easy to compute λ , but it is easy to verify its correctness using an eigenvector v as “certificate” by checking that $Mv = \lambda v$. Hence, the idea is to compute output λ and certificate v using a normal secure computation engine; and then check $Mv = \lambda v$ using a verifiable secure computation engine. More generally, suppose we have four kinds of parties: m input parties $i \in \mathcal{I}$, n computation parties $i \in \mathcal{C}$, a result party \mathcal{R} , and a verifier \mathcal{V} . The computation parties use one secure computation engine to compute $(\vec{a}, \vec{r}) = f(\vec{x})$, where \vec{x} is the input (of length m); \vec{a} is the certificate (say, of length k); and \vec{r} is the output (say, of length l). Let ϕ be a certificate such that, if $(\vec{a}, \vec{r}) = f(\vec{x})$, then $\phi(\vec{x}, \vec{a}, \vec{r})$ holds.³ The computation parties then use a verifiable secure computation engine to compute a proof that $\phi(\vec{x}, \vec{a}, \vec{r})$ is indeed the case, and deliver the result to the result party, and an encryption to the verifier. The idea is that the combined protocol inherits its privacy and robustness properties from the protocols for computing f ; but guarantees correctness regardless of corruptions.

5.3.1 Trusted Party for Validation

We have developed a trusted party [25] that combines security guarantees of different secure computation engines to achieve validation. Our trusted party \mathcal{T}_{val} for universal verifiability by certificate validation is shown in Algorithm 2. As inputs, it receives a set C of corrupted parties, of which those in A are actively corrupted; and public key pk used to encrypt the result for the verifier. First, the trusted party receives the inputs of the honest and corrupted parties (lines 4–5). We now need to distinguish two different “corruption conditions”: one under which the secure computation condition used to compute f no longer guarantees privacy, and one under which it no longer guarantees correctness. If the former corruption condition is not satisfied, then \mathcal{T}_{val} sends the inputs to the adversary (line 6). (Note that this happens *after* the adversary supplies its inputs, so this model guarantees independence of inputs.) The computation differs depending on whether the latter corruption condition is satisfied, i.e., on whether the first secure computation engine may manipulate the computation of f . If not, then the computation is performed according to function f (line 9). If so, then the adversary might arbitrarily interfere with the computation of f , which we capture by letting the adversary choose outputs a and r . However, the adversary can only choose outcomes for which ϕ holds, so \mathcal{T}_{val} checks this and otherwise sets r to \perp (line 11). Finally, the trusted party produces encryptions which it provides to the verifier (line 16);

³Note that we do not demand the converse, i.e., if $\phi(\vec{x}, \vec{a}, \vec{r})$, then it is not necessarily true that $(\vec{a}, \vec{r}) = f(\vec{x})$. For instance, the solution \vec{r} might not be unique: then f is an algorithm to find some solution, and ϕ merely checks if the solution is valid. Indeed, this will be the case for some applications.

Algorithm 2 \mathcal{T}_{val} : trusted party for verifiability by certificate validation

```

1. // compute  $f$  on  $\{x_i\}_{i \in \mathcal{I}}$  for  $\mathcal{R}$  with corrupted parties  $C$  and check  $\phi$ ;  $\mathcal{V}$  learns encryption
2.  $\mathcal{T}_{val}(C, A, pk) :=$ 
3.   // input phase
4.   foreach  $i \in \mathcal{I} \setminus C$  do  $x_i := \text{recv}(\mathcal{I}_i)$  // honest inputs
5.    $\{x_i\}_{i \in \mathcal{I} \cap C} := \text{recv}(\mathcal{S})$  // corrupted inputs
6.   if  $\langle \text{corruption condition} \rangle$  then  $\text{send}(\{x_i\}_{i \in \mathcal{I} \setminus C}, \mathcal{S})$  // send to adversary
7.   // computation phase
8.   if  $\langle \text{corruption condition} \rangle$  then
9.      $a_1, \dots, r_l := f(x_1, \dots, x_m)$ 
10.  else
11.     $a_1, \dots, r_l := \text{recv}(\mathcal{S})$ ; if  $\neg \phi(x_1, \dots, r_l)$  then  $r_1, \dots, r_l \leftarrow \perp$ 
12.    if  $\langle \text{corruption condition} \rangle$  then  $s_1, \dots, s_l \leftarrow \text{recv}(\mathcal{S})$  else  $s_1, \dots, s_l \in_R \langle \text{randomness} \rangle$ 
13.    // result phase
14.    if  $r_1, \dots, r_l \neq \perp$  then  $R_1, \dots, R_l := \text{Enc}_{pk}(r_1; s_1), \dots, \text{Enc}_{pk}(r_l; s_l)$ 
15.    else  $R_1, \dots, R_l \leftarrow \perp$ 
16.     $\text{send}(R_1, \dots, R_l; \mathcal{S})$ ;  $\text{send}(R_1, \dots, R_l; \mathcal{V})$ 
17.    if  $\langle \text{corruption condition} \rangle$  then  $\text{send}((r_1, s_1), \dots, (r_l, s_l); \mathcal{R})$ 

```

and it provides the corresponding plaintext and randomness to the result party (line 17); this latter step can be blocked by the adversary.

For our formal security definition, note that we need to slightly change the previous definition of the REAL and IDEAL distributions. Namely, apart from taking the set C of actively corrupted parties, we now need to refine the adversary model by allowing some parties to be actively corrupted and other parties only passively. Indeed, in practice, corruption conditions in \mathcal{T}_{val} will require this information. Hence, we add an argument A that captures which parties are actively corrupted; $C \setminus A$ are only passively corrupted. (We do not consider other types of attacker such as covert attackers in this particular definition, but they could be added in a straightforward way.) We get the following security definition:

Definition 2 ([25]) *Protocol Π implements secure function evaluation with universal verifiability by certificate validation if, for all adversaries \mathcal{A} corrupting set C of parties and actively corrupting $A \subset C$, there exists an adversary \mathcal{S}_A such that for all possible inputs x_1, \dots, x_m to f ,*

$$\text{REAL}_{\Pi, \mathcal{A}}(k; x_1, \dots, x_m; A; C; a) \stackrel{d}{=} \text{IDEAL}_{\mathcal{T}_{val}, \mathcal{S}_A}(k; x_1, \dots, x_m; A; C; a) .$$

(Here, $\stackrel{d}{=}$ denotes computational indistinguishability.)

5.4 Secure Computation in the Random Oracle Model

Finally, we briefly outline how definitions of secure function evaluation can be interpreted in the random oracle model, and what consequences this has for our security definitions. The random oracle model [13, 51] is an idealised model of hash functions. In this model, evaluations of the hash function \mathcal{H} are modelled as queries to a “random oracle” \mathcal{O} that evaluates a perfectly random function. When simulating an adversary, a simulator can intercept these oracle queries and answer them at will, as long as the answers look random to the adversary. Security in the random oracle model does not generally imply security in the standard model [31], but it is often used because it typically gives

simple, efficient protocols, and its use does not seem to lead to security problems in practice [51]. Indeed, the random oracle model is used for common constructions of non-interactive proofs, e.g., proofs of correctness of a computation. Hence, supported by existing work [24, 12], we expect that practical constructions for universal verifiability will need to be defined in this model.

5.4.1 Adapting Security Models

More precisely, in the random oracle model, evaluations of hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^{2l}$ are modelled as queries to a “random oracle” \mathcal{O} that evaluates a perfectly random function. When simulating an adversary that operates in the random oracle model, the simulator also simulates the random oracle with respect to the adversary. In particular, it can choose how to respond to the adversary’s queries (but, to achieve security, it should provide random values so that the adversary cannot distinguish between the real world and the simulation based on the output of the random oracle).

The simplest variant of the random oracle model is the so-called explicitly programmable random oracle model without dependent auxiliary input [51]. The random oracle is seen as a partial function that initially has an empty codomain (i.e., it is “without dependent auxiliary input”). In a real-world execution in this model, both the honest parties and the adversary use the random oracle for hash function evaluations. Namely, when a party calls the oracle on a value $v \in \text{dom}(\mathcal{O})$, it receives $\mathcal{O}(v)$; otherwise, a fresh random value is generated and \mathcal{O} is updated accordingly. At the end of the execution, \mathcal{O} contains all pairs of oracle queries made during the execution and their responses. In an ideal-world execution, the simulator can directly modify the pre-image/image pairs in \mathcal{O} ; the simulated adversary only has oracle access to \mathcal{O} as in the real-world execution. Again, at the end of the simulation, \mathcal{O} contains all values on which the oracle has been set. Computational (or statistical) indistinguishability between real and simulated executions is defined [51] by stating that no probabilistic polynomial time (or unbounded) algorithm can distinguish them, where the distinguisher has oracle access to \mathcal{O} .

