This is a pre-print. The final version appeared in Proc. of the Workshop on Security, Privacy, and Identity Management in the Cloud to be held at the 11th International Conference on Availability, Reliability and Security (ARES SECPID 2016), Conference Publishing Services (CPS), 2016.

PRISMACLOUD Tools: A Cryptographic Toolbox for Increasing Security in Cloud Services

Thomas Lorünser*, Daniel Slamanig†, Thomas Länger‡, Henrich C. Pöhls§
*AIT Austrian Institute of Technology GmbH, Digital Safety & Security, Austria
thomas.loruenser@ait.ac.at
† IAIK, Graz University of Technology, Austria
daniel.slamanig@tugraz.at
‡University of Lausanne, Switzerland
thomas.laenger@unil.ch
§Institute of IT-Security and Security Law & Chair of IT-Security, University of Passau, Germany
hp@sec.uni-passau.de

***Abstract*—The EC Horizon 2020 project PRISMACLOUD aims at cryptographically addressing several severe risks threatening end user security and privacy in current cloud settings. This shall be achieved by the provision of a reusable toolbox encapsulating cryptographic functionality from which dependably secure cloud services can be assembled. In order to provide a tangible abstraction of the complexity involved with the construction of cryptographically secured cloud services, we introduce the four-layer PRISMACLOUD architecture. Top down, it consists of a use cases (application) layer, a services layer, a tools layer, and a cryptographic primitives and protocols layer. In this paper we provide a detailed description of the PRISMACLOUD tools in terms of functional components, as well as how they interact to provide the desired security functionality. We also briefly describe the cutting-edge cryptographic primitives which are encompassed by the tools. Both the toolbox and the cryptographic primitives and protocols are being currently developed and will be provided as reference implementation by project end in July 2018.**

***Keywords*-Secure cloud computing, cryptography, privacy, information theoretic security, usability, privacy by design**

I. INTRODUCTION

Cloud computing has the potential to dramatically reduce the cost and complexity of provisioning information technology resources for end users – it lets businesses, administrations, or individual end users dispose of virtualised computing and storage services, without having to procure the infrastructures themselves, and without prior commitment to an amount of storage and computation required. Nevertheless, many end users with special security requirements are reluctant to move from private infrastructures to the cloud, because several existing risks threaten end user security and privacy. However, opting out will not be an option in the long term for businesses, because of the economic handicap they will face and competitive advantages they will lose on the market.

With an estimated size of 150 billion USD, cloud computing is the major growth area in the field of Information and Communication Technologies [1] [2] and, consequently, it is a highly contested market, in which the USA have a commanding lead in the fields of producers of cloud systems and cloud providers. As cloud computing is still on a comparatively early stage, especially as regards the provisioning of secure cloud services, the European Commission sees opportunities for European industries, as well as for European customers. It seeks to promote the commercialisation and exploitation by implementing a European Cloud Computing Strategy [3] and by devoting considerable funding to the Horizon 2020 Framework Programme for Research and Technical Development.

The Horizon 2020 research activity PRISMACLOUD[1] uses a privacy-by-design approach [5] [6] to cryptographically address several security and privacy issues, prevailing in currently available cloud offerings. The project will provide a *toolbox* whose configurable and parametrizable tools completely encapsulate several *cryptographic primitives and protocols*. The tools can subsequently be used to construct *secure, privacy preserving services* to be deployed in the cloud. These services can be accessed by *end-user applications* to yield the desired security functionalities. By project end *eight example services* shall be validated in *three pilots* in the fields of e-Health, e-Government and Smart Cities. The cloud services shall be ready for *commercial exploitation* by the industry partners of the PRISMACLOUD consortium and beyond shortly after project end. The consortium itself consists of 16 organisations with specific expertise in required fields, including cloud providers and end-users, as well as universities in non-technical orthogonal function (for usability issues, legal issues; for standardisation, business deployment, cybercrime research)

In Section II we give a high level description of the single layers of the proposed architecture (applications layer, services layer, tools layer, and primitives layer). Then, Section III details the *eight example services* which will be constructed from the PRISMACLOUD toolbox. Section IV presents a detailed description of the *tools*: we describe their functional components and how they interact, and reference the *cryptographic primitives and protocols* which are encapsulated to achieve the desired security functionalities. The cryptographic primitives

[1]This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 644962 PRISMACLOUD: "Privacy and security maintaining services in the cloud" [4]; duration 2/2015-7/2018; 16 partners; https://www.prismacloud.eu

This is a pre-print. The final version appears in the proceedings of ARES (SECPID) 2016.

and protocols themselves are described in Section V. Finally, a short summary and a conclusion is given in Section VI.