To use this model in the security model for secure computation, we can add the random oracle to the IDEAL and REAL distributions defined earlier in this chapter. In particular, define $\text{EXEC}_{\pi, \mathcal{A}}(k, (x_1, \dots, x_m), C, a)$ to be the random variable, given security parameter k , consisting of the outputs of all parties (including the adversary) and the set \mathcal{O} of actual oracle queries and responses that were performed. In the ideal execution, there is no random oracle; instead, the adversary chooses the set \mathcal{O} of oracle queries and responses (typically, those used to simulate a real-world adversary). As in the real-world case, $\text{IDEAL}_{\mathcal{T}_{\text{val}}, \mathcal{S}}(k, (x_1, \dots, x_m), C, a)$ is the random variable, given security parameter k , consisting of all parties’ outputs and \mathcal{O} . Note that, when defining security based on these distributions, we get slightly stronger versions of indistinguishability than mentioned above; namely, instead of giving the distinguisher oracle access to \mathcal{O} , we simply supply it with the full list \mathcal{O} . We can then simply use the normal, non-oracle, definitions for indistinguishability; this is clearly at least as strong.

5.4.2 Consequences for Practical Security

We remark that, while security in non-random-oracle secure function evaluation [20, 23] is preserved under (subroutine) composition, this is not the case for the random oracle variant presented here. The reason is that our model and protocols assume that the random oracle is not used outside of the protocol. Using the random oracle model with dependent auxiliary input [50, 51] might be enough to obtain a composition property; but adaptations are generally needed to make protocols provably secure in that model. This technical issue reflects the real-life problem that a verifier of a non-interactive proof cannot see if the proof has been recently produced, it is simply a replay from

an earlier computation transcript. As discussed in [50], these technical problems can be solved in practice by instantiating the random oracle with a keyed hash function, with every computation using a fresh random key.

Chapter 6

Secure Multi-Party Computation with Trusted Hardware

Secure multi-party computation SMC aims at outsourcing computations without revealing the input data of the computations to the computing entity. However, besides SMC there are other methods to achieve the goal of outsourcing computations while keeping the input data private. In the simplest case, if the computing party was trustworthy and would promise not to access the input data in an unintended way the goal could be considered achieved. However, in general the computing party cannot be assumed to be fully trustworthy, for different reasons. Those reasons include, that the computation party might have some inherent interest in revealing the input data. In other cases the computing party might be trustworthy in general but this trust cannot be fully extended to each and every of its sub-entities. For instance, a cloud service provider (CSP) might be trustworthy in general, since violation of customer's privacy will harm the CSP's business. But only because for the CSP as a whole it would be disadvantageous to violate customer's privacy this is not necessarily true for all sub-entities of the CSP, like the administrators maintaining the CSP's data center. An administrator might violate the customer's privacy for personal benefit (selling confidential business data to a comparator) or out of revenge because he was laid off. Another possibility would be that an external attacker gains administrator privileges and misuses them to violate the privacy of the cloud customers. Both cases, a malicious administrator and an attacker with administrative privileges, are considered insider attacks. The actions a computing party is performing on the data cannot be controlled. At least this is the case for standard computing systems. For the remaining of this chapter we will focus on the setup where computation of private data should be performed in the cloud. There, the computing party would be the CSP. For the reasons described above we cannot trust the CSP since we cannot assume that the CSP is invulnerable to inside attacks.

To face the fact that the cloud service provider cannot be trusted different measures can be taken, all aiming at preventing the disclosure of the private data that are to be processed. The first option is the use of secure multi-party computation (SMC). With SMC the data are not accessible by the CSP due to the distribution of the data and/or encryption of the data. The data in question is never fully and unencrypted in the possession of the CSP, hence, the CSP can never get access to the data. The second option is to transfer the full data to the CSP and making sure that the CSP never accesses the data. The first option is described in detail in the project as it is the focus subject of the project. The second option we will elaborate more on in this chapter, i.e., the use of trusted hardware to perform computation on sensitive data.

6.1 Trusted Hardware

Generally speaking there is a wide variety of trusted hardware used in many different use cases. Often the user of trusted hardware is not fully aware of the fact that he is using trusted hardware, for instance, smartcards or trusted execution environments in mobile devices. However, trusted hardware is often designed for special purpose use and limited in the capabilities. Furthermore, trusted hardware is usually considerably more expensive than commodity hardware.

To enable computing of sensitive data there are two important properties that must be given by a hardware. The first required property is that the sensitive data cannot be directly accessed by entities outside the trusted hardware. For instance, the memory content of a trusted hardware component should not be readable via an uncontrolled external interface. In certain scenarios it is even required that the data are protected from sophisticated hardware attacks that aim at extracting the data by physically tempering with the hardware. The second property that must be fulfilled is that there must be a way to control which operations are executed on the data. Uncontrolled operation on the data can easily lead to the revealing of the data. In the simplest case the data are outputted from the trusted hardware. Even if the data are not directly emitted from the trusted hardware indirect information can leave the trusted hardware which allow an attacker to gain knowledge about the processed sensitive data. Hence, it must be ensured that only those computations are performed on the data that are accepted by the data owner or some other entity responsible to prevent data leakage. This means that all software components that potentially could access the sensitive data must be validated to ensure that they do not reveal the sensitive data, directly or indirectly.

There are different kinds of trusted hardware which can be grouped in two categories. We will elaborate on these two different categories of trusted hardware subsequently. The first one is dedicated hardware like smartcards or hardware security modules. The second type are trusted execution environments (TEEs) which are built into general purpose processors.

6.1.1 Hardware Security Modules

Hardware Security Modules (HSMs) are used to perform computation on sensitive data in an adversarial environment. The dedicated trusted hardware is largely independent from all other components of the environment. There exists a wide variety of HSMs aiming at different use cases. Two main classes of HSMs can be distinguished: Smartcards and security processors as extensions for servers. While there is no clear definition telling those two apart there are some criteria which can serve to distinguish them. Smartcards are usually designed for a special propose and are limited in their functionality to perform the operations needed for this purpose.

Smartcards

Smartcards are most commonly used for authentication. Companies use smartcards to authenticate their employees when accessing their IT infrastructure. Cellphone providers use smartcards (known as SIM which is short for subscriber identity module) to identify their customer when accessing the cellular network. Banking cards have smartcard capabilities to authenticate the user and allowing access to her account. Since smartcards are dedicated hardware devices access to the data store and process on them is not directly possible from outside the smartcard. The interface of the smartcard prevents uncontrolled access to the data stored on it and physical protections prevent the access through physical tempering, which fulfills the first requirement stated above. Furthermore, the program code executed on a smartcard is either fixed or can only be loaded when digitally signed with an appropriate key. In both cases the code which is executed on the smartcard and which accesses the sensitive data can be controlled, i.e., it fulfills the second requirement stated above. However, due to

the nature of smartcards to be designed for a special purpose use they are limited in type of operations they can perform (e.g., only create digital signatures) and also constrained in their computational power. Hence, for general purpose computation on not markedly small data they are not well suited.

Security Processors

In contrast to smartcards security processors are usually capable of performing general purpose computations. They are mainly used in enterprise environments like servers to perform security critical task, e.g., key management. Security processors are dedicated devices which are independent of their environment and as for smartcards they have controlled interfaces to the outside world. Hence, they fulfil also the requirement of preventing direct access to the sensitive data as mentioned above. Furthermore, security processors are designed to be tamper proof averting physical access to the sensitive data. And, as smartcards, the code loaded onto security processors can be verified before execution, fulfilling the second property described above. While security processors are more powerful than smartcards and are mostly not limited in the operations they can perform they are nevertheless not suitable for general propose calculation on large data. Secure processors are considerably slower than commodity computing systems and are equipped with less resources, like random access memory (RAM). Furthermore, they are more expensive than standard components by orders of magnitude.