II. Architecture

In order to tackle and organize the complexity involved with the construction of the cryptographically secured services, we introduce the conceptual model of the layered PRISMACLOUD *architecture*, which is organized in 4 tiers (cf. Figure 1). These layers of abstraction also define connection points between the different disciplines involved: cryptographers, software engineers/developers and cloud service architects. On the uppermost layer are the end user applications, represented in the project by the three selected use cases from the *(i) Use Cases layer*. This layer uses the cloud services of the *(ii) Services layer* to achieve the desired security functionalities. These cloud services are a representative selection of possible services which can be built from the tools organized in the *(iii) Tools layer*. In particular, they represent a way to deliver the tools to service developers and cloud architects in an accessible and scalable way. Together, the tools constitute the PRISMACLOUD toolbox. Each tool encapsulates the needed cryptographic primitives and protocols from the *(iv) Primitives layer*, which is the lowest layer of the PRISMACLOUD architecture. In the following we shortly review the layers and the rationale behind the structure beginning from the bottom.

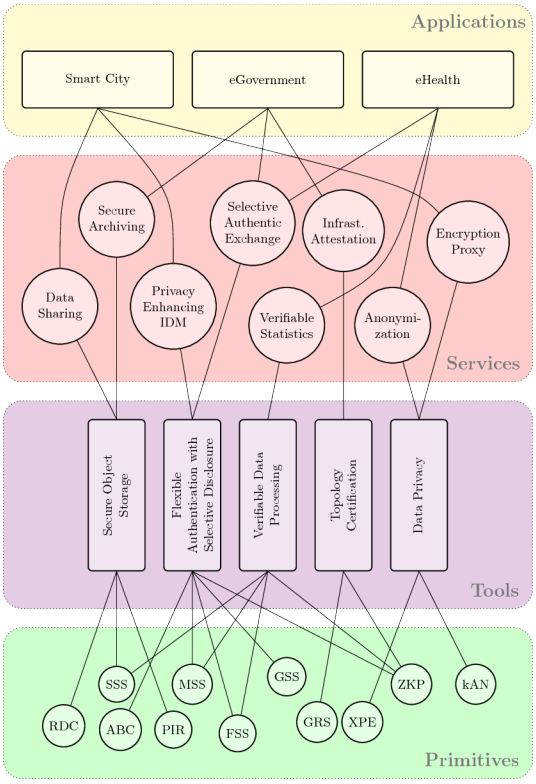

Fig. 1: The PRISMACLOUD Architecture

(Primitives abbreviations: RDC: Remote Data Checking; SSS: Secret Sharing Schemes; ABC: Attribute-Based Credentials; PIR: Private Information Retrieval; MSS: Malleable Signature Schemes; FSS: Functional Signature Schemes; GSS: Group Signature Schemes; GRS: Graph Signature Schemes; XPE: Format- and Order-Preserving Encryption; ZKP: Zero-Knowledge Proofs; kAN: k-Anonymity)

Primitives. At the lowest layer we have *cryptographic primitives and protocols* which represent basic cryptographic primitives, such as signature schemes or cryptographic protocols, such as secret sharing. This is the layer where the project consortium conducts cryptographic research and will advance the state-of-the-art.

Major *outcome* in this layer will be cryptographic primitives and protocols, specifically addressing features required in the project. The results of cryptographic research is continuously published in scientific venues.

Tools. The second layer, denoted as PRISMACLOUD *tools*, represents a collection of basic tools such as the *Secure Object Storage Tool*. A tool can therefore be regarded as an abstract concept or piece of software, e.g., a library, which is composed of various primitives which can be parametrized in various different ways.

The major *outcome* on this level is twofold. On the one hand, we define and specify parametrisable PRISMACLOUD tools which can be used to build or augment services with cryptographic features. On the other hand, the tools and their most important features are going to be implemented during the project. They will be available in form of software libraries which will be used to build the services in the pilots. The PRISMACLOUD tools will be the foundation for mid-term exploitation of PRISMACLOUD results.

Services. From the tools of the toolbox, the *services* of the next layer can be built. A service can therefore be seen as a customisation of a particular tool for one specific application. It is a way to deliver the tool to system and application developers, the users of the tools, in a preconfigured and accessible form. They will be able to integrate the services without deeper understanding of tools and primitives and ideally without even being an IT security expert. A service provides a full implementation of all the required features as well as concrete interfaces in the form of an application programming interface (API), suitable to be deployed as a cloud service. We have chosen to specify a selection of services, which we will develop during the project and which are suitable for showcasing the suitability of the chosen primitives and the tools constructed from them within the selected use cases.

Applications. At the top most layer, above the services, are the applications represented by the use cases in PRISMACLOUD. They demonstrate how to use and compose different services in order to enrich the use cases with security and privacy functionality.

In addition to the localisation of the services within the architecture, it is also important to note where the services are located in the classical cloud stack, i.e., with respect to the Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Services (SaaS) layers. We provide

an overview of where the services are located in Figure 2. As it can be seen, we essentially have two services that are located on the SaaS level, three services on the PaaS level and three services at the IaaS level and thus covering the entire cloud stack.

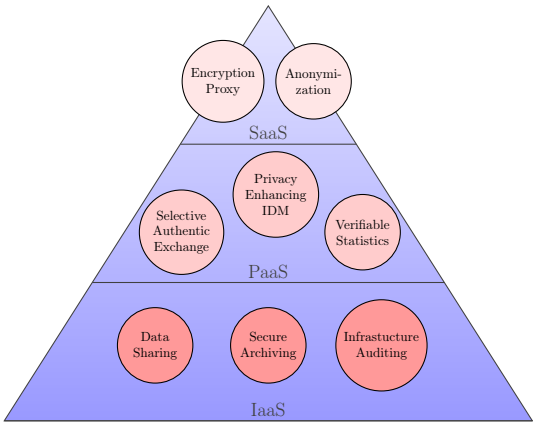

Fig. 2: The proposed services in the reference architecture.

With this architecture we encapsulate the cryptographic knowledge of the primitives layer inside the tools and their usage inside services. Building the tools requires in depth cryptographic and software development knowledge. However, once built, they can be used by cloud service designers to build cryptographically secure and privacy preserving cloud services. This cloud services are then exposed to application developers who can combine them with other technologies and services into the real end-user applications. The elements of the architecture required by the use cases will be implemented in the PRISMACLOUD project in software and provided for exploitation by the commercial project partners, and through them also beyond the project consortium.

Additionally to the discussed advantages, the PRISMACLOUD architecture further facilitates exploitation of project results. Each layer provides a dedicated project outcome with a specific exploitation path. Research progress on the primitives layer leads to scientific publications and typically associated exploitation. Tool developers will be able to commercialize software developments and intellectual property rights. Service developers are able to transform the project results in very short term into products. Their services will be almost ready for deployment in production environments of cloud providers. Hence, they will be accessible to a broader community relatively soon after the project's end. The project also features a specific *standardization activity* to disseminate the tools' specifications into standards to support further adoption.

III. PRISMACLOUD SERVICES

Recall that a service can be seen as customization of one particular tool (or several particular tools) for one specific application scenario. It provides a specific set of features which has been identified as particularly useful for a broader class of applications scenarios the service is targeting.

A. Data Sharing Service

The PRISMACLOUD data sharing service allows multiple parties to securely store data in a cloud-of-clouds network such that no single storage node learns plaintext data, while still enabling the owner to share the data with other users of the system, i.e., the data sharing service supports secure collaboration without the need to trust one single storage provider. The delivery model of this service is IaaS.

B. Secure Archiving Service

The PRISMACLOUD secure archiving service is a generic infrastructure service which can easily be integrated into cloud based backup scenarios while providing a demonstrable higher level of data privacy and availability than current cloud-based archiving solutions. The delivery model for this service is IaaS.

C. Selective Authentic Exchange Service

This service enables users to move their authentic documents to a cloud service and then delegate the selective sharing of parts of these documents to another party, while maintaining the authenticity of the selected parts. The other party can then verify the authenticity of the received data. The delivery model of this service is PaaS.

D. Privacy Enhancing ID Management Service

This service offers the capability of a privacy enhanced identity management. In particular, it allows users to store their attribute credentials obtained from some entity (e.g, a service provider or an authority) in this component and to realize a selective attribute disclosure functionality. The delivery model of this service is PaaS.

E. Verifiable Statistics Service

This service provides the functionality to delegate the computation of verifiable statistics on authenticated data in a secure way. The computations have the feature of being public verifiability, i.e., any verifier can check whether an outsourced computation has been performed correctly, or not. The delivery model for this service is PaaS.

F. Infrastructure Auditing Service

The infrastructure auditing service offers the capability to certify and prove properties of the topology of a cloud infrastructure without disclosing sensitive information about the actual infrastructure's blueprint. The delivery model associated to this service is IaaS.