6.1.2 Trusted Execution Environments

Trusted Execution Environments (TEEs) describes a runtime environment which is isolated and allows the protected execution of code. TEEs mainly focus on isolating code and data from other software components of a system. However, due to the high integration degree of modern computer systems they are well protected against physical attacks as well. For instance, if the code and sensitive data are only stored within the boundaries of a system-on-a-chip (SoC) as they are commonly used in mobile devices like smartphones physically accessing the data requires very high effort and specialized equipment. Today, TEEs are mainly available on platforms which are most common in mobile devices, i.e., the ARM architecture.¹ ARM processors can be equipped with a trusted execution environment, called TrustZone.² The predominant processor architecture in the server segment is Intel's x86, which currently does not provide a full-fledged trusted execution environment. This will change in the future when Intel's Software Guard Extension (SGX) will get available. Since TrustZone and SGX are the trusted execution environment solutions with the largest (future) user base we will discuss those two in more detail below. The focus will be primarily on SGX since this technology is likely to become widely deployed in server, and thus in cloud environments.

ARM TrustZone

The TrustZone is a set of security enhancements to chipsets based on the ARM architecture. These enhancements cover the processor, memory and peripherals. With TrustZone, the processor can execute instructions in one of two security modes at any given time, a *normal world* and a *secure world*. A third *monitor mode* facilitates switching between the normal and the secure worlds. The secure and normal worlds have their own address spaces and different privileges. The processor can switch from the normal world to the secure world via an instruction called the secure monitor call (*smc*). When a *smc* instruction is invoked from the normal world, the processor context switches to the secure world (via monitor mode) and freezes execution of the normal world.

¹<http://www.arm.com/>

²<http://www.arm.com/trustzone>

TrustZone can partition memory into two portions, with one portion being exclusively reserved for the secure world. It also allows individual peripherals to be assigned to the secure world. For these peripherals, hardware interrupts are directly routed to and handled by the secure world. While the normal world cannot access peripherals or memory assigned to the secure world, the secure world enjoys unrestricted access to all memory and peripherals on the device. It can therefore access the code and data of the normal world. The secure world can execute arbitrary software, ranging from simple applications to an entire operating system.

A device with ARM TrustZone boots up in the secure world. After the secure world has initialized, it switches to the normal world and boots the operating system there. Most TrustZone-enabled devices are configured to execute a *secure boot* sequence that incorporates cryptographic checks into the secure world boot process [4]. For example, the device vendor could sign the code with its private key, and the vendor's code in the boot ROM would verify this signature using the vendor's public key. These checks ensure that the integrity of the boot-time code in the secure world has not been compromised, e.g., by reflashing the image on persistent storage. Most vendors lock down the secure world via secure boot, thereby ensuring that it cannot be modified by end-users. This feature allows hosts to trust software executing in the secure world and treat it as part of the TCB.

The TrustZone feature fulfil the requirements for trusted execution mentioned above, i.e., (1) isolation of the sensitive data which is achieved through the memory protection applied with TrustZone and (2) control over the executed code which is achieved by the secure boot mechanism. However, since the code executed in the TrustZone is protected by secure boot, and secure boot is controlled by the device vendor usually TrustZone is not available for use by third parties. This leads to the situation that, although TrustZone is a powerful TEE (unlike HSMs) it is not available for use in practice.

Intel Software Guard Extension

The Intel x86 processor architecture is the most used computing platform in servers and in cloud computing. While current generations of x86 processors do not provide a trusted execution environment Intel has announced such a feature for future generations, called Software Guard Extension (SGX) [41]. SGX enables the isolation of (parts) of an application from the remaining system, including privileged software like the operating system or a hypervisor. This isolated part is called *enclave* in Intel's terminology. Since not only the enclave is protected from the system but also the other way around the system is protected from the enclave there is no need for a platform owner to restrict the use of SGX on a system. This will most probably lead to the situation that cloud customers can freely use SGX to create enclaves in cloud environments.

SGX enclave memory is not only protected from access through software on the system but also from physical attacks. Enclave memory which is transferred to the memory modules of a system is encrypted, the enclave memory is only stored in plaintext inside the processor chip. This significantly raises the bar for physical attacks on SGX enclaves. The memory protection provided by SGX thus fulfils the first requirement identified above. Inside an SGX enclave arbitrary code can be executed. However, SGX provides means to attest the code that is executing in an enclave [34, 9]. This feature can be used to develop a scheme which fulfils the second requirement state above, i.e., that the operating performed on sensitive data can be controlled. In short, this can be achieved by providing sensitive data only to an enclave which was loaded with controlled code. A detailed protocol for this scenario is provided subsequently.

6.2 Secure Multi-party Computation with SGX

Subsequently we describe three protocols for SMC with SGX. In the first version the computations are public but the data which are processed are kept secret. In the second version not only the data are kept confidential and the computation are only revealed to the input parties of the computation. In the third version also the input is kept confidential while the computations are only known by one party.

Protocol Version 1. Figure 6.1 shows a protocol for performing secure multi-party computation with SGX. The operations performed are not secret and can be stored in the untrusted cloud. However, the integrity of the code when loaded into the SGX enclave is verified.

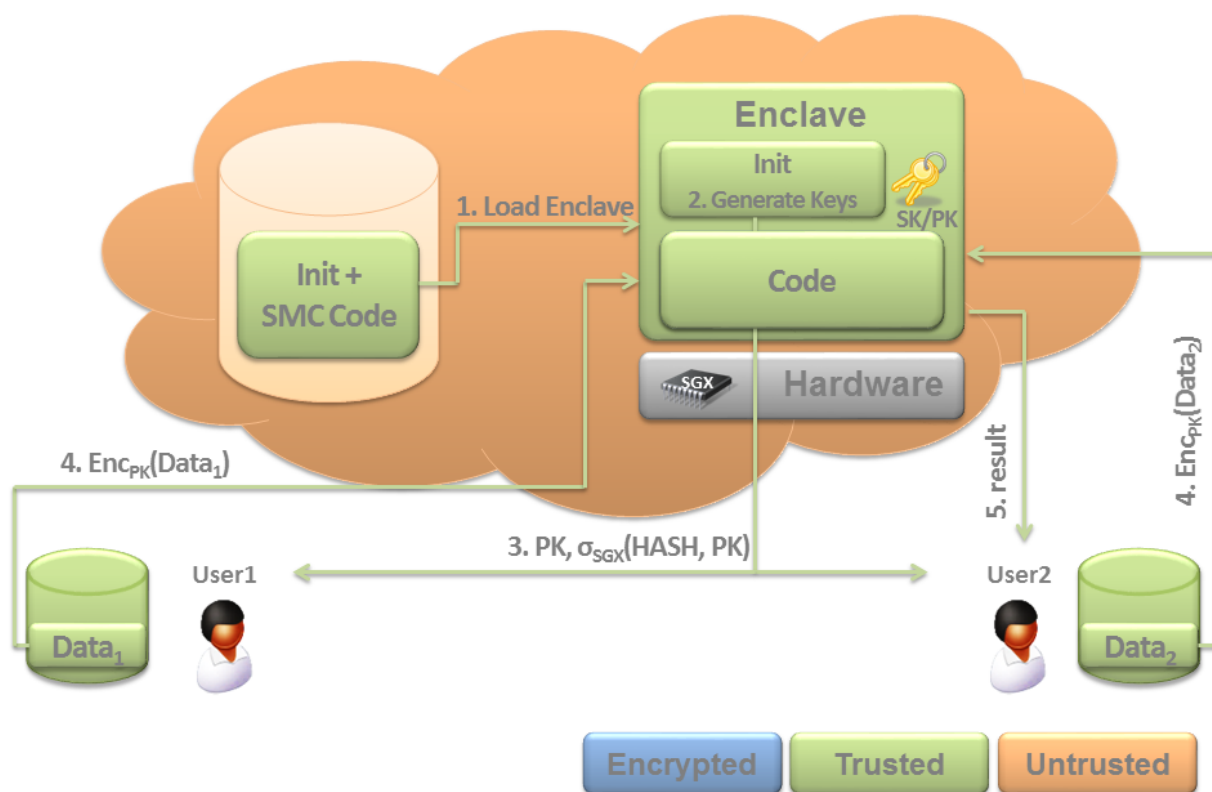


Figure 6.1: SMC protocol with SGX - non-confidential operations

The first step of the protocol is to load the SMC code and some initialization code is loaded into an enclave. During the load process a measurement of the loaded code is made (this measurement will be used in step 3 to prove the integrity of the loaded code to the input parties). The code inside the enclave consists not only of the code which will perform the computations on the private data but also an initialization routine. The second step is the execution of the initialization routine that is responsible for generating a key pair for public key cryptography, e.g., RSA or elliptic curves. In the third step the just generated public key (PK) is transmitted to the input parties. The PK is signed by a platform key which is only used by the trusted hardware to sign data from an enclave. The signature also contains the measurement of the enclave. This way the input parties can verify that indeed the PK comes from an enclave with the correct code loaded. In the fourth step the input parties provide

their inputs to the enclave. For that they encrypt their data with the PK they received and send the data out. Since the key pair was generated inside the enclave the secret key never leaves the enclave, and hence, the input data can only be decrypted inside the enclave. In the last step the computation on the input data is performed and the result is transmitted to the designated parties (in Figure 6.1 the receiving parties is User 2).