G. Encryption Proxy Service

The service supports moving legacy applications to the cloud by encrypting sensitive information identified within HTTP traffic in a format and/or order preserving way. The delivery model associated to this service is SaaS.

H. Anonymization Service

This service enables users to anonymize large data sets, and in particular database tables. The service allows users to identify private and sensitive information in the data sets and produce an anonymized version of the data set. The delivery model associated to this service is SaaS.

IV. TOOLBOX

As already mentioned before, a tool can be thought of a an abstract concept, or piece of software, which is composed of various primitives and can be parametrized in various different ways. Subsequently, we briefly discuss all the tools.

A. Secure Object Storage Tool

This tool provides strong security guarantees, i.e., confidentiality and availability, for cloud storage and backup services. It builds upon a federated cloud-of-clouds approach [7] and the different components of the tool are shown in Figure 3. A *dealer* component generates data fragments and sends them to storage nodes called *server*. The servers can be considered active components with the ability to execute protocol logic in addition to the basic read and write operations supported by passive interfaces. The *reader* component is reconstructing the data for read operations and the *verifier* is able to audit the storage system, i.e., it can remotely check the state of the data stored in the system.

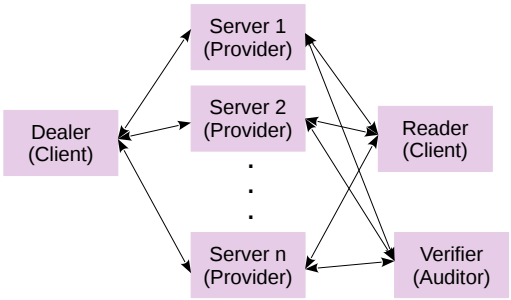

Fig. 3: Secure Object Storage Tool

To achieve the desired security properties we are leveraging the concept of cloud federation and information dispersal – data is fragmented and distributed over different servers to build a secure and reliable virtual service on top of multiple less reliable services. Our goal is to design a storage layer which is more than the sum of its parts. No single cloud provider shall have access to plain-text data or be able to tamper with them by modifying local fragments, hence, the Secure Object Storage Tool prevents from most of provider related threats in cloud usage.

In PRISMACLOUD we apply secure variants of data fragmentation called secret sharing (see Section V-A). With secret sharing, data can be encoded into multiple fragments such that no single fragment contains any information about the original content and a predefined threshold of k fragments is required to reconstruct the data. If these fragments (shares) are distributed to different storage servers, the data stays secure w.r.t. confidentiality protection as long as less than k servers are colluding to reconstruct the information and the data stays secure w.r.t. availability as long as more than k honest servers are reachable. Additionally, it must be noted that the confidentiality and availability guarantees of the PRISMACLOUD *Secure Object Storage Tool* are also keyless, which further facilitates sharing of data between different users in the system.

From a functional point-of-view, the tool provides a way for outsourcing data storage to potentially less trustworthy and reliable storage providers with both increased data availability and confidentiality [8]. Furthermore, we are targeting robustness against strong adversary models like active attackers or adversaries with unlimited computational power. Additionally the tool is supporting concurrent multi-user access by potentially malicious clients and provides other interesting features discussed below.

Technically we are combining protocols from distributed computing—Byzantine fault tolerance [9] in our case—and augment them with data confidentiality by means of secret sharing. The integration of secret sharing techniques is very appealing but challenging [10], as Byzantine protocols lose many of their properties if the processed data is not replicated plaintext anymore. To remediate this problem verifiable secret sharing (VSS) [11] can be applied. VSS is basically the cryptographic version of a Byzantine protocol, however, plain VSS based systems are too inefficient to be used in storage scenarios. We are developing new batch verifiable secret sharing protocols which are practically efficient and can be used for large amounts of data. They also protect from malicious clients and help within the system recovery and reconfiguration procedures. To further increase the user's privacy we are also investigating techniques for access privacy, i.e., private information retrieval (PIR) (cf. Section V-C). Additionally, we are developing efficient and privacy preserving protocols for remote data checking (see Section V-B). If no information about the data is leaked during an audit run, the regular system audits can be delegated to external untrusted parties.

Dealer Component. The main function of the dealer component (Dealer) is to encode the data and generate the different fragments used in the distributed storage network. The dealer works on unstructured data, and at its core, it implements secret sharing protocols. However, in order to support different application scenarios, adversary models and network models, the component must support various encoding schemes and protocols. The common functionality behind the different protocols is that of threshold secret sharing. The choice of threshold sharings used ranges from plain information theoretical solutions (PSS) to more efficient but only computational secure variants (CSS). To better protect from active attacks robust secret sharing schemes (RSS) are also an option in certain cases. Furthermore, if the dealer is untrusted, verifiable secret sharing schemes (VSS) can be used to prevent from inconsistent system states due to distribution of inconsistent

shares.

On the connectivity level, the dealer has one authentic and private channel to each server in the system in order to send the data to the various servers.

Server Component. The server component represents the storage backend used to compose the cloud-of-clouds storage layer. Multiple servers are required and each of them communicates with the Dealer over a secure channel (authentic and private). The different servers are holding the data and represent storage nodes in different trust zones of the system. The trust zones are used to model the non-collusion assumption. In practice, storage options range from fully-federated dedicated cloud providers to storage nodes under different administrative domains within a cloud provider's data center.

In addition to the client interface each server also has secure channels to all other servers. Using interconnected active nodes enables advanced functionalities, in particular concurrency, multi-user access, verifiability protocols, remote data checking and proactive security measures for long-term security. This features depend upon server side support for application logic. To be widely compatible with existing cloud offerings—thus preventing provider lock-in—the servers will be instantiated by small preconfigured virtual machines running a PRISMACLOUD server instance managing the respective locally allocated storage.

Reader Component. The reader component is responsible for reconstruction of data stored in the system. Basically it encapsulates the reconstruction procedure of the used secret sharing method plus the interaction protocol to get sufficient shares to recover the plaintext information desired. We assume secure channels between a reader and all involved servers in the system. A client which has read and write access can incorporate both a Reader and a Dealer component, which could be seen as an overarching storage component. The split between read-only and read-write components allows further security constraints when designing read-only clients.

Verifier Component. This component is responsible for the verification of data stored in the system. Together with the servers it conducts a protocol to obtain a proof about the retrievability of stored data, i.e., it remotely checks if the servers are still storing consistent fragments. The verifier is not considered to be trusted, therefore, the auditing protocols executed must be privacy preserving, i.e., it must not leak any information about the data stored. Based upon this, the verifier must have an authentic channel to each of the servers, but those channels don't need to be confidentiality protected. The Verifier is intended to model a third-party auditing service which can check the data consistency remotely with strong cryptographic properties but without learning anything about the data.

B. Flexible Authentication with Selective Disclosure Tool

This tool supports the authentication of arbitrary messages (or documents) by means of digital signatures with selective disclosure features. This tool has three different components (cf. Figure 4), being an *authentication component*, a *selective disclosure component*, and a *verification component*.

Authentication Component. The originator generates a signed message that carries in some well defined rules (policy) what parts of the message can be selectively disclosed.

Selective Disclosure Component. Given a signed message from the authentication component, it provides the functionality to selectively disclose parts of the information of the original signed message (or document) to other receiving parties. When this selective disclosure happens according to the defined policy, the authenticity can still be verified.

Verification Component. A verifying party can then use this part of the tool to verify the authenticity of any partial information that was created from authentic information just by means of the originator's verification key. Note, the *verification component* checks if only authorised, i.e., conforming to policy, selections were done.

Fig. 4: Flexible Authentication with Selective Disclosure Tool

For the realisation of this component several cryptographic primitives can be used. Especially if the authentication component shall be decoupled from the selective disclosure component, so that the latter can operate bound by the policy but without the need of an interaction, then a group of special digital signature schemes referred to as malleable signatures can be facilitated. Suitable primitives are discussed in Section V-D.

C. Verifiable Data Processing Tool

This tool supports the delegation of processing authenticated data in a way that the result can be efficiently verified for correctness. It comprises three different components depicted in Figure 5. The data (and potentially some additional metadata) originates at the *client component*. The *data processing component* is given a set of input data and a description of the processing rules, and outputs the result of the computation, as well as a proof certifying the correctness of the delegated computation. The *verification component* takes a result and a proof (and potentially additional information) and can efficiently verify the correctness of the computation.

On a high level, the Verifiable Data Processing Tool allows to perform verifiable computations on data such as computing statistics on medical data. Depending on the type of verifiability, thereby, the data processed can be either only verified by the data owner or any third party. Although there are very powerful technical tools for very broad classes of functions, they are not yet practically efficient. In our Deliverable D5.8 [12] we provide a comprehensive overview of the current state of the art in verifiable computing and discuss shortcomings of existing solutions.