Protocol Version 2. The second version of the protocol is shown in Figure 6.2. In contrast to the previous protocol version the code that is executed is confidential and only known to the input parties. To achieve this the code doing the SMC computation is only stored encrypted while outside of the enclave.

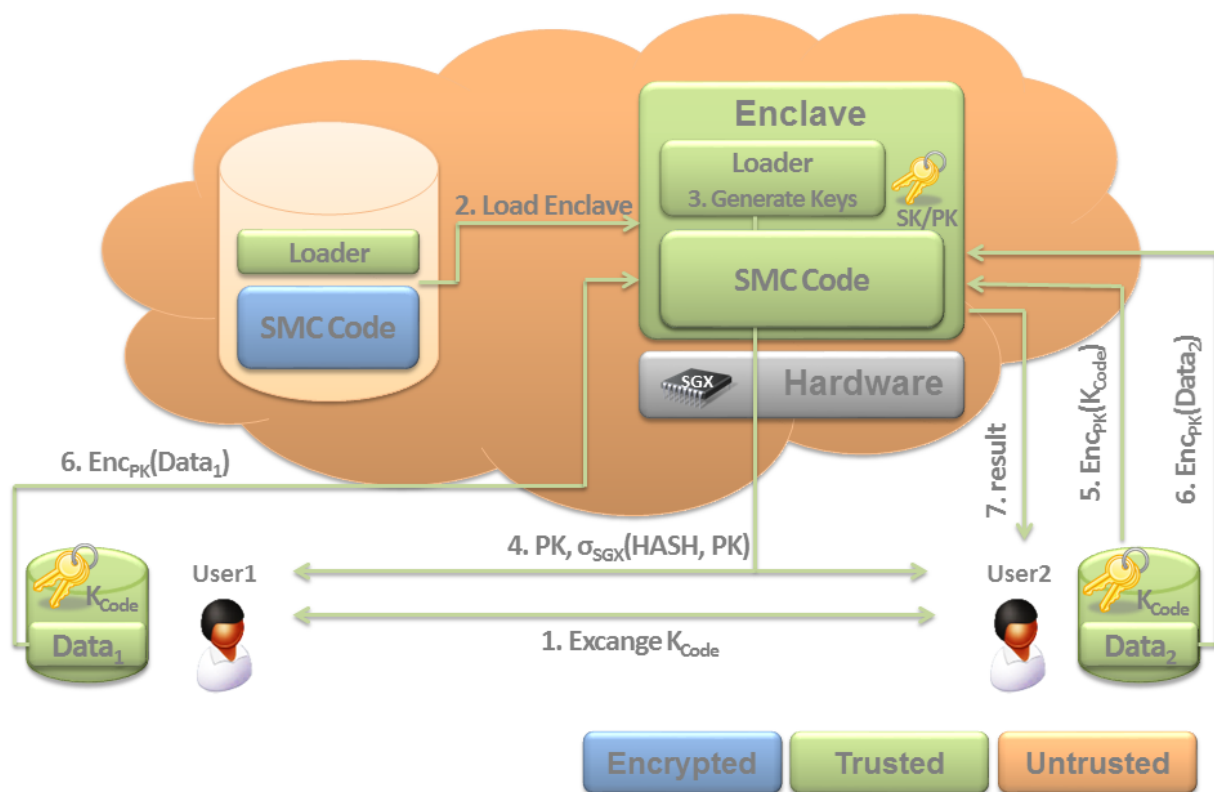


Figure 6.2: SMC protocol with SGX - operations revealed to input parties

The key K_{code} used to encrypt the SMC code is shared between the input parties (step 1 in Figure 6.2). This way all input parties can decrypt the code locally and inspect it, e.g., to verify that the code does not leak any of the inputted data. In the second step the encrypted SMC and a loader component is loaded into the enclave. In the second step a key pair for public key cryptography is generated. Next, in step four the public key (PK) is transferred to the input parties. The input parties use the PK to transfer the K_{code} to the enclave so that the enclave can decrypt the code (step 5). Afterwards, in step six, the input parties provide their data to the enclave, again encrypted by PK. In step seven, the SMC computation is performed and the result is provided to the result party, as in the previous protocol version.

Protocol Version 3. The third version of the protocol concerns the case when the computation performed on the input data should stay confidential, i.e., not all input parties may have access to the SMC code.

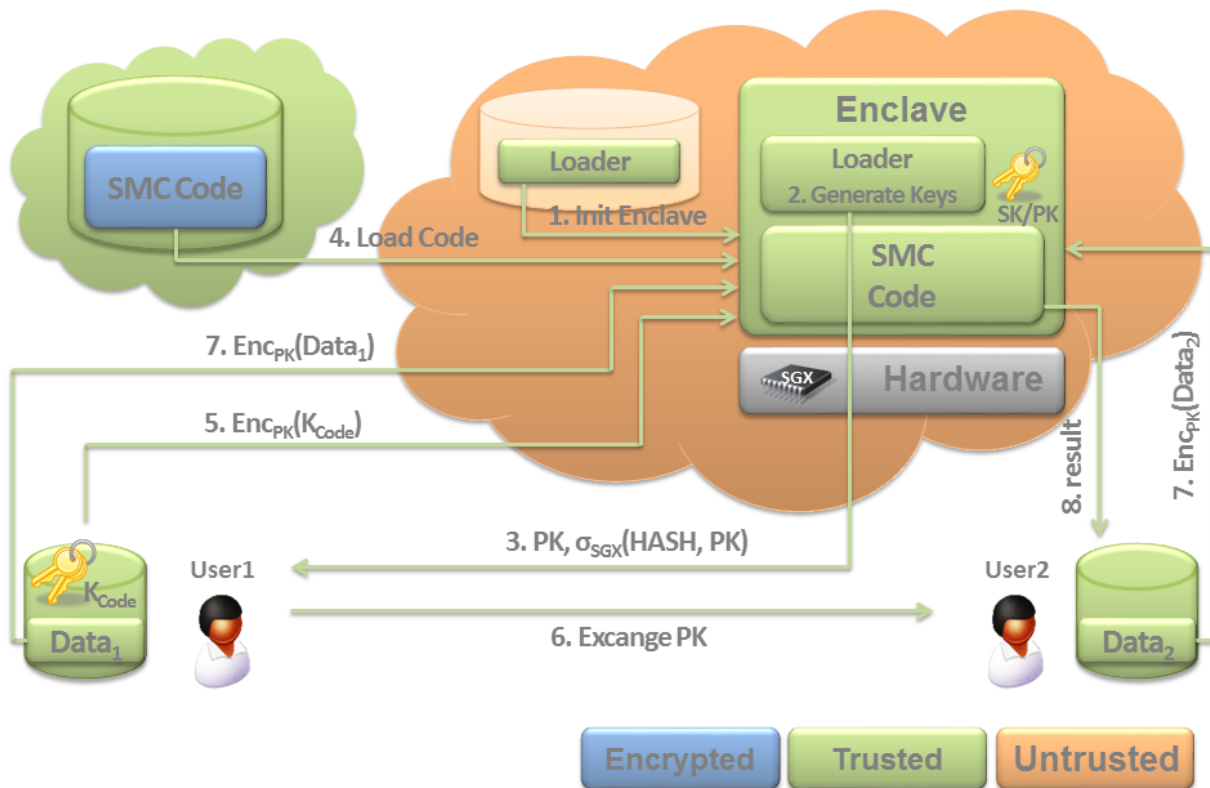


Figure 6.3: SMC protocol with SGX

The third version of the protocol is shown in Figure 6.3. First, as in the previous versions, a loader is used to initialize the enclave. In the second step it generates a public key cryptography key pair and loads the encrypted SMC code and will decrypt it once the enclave receives the appropriate key K_{code} . In step three the PK is transferred to the users (in Figure 6.3 its send to User 1 who forwards PK to User 2). In step four the encrypted code is loaded to the enclave. In this scenario the code is not stored in the public cloud where the computations are performed but in a private cloud of the party providing the code. The key K_{code} to decrypt the code is sent to the enclave in step five. In the Figure the party in possession of K_{code} is one of the input parties, however, it could be also another entity. In step seven the input data are sent to the enclave. As before, the input data is encrypted under the PK generated in the enclave. Finally, in step 8, the computations on the input data is performed and the results are sent to the result party.