Fig. 5: Verifiable Data Processing Tool.

In PRISMACLOUD we thus aim at developing a tool that provides efficient solutions for restricted classes of functions. Besides efficiency, we aim for schemes that are secure against so called strong (adaptive) adversaries, i.e., adversaries who can adaptively ask computation queries and also learn about whether the forgery attempts verify. This is required, because some data processed has a high protection level. For the same reason we also want to support schemes that offer input and/or output privacy. In a first step we only address computationally bounded adversaries and aim at extending the tool by variations covering computationally unbounded adversaries at a later stage. With respect to the underlying primitives, we focus on approaches where primitives with reasonable performance are available. For each variation we will take both types of verifiability, i.e., private and public verifiability, into account. The latter one is preferable, because it allows to perform third party audits. However, for some applications it might be sufficient that only the data owner, i.e., client component, is able to perform the verification and for this scenario we might get a more efficient solution when providing only private verifiability. The tool will heavily rely on malleable signature primitives (see Section V-D). In particular, to authenticate the input data which then support to evaluate arithmetic circuits. In addition the tool may also require secret sharing schemes (see Section V-A), functional signature schemes (see Section V-E) and zero-knowledge proofs (see Section V-H).

The components of the Verifiable Data Processing Tool are as follows:

Client Component. The *client component* produces a set of signed data on which it wants to have a function, represented as an (arithmetic) circuit $\mathcal{C}$, evaluated.

Computation Component. The *computation component* receives a circuit $\mathcal{C}$ and signed personal data from one or more client components and produces a signed output together with a proof of correct computation.

Verification Component. The *verification component* takes the result and proofs computed by the computation component and verifies the correctness of the evaluation of a circuit $\mathcal{C}$.

The tool developed will contain several variations that differ with respect to the functionalities supported. More precisely, we will provide different classes of operations and several levels of security, privacy, and verifiability. This allows to use the tool for many different applications.

D. Topology Certification Tool

The Topology Certification Tool supports the application of graph signatures (cf. Section V-I) to certify and prove properties of topologies represented as graphs. The tool is realized as an interactive protocol framework between the roles of an *issuer*, a *prover* and a *verifier*. It consists of three abstract components, depicted in Figure 6, that encapsulate the different roles. The topology is provided by another entity in a standard graph format.

Fig. 6: Topology Certification Tool

This specification relates to the cryptographic protocol framework proposed by Groß [13].

Issuer Component. Given a graph representation of the topology in a standard format, the issuer is responsible for the certification of the encoding for the topology certification framework, as well as for issuing a topology certificate to the prover. The issuer outputs a graph signature on that graph.

Prover Component. The prover compiles a zero-knowledge proof on the topology certificate that can convince a verifier of the requested security properties of the graph in a zero-knowledge proof of knowledge.

Verifier Component. The verifier validates the proof against the public key of the issuer and is convinced of the requested security properties.

On a high level, the Topology Certification Tool supports the creation of digital signatures on topology graphs in such a way that the graphs are accessible to zero-knowledge proofs of knowledge. This will support the infrastructure auditing service of PRISMACLOUD described in Section III-F. The infrastructure auditing service is tasked to show isolation of different resources, for example, whether the resources of a Tenant A are segregated from the resources of a Tenant B. Consequently, the infrastructure auditing service requires from the Topology Certification Tool the following functionalities beyond the foundational operations of the Issuer, Prover and Verifier components. First, the Issuer needs to be capable of issuing topology certificates in the size of the infrastructure in question. Second, the Prover and the Verifier need to be able to compute isolation proofs on topology certificates. Such proofs have been specified by Groß [13]. These proofs require that the proof of possession on the topology certificate needs to be further decomposed into commitments on the edges of the graph. Furthermore, the prover needs to prove equality of the committed values with the topology certificate, compute a cumulative product in commitments and finally show that these products are co-prime.

To facilitate these proofs, the graph considered should not be edge-labeled. For the interface of the infrastructure we only consider undirected edge-unlabeled graphs. The decomposition of the topology certificate in a number of commitments takes computational time proportional to the number of edges of the entire topology graph. To allow for life demonstrations, the topology size should be selected such that the proof can

be computed in reasonable time.

E. Data Privacy Tool

This tool provides the means for processing data in different ways before they are moved to untrusted environments. It includes components providing the capabilities to encrypt data while preserving the format or ordering of the data (cf. Section V-J). This tool enables users of legacy applications to move their databases to a public cloud, while preserving data privacy and confidentiality. Moreover, the tool provides components for data generalization as means for anonymizing bulk data using k-anonymity techniques (cf. Section V-K).