6.3 Adversary and Trust Model for SMC with Trusted Hardware

The adversary and trust model when using trusted hardware have to account for the fact that the *trusted hardware* needs to be trusted by definition. This means, that there must be no way to circumvent the protection mechanism provided by the hardware. The protection could be circumvented either by a

hardware fault or modification, or by a software fault or attack. This means, that the manufacturer of the hardware must be trusted with respect to the correct functioning of the device and with respect to not including any malicious modifications to the device, e.g., backdoors.

Furthermore, all software executed in an isolated environment must be trusted. This includes for instance, software libraries, (small) operating systems and other software components which might be executed in a trusted execution environment. For most trusted hardware the code which is executed in isolation is verified or attested by means of cryptographic signatures. This requires that the entity that manages the required keys needs to be trusted to preserve the confidentiality of those keys. In many cases that entity is the hardware manufacturer.

Lastly, also the software performing the secure multi-party computation must be trusted, if it cannot be verified explicitly. A malicious software could simply output all inputs to an adversary.

Chapter 7

Conclusion

This deliverable collects and compares the adversary, trust, communication and system models for multiple real-world application scenarios which greatly benefit from secure computation techniques. As such models may be different for every application a per-application approach was taken in this document. The adversary, trust, communication and system models are specified for every individual application scenario studied in work package WP12 of the project PRACTICE. A summary of the application-specific models is presented in this chapter. It also investigates which characteristics are commonly required in different use cases and what the most important differences are.

The most important and irradiative results of the analyses performed in this work package are summarized at the end of this chapter as final statements.

7.1 Adversary Model

In order to get a comparative overview of the adversary models provided for different application scenarios, the specified models are shown collectively in Table 7.1 below.

The relevant adversary models for each application scenario are highlighted in yellow in order to provide a visual overview of which adversary types are present in different applications. This provides a comparative overview of the nature of participants of distinguished application scenarios when the behaviour of those actors is analysed in adversary model. The presence or the absence of particular adversary models in different application scenarios is a good indicator if there are similarities between different applications. Comparing the adversary model also helps to discover the major differences between the adversary models of the different application scenarios. All similarities and differences are considered to derive a general model that is applicable for all or the majority of application scenarios studied in this work package.

An important feature of this project is to evaluate the role of cloud in the design and implementation of privacy-preserving applications that simultaneously take advantage of secure computation and cloud services. The cloud service provider (CSP) is highlighted in blue in the table below. The colouring enables the reader to quickly compare the expected behaviour of the *cloud* party in different application scenarios based on its adversary model. This helps to draw conclusions and make decisions on the way that the cloud should be seen in each application scenario. The ultimate goal of performing privacy-preserving computations in the cloud can only be achieved if the threats and benefits of using cloud services in the applications of interest are evaluated carefully and described exactly.

Application Scenario	Adversary Model						
<p>Aeroengine Fleet Management</p> <ul style="list-style-type: none"> • P1: Cloud Service Provider \mathcal{C}^l • P2: Airline Companies $\mathcal{I}^k \mathcal{R}^k$ • P3: MRO Service Provider $\mathcal{I}^m \mathcal{R}^m$ • P4: Suppliers $\mathcal{I}^n \mathcal{R}^n$ 	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority
	P1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P1	<input checked="" type="checkbox"/>
	P2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	P2	<input type="checkbox"/>
	P3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	P3	<input type="checkbox"/>
	P4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	P4	<input type="checkbox"/>
<p>Platform for Auctions</p> <ul style="list-style-type: none"> • P1: Cloud service providers \mathcal{C}^k • P2: Auction service provider \mathcal{C} • P3: Buyers one or more – $\mathcal{I}^n \mathcal{R}^m \mathcal{V}^n$ • P4: Sellers one or more – $\mathcal{I}^n \mathcal{R}^n \mathcal{V}^n$ • P5: Competition regulator – \mathcal{V} 	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority
	P1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P1	<input checked="" type="checkbox"/>
	P2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P2	<input type="checkbox"/>
	P3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P3	<input type="checkbox"/>
	P4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P4	<input type="checkbox"/>
	P5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P5	<input type="checkbox"/>
<p>Platform for Benchmarking</p> <ul style="list-style-type: none"> • P1: Cloud service providers \mathcal{C}^k • P2: Benchmarkee (entity to be benchmarked) \mathcal{I} • P3: Data providers (providing data to benchmark against) \mathcal{I}^n • P4: Benchmark recipient \mathcal{R} 	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority
	P1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P1	<input checked="" type="checkbox"/>
	P2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P2	<input type="checkbox"/>
	P3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P3	<input type="checkbox"/>
	P4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	P4	<input type="checkbox"/>

<p>Consortium Gathering Information From its Members</p> <ul style="list-style-type: none"> • $P1$: Consortium board – \mathcal{R}, \mathcal{V} • $P2$: Consortium members – $\mathcal{I}^n, \mathcal{V}^n$, where n is the total number of members • $P3$: Consortium members that execute the SMC for all members – $\mathcal{C}^k, k \leq n$ 	<table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>Party (collectively)</th> <th>honest majority</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (indiv. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>							
Party (indiv. entities)	Party (indiv. entities)				Party (collectively)																																					
	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																				
$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>																																				
$P2$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>																																				
<p>Tax Fraud Detection</p> <ul style="list-style-type: none"> • $P1$: Private companies – \mathcal{I}^n • $P2$: State cloud – \mathcal{C}^k • $P3$: Revenue office – \mathcal{ICR} • $P4$: Referee – \mathcal{V} 	<table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>Party (collectively)</th> <th>honest majority</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (indiv. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>
Party (indiv. entities)	Party (indiv. entities)				Party (collectively)																																					
	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																				
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>																																				
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input checked="" type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input type="checkbox"/>																																				
$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>																																				
<p>Joint Statistical Analysis Between State Entities</p> <ul style="list-style-type: none"> • $P1$: Hosts – \mathcal{C}^m • $P2$: State entities – \mathcal{I}^k • $P3$: Data analyst(s) – \mathcal{R}, \mathcal{V} • $P4$: Referee – \mathcal{V} 	<table border="1"> <thead> <tr> <th rowspan="2">Party (indiv. entities)</th> <th colspan="4">Party (indiv. entities)</th> <th colspan="2">Party (collectively)</th> </tr> <tr> <th>trusted</th> <th>semi-honest</th> <th>covert</th> <th>malicious</th> <th>Party (collectively)</th> <th>honest majority</th> </tr> </thead> <tbody> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Party (indiv. entities)	Party (indiv. entities)				Party (collectively)		trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>	$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>
Party (indiv. entities)	Party (indiv. entities)				Party (collectively)																																					
	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																				
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																				
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																																				
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>																																				
$P4$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P4$	<input type="checkbox"/>																																				

<p>Privacy Preserving Genome-Wide Association Studies Between Biobanks</p> <ul style="list-style-type: none"> • $P1$: Biobanks – \mathcal{ICR}^k • $P2$: Referee or state entity – \mathcal{V} 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																					
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																					
$P1$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P1$	<input type="checkbox"/>																																					
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																																					
<p>Privacy Preserving Personal Genome Analyses and Studies</p> <ul style="list-style-type: none"> • $P1$: Laboratory – \mathcal{C} • $P2$: Donor(s) – \mathcal{I}^k • $P3$: State entities – \mathcal{C}^m • $P4$: Data analyst(s) – $\mathcal{R}^n, \mathcal{V}^n$ • $P5$: Referee or state entity – \mathcal{V} 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P2$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P3$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P4$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P4$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P5$</td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>$P5$</td> <td><input type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>	$P5$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P5$	<input type="checkbox"/>
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																					
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input type="checkbox"/>																																					
$P2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P2$	<input type="checkbox"/>																																					
$P3$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P3$	<input checked="" type="checkbox"/>																																					
$P4$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P4$	<input type="checkbox"/>																																					
$P5$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$P5$	<input type="checkbox"/>																																					
<p>Platform for Surveys on Sensitive Data</p> <ul style="list-style-type: none"> • $P1$: Cloud service provider – \mathcal{C}^m • $P2a$: Survey creator – \mathcal{I} • $P2b$: Survey evaluator – \mathcal{R} • $P3$: Survey participant(s) – \mathcal{I}^k 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td>$P1$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P1$</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>$P2a$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2a$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P2b$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P2b$</td> <td><input type="checkbox"/></td> </tr> <tr> <td>$P3$</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>$P3$</td> <td><input type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>	$P2b$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2b$	<input type="checkbox"/>	$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>							
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																																					
$P1$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P1$	<input checked="" type="checkbox"/>																																					
$P2a$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2a$	<input type="checkbox"/>																																					
$P2b$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P2b$	<input type="checkbox"/>																																					
$P3$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	$P3$	<input type="checkbox"/>																																					

<p>Location Sharing with Nearby Contacts</p> <ul style="list-style-type: none"> • <i>P1</i>: Users (reveals her location; potentially learning other participants location) – IRC^k 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td><i>P1</i></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><i>P1</i></td> <td><input checked="" type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	<i>P1</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P1</i>	<input checked="" type="checkbox"/>														
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																							
<i>P1</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P1</i>	<input checked="" type="checkbox"/>																							
<p>Privacy Preserving Satellite Collision Detection</p> <ul style="list-style-type: none"> • <i>P1</i>: Hosts – ICR^m • <i>P2</i>: Satellite operators – IR^n, \mathcal{V}^n • <i>P3</i>: Authority – \mathcal{V} 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td><i>P1</i></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><i>P1</i></td> <td><input type="checkbox"/></td> </tr> <tr> <td><i>P2</i></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><i>P2</i></td> <td><input type="checkbox"/></td> </tr> <tr> <td><i>P3</i></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><i>P3</i></td> <td><input type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	<i>P1</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P1</i>	<input type="checkbox"/>	<i>P2</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P2</i>	<input type="checkbox"/>	<i>P3</i>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<i>P3</i>	<input type="checkbox"/>
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																							
<i>P1</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P1</i>	<input type="checkbox"/>																							
<i>P2</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P2</i>	<input type="checkbox"/>																							
<i>P3</i>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<i>P3</i>	<input type="checkbox"/>																							
<p>Key Management</p> <ul style="list-style-type: none"> • <i>P1</i>: User ICR • <i>P2</i>: Cloud service providers SS^n 	<table border="0"> <tr> <td>Party (indiv. entities)</td> <td>trusted</td> <td>semi-honest</td> <td>covert</td> <td>malicious</td> <td>Party (collectively)</td> <td>honest majority</td> </tr> <tr> <td><i>P1</i></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><i>P1</i></td> <td><input type="checkbox"/></td> </tr> <tr> <td><i>P2</i></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><i>P2</i></td> <td><input checked="" type="checkbox"/></td> </tr> </table>	Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority	<i>P1</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<i>P1</i>	<input type="checkbox"/>	<i>P2</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P2</i>	<input checked="" type="checkbox"/>							
Party (indiv. entities)	trusted	semi-honest	covert	malicious	Party (collectively)	honest majority																							
<i>P1</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<i>P1</i>	<input type="checkbox"/>																							
<i>P2</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<i>P2</i>	<input checked="" type="checkbox"/>																							

Mobile Data Sharing	Party (indiv. entities)				Party (collectively)	
	trusted	semi-honest	covert	malicious	honest majority	
<ul style="list-style-type: none"> • <i>P1a</i>: Cloud storage service provider (source storage) <i>SS</i> • <i>P1b</i>: Cloud storage service provider (destination storage) <i>SS</i> • <i>P2</i>: Sender <i>IC</i> • <i>P3</i>: Receiver <i>RC</i> • <i>P4</i>: Cloud (storage) service provider (intermediate storage) <i>C</i> 	<i>P1a</i>	<i>P1b</i>	<i>P2</i>	<i>P3</i>	<i>P4</i>	
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Table 7.1: Comparative List of Adversary Models

From to the adversary models of the individual application scenarios listed in Table 7.1, the following conclusions can be drawn:

- *Trusted* parties are not present in any of the analysed application scenarios, except for one. The one exception is the scenario *Key Management* in which the trusted party is the user of the system who of course trusts himself.
- *Semi-honest* behavior of participants is very common and can be observed conventionally in the majority of use case scenarios evaluated in this document.
- At least one participant can be considered *malicious* in every scenario except for one. This means that one party can be fully mistrusted. Usually, the mistrusted party is the cloud service provider (CSP). In some scenarios there is no CSP included, however, in those scenarios the party that can be considered malicious can easily be outsourced to the cloud. The only exception where no malicious party is present is the scenario *Privacy Preserving Genome-Wide Association Studies Between Biobanks*. In this scenario different biobanks *collaborate* in order to perform a joint genome-wide association study. They use each other’s data mutually so as to get more accurate results. Therefore, none of the biobanks acts in a malicious way. Rather, they are all semi-honest adversaries that follow the protocol correctly and just attempt to learn additional information.
- *Malicious* or *covert* behaviour is an attribute of individual participants. However, in many cases not the behaviour of an individual participant is relevant but rather the joint behavior of all individuals of one party. The security properties of the protocol hold if an *honest majority* can be assumed.
- Cloud service provider (CSP) is assumed to be *malicious* in all use case scenarios except for the *Tax Fraud Detection* scenario, in which the state cloud is used. Considering the cloud service provider as a malicious actor is perfectly in line with the objectives of the project, which aims at providing solutions for privacy-preserving computation in the *malicious* cloud.

7.2 Trust Model

The comparison of the scenario specific trust models shows that using *trusted hardware* provides a good means to enhance the security and perform of some actions in an easier and straightforward way. We elaborated on the possibilities of performing secure multi-party computation with trusted

hardware in Chapter 6. In some cases trusted hardware can lower the level of trust devoted to one or multiple participant; using trusted hardware enhances the trust in the correctness of the performed operations and computations. Table 7.2 lists the application scenarios in which use of trusted hardware can be beneficial in this sense. It summarizes the types of trusted hardware which can be used in these scenarios. Further, the purpose or usage of the trusted hardware components in the applications scenarios is explained in Table 7.2.

Application Scenario	Trusted Hardware Type	Usage
Platform for Auctions	Intel SGX	Use on the cloud service provider
Consortium Gathering Information From its Members	Intel SGX	Intel SGX can be used to enhance the level of trust given to computing parties that execute SMC protocols.
Tax Fraud Detection	Intel TXT, Intel SGX	Use in state cloud servers to enhance the trust in the correctness of the performed computations.
Joint Statistical Analysis Between State Entities	Intel SGX	Provide assurance as to the correctness of the computations on the servers (or hosts) by execution of (integrity-checked) code in isolated environments.
Mobile Data Sharing	Mobile TEE such as ARM TrustZone	Perform critical operations such as key management inside a TEE.

Table 7.2: Application of Trusted Hardware

Intel's Software Guard Extensions (Intel SGX) is a very comprehensive trusted hardware solution providing isolated enclaves, i.e., trusted execution environment (TEE), and attestation facilities [41, 34, 9]. This is why it was chosen as potential trusted hardware for use in multiple application scenarios. Particularly, on cloud servers using trusted hardware such as Intel SGX enhances the trust in the correctness of computations performed in the cloud (compare Chapter 7.2).

Hardware Security Modules (HSMs), especially smartcards, are a kind of trusted hardware with extremely limited capabilities. These limitations cause smartcards not to scale appropriately for use in the servers. Therefore, smartcards are not appropriate when a trusted hardware device needs to be selected for application in the use case scenarios in question.