On a high level, this tool will provide the means to process data in different ways, supporting different purposes and different privacy requirements. It will include different components providing the following capabilities: (1) data encryption while preserving format or order, and (2) data generalization to guarantee k-anonymity. The tool's input may range from a single data item (e.g., to be encrypted) to a complete data set (e.g., to be anonymized). Based on the input data and the preferred privacy method, the processing will produce the required output.

V. Cryptographic Primitives and Protocols

In this section we intended to give a general overview on the capabilities of the cryptographic primitives and protocols which are used as technical basis for the PRISMACLOUD tools. We briefly recall them and provide reference to the respective deliverables or scientific literature which provide the detailed information.

A. Secret Sharing Schemes (SSS)

Secret sharing protocols [14] can be used to split a file into multiple fragments, such that every designated qualified set of these fragments can be used to reconstruct the file, while any other set of fragments does nor reveal any information about the file. In PRISMACLOUD we are engaging in research towards practically efficient verifiable secret sharing for large amounts of data and protocols which make use of this schemes, e.g., proactive secret sharing. We are developing dedicated batch versions of VSS [15], which reduce the induced computation and communication overhead for typical storage scenarios. First results have been achieved and are available in Deliverable D4.1 [16]. The results of our research will be used in tool secure object storage and verifiable computing.

B. Remote Data Checking (RDC)

Remote data checking [17], [18] allows a client to assure that the data she has stored on a remote storage server is still correct and available if needed. RDC is able to provide such a proof, while significantly reducing the communication and computation overhead, compared to a naive approach, i.e., checking the data integrity and availability by retrieving the entire data set. Ideally, such proofs are of constant length (independent of the size of the data) and the verification of these proofs requires only a small and constant number of computations at the server as well as at the client. In PRISMACLOUD we are developing such ideal proofs optimized for our multi-server setting in the secure object storage tool. Additionally, we are designing privacy preserving version as required for outsourcing of the auditing task to untrusted third parties [19].

C. Private Information Retrieval (PIR)

Private information retrieval (PIR) [20] is an approach which allows clients to query data items from a server without revealing to the server which item is retrieved. PIR can be designed for different settings, such as for data that is hosted by a single server or by multiple non-communicating servers [21], [22], and the goal is to come up with solutions that have a significantly lower communication complexity as the naive approach of retrieving all the data. In PRISMACLOUD we are developing novel efficient PIR protocols which are tailed for our multiple-server setting in the storage tool which leverages the existing communication channel between the severs.

D. Malleable Signature Schemes (MSS)

Malleable signature schemes support controlled modifications of signed messages that do not require the signer's secret key and do not destroy the validity of the digital signature. Thus, such schemes can be used to perform computations on signed (i.e., authenticated) data, as well as to add accountability to business processes and workflows. Practical instantiations of such schemes are for instance redactable [23], [24], [25] and sanitizable signatures [26] as well as various other types of homomorphic signature schemes. Among the many primitives in this group PRISMACLOUD has found especially accountable [27] or mergeable redactable signatures [28], blank digital signatures [29], and extended sanitizable signatures [30] of interest. Details about the state of the art and an overview has been published as Deliverable D4.4 [31].

E. Functional Signature Schemes (FSS)

Functional signatures [32], [33] allow the delegation of signature generation to other parties for a class of messages meeting certain conditions. Such schemes can be used to certify computations and processes. From this class of signature schemes we consider schemes that support functional signing keys, i.e., the delegation of signing keys which somehow include a functionality limiting their use with respect to which messages can be signed, or when, or how they can be signed. For more details we refer to Deliverable D4.4 [31].

F. Attribute-Based Credentials (ABC)

Attribute-based anonymous credentials [34] provide a means for anonymous authentication. An ABC system is a multi-party protocol involving a user, an organization (or issuer) and a verifying party. The user can obtain from an organization a credential on multiple attributes, such as her or his nationality, age, or gender, and later on present the credential to a verifier. By doing so, the user only reveals certain attributes (or proves a relation about these attributes).

While not learning any information about the user, the verifier can still be sure that presented information (through the shown attributes) is authentic. In a multi-show credential system, a user can perform an arbitrary number of unlinkable showings. For more details we refer to D4.6 [35]).