Similarly, secure processors, which are another common type of HSMs, are limited in their performance and capabilities. Furthermore, secure processors are expensive given the limited functionality they provide. Hence, applicability of HSMs on the protocols used in the investigated use cases is questionable and thus not easily given.

The usage of trusted execution environments (TEE) on mobile devices can also provide a higher level of trust in the confidential and correct execution of computations and operations. In contrast to TEEs in desktop or server environments hardware security features on mobile phones are widely available today. In particular, the ARM TrustZone feature is available on almost every smartphone

sold today [4]. However, access to the security features are usually limited by the device vendor, hence, solutions building on top of TrustZone need cooperation from the device vendor.

7.3 Communication Model

The analysis of the communication models of the different use cases shows that for most use cases it is necessary for the participants to be *online* during the operations and computations. Especially the cloud service provider is assumed to be always online. The users count on the permanent availability and responsiveness of cloud services which is a reasonable assumption given that it is in line with the CSP's business model.

Asynchronous communication is possible in several application scenarios. Even when being online during the communication is required, data may be communicated asynchronously. For example, different input parties are allowed to provide their input at different times without caring about other parties. This enables individual (input) parties to be offline most of the time and send their data (input) at arbitrary points in time.

The scenario specific models indicate that most of the communication channels established between the parties must be authenticated, confidential, integrity-protected and provide freshness. However, this can be achieved with standard methods such as Transport Layer Security (TLS). It is important to note that authenticity of the communications is an important requirement required for the majority of the application scenarios.

Network bandwidth is shown to be an important issue in a limited number of use case scenarios which require the exchange of huge amounts of data. However, for most applications high-bandwidth network connections are not required, since the volume of input/output data is low.

Network delays are tolerated by the majority of application scenarios. However, the network connection must be reliable since the protocols cannot operate on incomplete data. Standard network protocols like Transmission Control Protocol (TCP) can be used to transfer data reliably.

Only two use cases – *Platform for Surveys on Sensitive Data* and *Mobile Data Sharing* – require the data transfers to be transactional. In the former scenario, input data must be transferred in a transactional way and in the latter scenario, the user files are supposed to be transferred either completely or not at all.

7.4 System Model

The analysis of the system models of the individual application scenarios shows that high computational power (CPU) and a considerably large amount of system memory is required in a few scenarios. Not all participants in these scenarios need to provide these resources, usually the party performing the secure multi-party computations need to have sufficient resources to be responsive, i.e., finish the computations within an appropriate time frame. These requirements can be met through computations in distributed environments, e.g., by using powerful and scalable cloud services. Two scenarios claim to be suitable for taking benefit from parallelism of computations. This small number of scenarios is normal, as computations usually need to be adapted to benefit from parallelism and the adaptation usually requires significant implementation effort.

The computational limitations of smartphones were found to not be restrictive for any of the investigated application scenarios. This is for two reasons:

- (1) In the majority of scenarios smartphones are not used by any of the participating parties.
- (2) In the cases where some parties are using smartphones, all computationally expensive operations of the protocol are performed by the other (not resource constrained) parties of the scenario, e.g., the

cloud service provider.

A few application scenarios require a high network bandwidth for exchanging data, because a large amount of data is transmitted in those scenarios. While two use case scenarios declare explicitly that a low-bandwidth communication channel is adequate, the remaining use case scenarios leave this to the size of the data being communicated. For large input/output data the bandwidth should be high and for small data lower bandwidths are acceptable without affecting the responsiveness of the application in a negative way. The bandwidth requirements for the application scenario *Joint Statistical Analysis Between State Entities* can be relaxed, if data is uploaded asynchronously. The analyses of different use case scenarios indicate that regardless of the network bandwidth, the connection must be reliable for the system to function properly and achieve the defined goals and get the results in an acceptable amount of time.

7.5 Guarantees

All application scenarios require the correct execution of the algorithms and functions, regardless of the location of the computations, i.e., it does not matter, if own computing resources and devices are used or third party computing resources like cloud services are used. However, only for some use case scenarios the operations need to be explicitly verifiable. Hence, most of the use cases neither require *universal verifiability* nor *designated verifiability*.

The confidentiality of cryptographic keys and their proper delivery to designated receivers is an important goal which must be guaranteed in all application scenarios investigated in this document. The cloud service provider must be prevented from accessing any confidential data of the participating parties, which is mostly achieved by avoiding the transfer of data to the cloud such that a single cloud provider can retrieve sensitive information from it.

The confidentiality of input data must be guaranteed in almost all use case scenarios. It means that the input providing parties want to keep their data secret while it is processed. The input data must be confidential against the host which executes an algorithm or a function on the data as well as other parties that provide input. Therefore, if there are several participants in a use case scenario which provide input to the algorithm, mutual confidentiality of data must be also met in addition to the confidentiality in relation with the host.

Contrary to the confidentiality of input, which is required by all application scenarios, the need for anonymity of input was mentioned explicitly for only two applications: *Consortium Gathering Information From its Members* and *Privacy Preserving Genome-Wide Association Studies Between Biobanks*.

Similar to the input data, the confidentiality of output data (results) needs to be guaranteed as well. Only the designated receiver/s of the output is/are supposed to get the results and no other participant is allowed to receive the results of the computations.

The correctness of the computations and the generated results is another security goal which must be achieved in all analysed scenarios. The analyses performed on different use case scenarios indicate that the involvement of cloud services is irrelevant in this case. It means that although achieving correctness gets more difficult when the computations are outsourced to the cloud, this requirement remains intact for all application scenarios evaluated in this deliverable.

Nearly half of the analysed use cases require the results of the computations to be verifiable by designated parties. Universal verifiability was not declared as a requirement in the application scenarios studied in this deliverable. Designated verifiability means that the correctness of the results must be verifiable only by a specific party, which is normally the designated receiver of the output.

7.6 Final Statements

In this very last section of this deliverable some concluding statements are made which summarize the final results of the analyses accomplished in this work package. The activities performed in this work package include the following:

- Identification of several application scenarios which greatly benefit from secure computation.
- Specification of the adversary, trust, communication and system models for each of the identified application scenarios.
- Analysis of the feasibility of implementing investigated use case scenarios using different secure computation engines that are part of the project's overall architecture.

Based on the analysis of the individual applications scenarios the most important findings are listed below.

- The security models are diverse, i.e., no general uniform model can be identified.
- The cloud can always be assumed to be malicious or semi-honest.
- Each application scenario can be implemented using at least one of the secure computation engines from the PRACTICE architecture.
- No secure computation engine exists which efficiently can implement all studied application scenarios.
- Network connections must be reliable regardless of the fact if high or low bandwidth is required. Unreliable network connections will lead to improperly functioning systems and thus do not achieve the desired goals.
- The cloud service provider (CSP) is expected to be always online and responsive. This is indeed one of the reasons for using cloud services instead of own devices and servers which may fail to respond permanently due to lack of resources or maintenance issues. The strong infrastructure of CSPs make strong assumption on the quality of services provided by the CSPs reasonable.
- Trusted hardware can be used to perform secure multi-party computation. When combining trusted hardware and secure computation protocols stronger adversary models can be assumed, and stronger guarantees on the correctness and verifiability of the computations can be achieved.

As expected, the most desired impact of using cloud and distributed computing is acceleration of executions and reliability of services (e.g., network bandwidth) due to the powerful infrastructure of the cloud. This means that the utilization of cloud services is a great benefit in the application scenarios evaluated in this document while not risking the privacy of the processed data even if the CSP is considered malicious.

In a nutshell it can be concluded that there is no unique and standard model which can cover all different application scenarios. The adversary, trust, communication and system models specified for different application scenarios were analysed in this deliverable. This analysis showed that despite similarities between the security models of different scenarios with varying number of participants from miscellaneous business areas, one single model cannot meet the requirements of all applications. Therefore, individual (per-application) identification and specification of such models cannot be avoided when one or another of the studied application scenarios is implemented.

Similarly, there is no single secure computation engine/tool which can implement all use case scenarios. The secure computation engines that are considered for the final implementations in the PRACTICE project provide different functionalities and advantages. Each tool is suitable for implementing at least one of the application scenarios in question. Based on the number of participants and the adversary model of a specific use case scenario, one secure computation engine may or may not be appropriate for the implementation of that scenario. This shows the importance of exact and thorough security models. In some cases, use of a particular tool for implementing some scenarios is possible but not expedient. Factors such as performance, number of participants, presence or absence of colluding parties, etc. determine whether the use of a tool to implement a use case scenario is suitable or not.

Chapter 8

List of Abbreviations

CPU	Central Processing Unit
CSP	Cloud Service Provider
DAGGER	Distributed Aggregation and Security Services
EC	European Commission
HSM	Hardware Security Module
IA	Intel Architecture
MPC	Multi-Party Computation
MRO	Maintenance, Repair, and Overhaul
PaaS	Platform as a Service
PK	public key
RAM	Random Access Memory
SGX	Software Guard Extensions
smc	Secure Monitor Call
SMC	Secure Multi-party Computation
SoC	System on a Chip
SPEAR	Secure Platform for Enterprise Applications and Services
SVM	Secure Virtual Machine
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPM	Trusted Platform Module
TTP	Trusted Third Party
TXT	Trusted Execution Technology
WP	Work Package

Bibliography

- [1] SAP HANA. <http://hana.sap.com/>.
- [2] Sharemind. <https://sharemind.cyber.ee/>.
- [3] VIFF, the Virtual Ideal Functionality Framework. <http://viff.dk/>.
- [4] ARM security technology – Building a secure system using TrustZone technology, 2009. ARM Technical Whitepaper. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.
- [5] 23andMe Project. <https://www.23andme.com/>.
- [6] Tiago Alves and Don Felton. TrustZone: Integrated Hardware and Software Security. *Information Quaterly*, 3(4), 2004.
- [7] AMD. AMD64 Virtualization Codenamed “Pacifica” Technology — Secure Virtual Machine Architecture Reference Manual. Technical Report Publication Number 33047, Revision 3.01, AMD, May 2005.
- [8] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, page 10, 2013.
- [9] Ittai Anati, Shay Gueron, Simon P. Johnson, and Vincent R. Scarlata. Innovative technology for CPU based attestation and sealing. In *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.
- [10] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography*, pages 137–156. Springer, 2007.
- [11] Jerome Azema and Gilles Fayad. M-Shield™ Mobile Security Technology: making wireless secure. Technical report, Texas Instruments, 2008.
- [12] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly Auditable Secure Multi-Party Computation. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 175–196, 2014.
- [13] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

- [14] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08*, pages 257–266, New York, NY, USA, 2008. ACM.
- [15] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, LNCS. Springer, 2015. To appear.
- [16] Dan Bogdanov, Liina Kamm, Sven Laur, and Pille Pruulmann-Vengerfeldt. Secure multi-party data analysis: end user validation and practical experiments. Cryptology ePrint Archive, Report 2013/826, 2013.
- [17] Dan Bogdanov, Liina Kamm, Sven Laur, Pille Pruulmann-Vengerfeldt, Riivo Talviste, and Jan Willemson. Privacy-preserving statistical data analysis on federated databases. In *Proceedings of the Annual Privacy Forum. APF'14*, LNCS. Springer, 2014.
- [18] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis (short paper). In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security. FC'12*, pages 57–64, 2012.
- [19] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure Multiparty Computation Goes Live. In Dingledine and Golle [27], pages 325–343.
- [20] Ran Canetti. Security and Composition of Multi-party Cryptographic Protocols. *Journal of Cryptology*, 13:2000, 1998.
- [21] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. *IACR Cryptology ePrint Archive*, 2000:67, 2000.
- [22] Intel Corporation. Software Guard Extensions Programming Reference, September 2013.
- [23] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300. Springer Berlin Heidelberg, 2001.
- [24] Sebastiaan de Hoogh. *Design of large scale applications of secure multiparty computation: secure linear programming*. PhD thesis, Eindhoven University of Technology, 2012.
- [25] Sebastiaan de Hoogh, Berry Schoenmakers, and Meilof Veeningen. Universally Verifiable Outsourcing and Application to Linear Programming. To be published as a book chapter, 2015.
- [26] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.
- [27] Roger Dingledine and Philippe Golle, editors. *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*. Springer, 2009.

- [28] Cynthia Dwork. The Differential Privacy Frontier (Extended Abstract). In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 496–502. Springer, 2009.
- [29] Yael Eijgenberg, Moriya Farbstein, Meital Levy, and Yehuda Lindell. SCAPI: The Secure Computation Application Programming Interface. *IACR Cryptology ePrint Archive*, 2012:629, 2012.
- [30] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Proceedings of the 30th Annual Conference on Advances in Cryptology, CRYPTO'10*, pages 465–482, Berlin, Heidelberg, 2010. Springer-Verlag.
- [31] Shafi Goldwasser and Yael Tauman Kalai. On the (In)security of the Fiat-Shamir Paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 102–113, 2003.
- [32] Georg Hafner, Mario Münzer, Ferdinand Brasser, Janus Dam Nielsen, Peter Sebastian Norholdt, Dan Bogdanov, Riivo Talviste, Liina Kamm, Marko J oemets, Meilof Veenigen, Niels de Vreede, Antonio Zilli, and Kurt Nielsen. PRACTICE Deliverable D12.1: Application Scenarios and their Requirements, 2013. Available from <http://www.practice-project.eu>.
- [33] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, page 11. ACM, 2013.
- [34] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.
- [35] Dennis Hofheinz and Victor Shoup. GNUC: A New Universal Composability Framework. *Cryptology ePrint Archive*, Report 2011/303, 2011. <http://eprint.iacr.org/>.
- [36] Intel Corporation. Intel Trusted Execution Technology MLE Developer's Guide. Technical Report Document Number: 315168-006, Intel Corporation, December 2009.
- [37] Liina Kamm, Dan Bogdanov, Sven Laur, and Jaak Vilo. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, 29(7):886–893, 2013.
- [38] Liina Kamm and Jan Willemsen. Secure Floating-Point Arithmetic and Private Satellite Collision Analysis. *Cryptology ePrint Archive*, Report 2013/850, 2013. <http://eprint.iacr.org/>.
- [39] Florian Kerschbaum, A Schröpfer, Antonio Zilli, Richard Pibernik, Octavian Catrina, Sebastiaan de Hoogh, Berry Schoenmakers, Stelvio Cimato, and Ernesto Damiani. Secure collaborative supply-chain management. *Computer*, 44(9):38–43, 2011.
- [40] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated

- execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–1. ACM, 2013.
- [41] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative instructions and software model for isolated execution. In *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.
- [42] Janus Dam Nielsen, Jakob Illeborg Pagter, and Michael Bladt Stausholm. Location privacy via actively secure private proximity testing. In *PerCom Workshops*, pages 381–386. IEEE, 2012.
- [43] Claudio Orlandi, Benny Pinkas, Bogdan Warinschi, Dan Bogdanov, Thomas Schneider, Michael Zohner, Meilof Veeningen, and Niels de Vreede. PRACTICE Deliverable D11.1: A Theoretical Evaluation of the Existing Secure Computation Solutions, 2014. Available from <http://www.practice-project.eu>.
- [44] Berry Schoenmakers and Meilof Veeningen. Universally Verifiable Multiparty Computation from Threshold Homomorphic Cryptosystems. Cryptology ePrint Archive, Report 2015/058, 2015. Accepted at ACNS '15.
- [45] Axel Schröpfer, Florian Kerschbaum, Debmalya Biswas, Steffen Geißinger, and Christoph Schütz. L1-Faster Development and Benchmarking of Cryptographic Protocols. In *ECRYPT Workshop on Software Performance Enhancements for Encryption and Decryption and Cryptographic Compilers (SPEED-CC'09)*, pages 12–13, 2009.
- [46] Axel Schropfer, Florian Kerschbaum, and Gunter Muller. L1-an intermediate language for mixed-protocol secure computation. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 298–307. IEEE, 2011.
- [47] Riivo Talviste. Deploying secure multiparty computation for joint data analysis—a case study. Master’s thesis, Institute of Computer Science, University of Tartu, 2011.
- [48] Tomas Toft. Solving Linear Programs Using Multiparty Computation. In Dingledine and Golle [27], pages 90–107.
- [49] Trusted Computing Group. TPM Main Specification, Version 1.2, Revision 103, July 2007.
- [50] Dominique Unruh. Random Oracles and Auxiliary Input. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 205–223, 2007.
- [51] Hoeteck Wee. Zero Knowledge in the Random Oracle Model, Revisited. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 417–434, 2009.