G. Group Signature Schemes (GSS)

Group signatures [36], [37] allow users to anonymously sign messages on behalf of a group, where group signing keys for single users may either be generated on behalf of the user by an issuing authority, or users may dynamically join the group in a way that the issuer does not learn the entire group signing key of a user. In case of a dispute, a so-called opening authority is able to reveal the identity of the actual signer from a given signature. For more details we refer to D4.6 [35].

H. Zero-Knowledge Proofs (ZKP)

Zero-knowledge proofs [38] are interactive proofs allowing one party to convince another party of the validity of a statement (a word in a formal language) without revealing any more information than the validity of the statement. Specific types of proof systems (such as Sigma proofs or Groth-Sahai proofs [39]) are particularly efficient and highly useful in the design of a multitude of privacy-preserving cryptographic schemes and protocols. For more details we refer to Deliverable D4.6 [35]. Besides interactive ZKPs, an important variant are non-interactive zero-knowledge proofs (NIKZ) and for verifiable computing non-interactive computationally sound (i.e., argument systems) denoted as (zk-) SNARKS are very interesting. For more details we refer to D4.4 [31] and D5.8 [12]).

I. Graph Signature Schemes (GRS)

Graph signatures [40] allow signing the representation of a graph (i.e., a set of vertices and edges). Given such a graph signature, it is possible to convince a verifier that the signed graph fulfils certain properties (e.g., isolation or connectedness) without disclosing the graph itself. For more details we refer to D4.6 [35].

J. Format- and Order-Preserving Encryption (XPE)

A format-preserving encryption (FPE) scheme [41] is an encryption scheme where the ciphertext space is identical to the plaintext space. A format can for instance be be the set of all possible valid credit card numbers. Order-preserving encryption (OPE) is a type of deterministic encryption whose encryption function preserves the numerical ordering of the plaintext. For more information on these encryption schemes we refer to Deliverable D4.9 [42].

K. k-Anonymity (kAN)

k-Anonymity [43] is a method for the anonymization of relational data (databases) in a way that, under well specified assumptions, it is provably guaranteed that no individual record can be distinguished from at least $k-1$ other records and thus under certain assumption allows to provably anonymize data. For more information we refer to Deliverable D5.5 [44].

VI. Summary and Conclusion

As solely relying on legal contracts and trusting the cloud is not a solution, PRISMACLOUD tackles the problems of security and privacy with the help of strong cryptographic primitives. The current low pervasion of already existing and maybe nearly usable strong cryptographic primitives to the practice withholds cloud usage for many security and privacy conscious usage scenarios, e.g., eHealth or eGovernment. We partly hold the required deep cryptographic knowledge to bring the primitives to practical use accountable for the bad proliferation. What we termed the PRISMACLOUD architecture can be seen as a recipe to bring cryptographic primitives and protocols into cloud services that empower cloud users to build more secure and more privacy-preserving cloud services. In its core we encapsulate the cryptographic knowledge in specific tools and offer basic but cryptographically enhanced functionality for cloud services. In PRISMACLOUD we will harvest the consortium members' cryptographic and developer knowledge to build the tool box and the services. The resulting PRISMACLOUD services hide and abstract away from the core cryptographic implementations and can then be taken by cloud service designers. On this level of cloud services, the PRISMACLOUD services will show how to provision (and potentially market) services with cryptographically increased security and privacy.

References

[1] Transparency Market Research, "Cloud computing services market – global industry size, share, trends, analysis and forecasts 2012-2018," 2012, (online 31.3.2015). [Online]. Available: http://www.transparencymarketresearch.com/cloud-computing-services-market.html

[2] PRWeb, "A cloud computing forecast summary for 2013-2017 from idc, gartner and kpmg, citing a study by accenture," 2013, (online 31.3.2015). [Online]. Available: http://www.prweb.com/releases/2013/11/prweb11341594.htm

[3] European Commission, "European Cloud Computing Strategy "Unleashing the Potential of Cloud Computing in Europe"," 2012, (online 31.3.2015). [Online]. Available: http://ec.europa.eu/digital-agenda/en/european-cloud-computing-strategy

[4] T. Lorünser, C. B. Rodriguez, D. Demirel, S. Fischer-Hübner, T. Groß, T. Länger, M. des Noes, H. C. Pöhls, B. Rozenberg, and D. Slamanig, "Towards a new paradigm for privacy and security in cloud services," in *CSP Forum 2015*, ser. LNCS, vol. 8874. Springer, 2015, pp. 1–12.

[5] N. Notario, A. Crespo, Y. S. MartÃn, J. M. D. Alamo, D. L. MÃ©tayer, T. Antignac, A. Kung, I. Kroener, and D. Wright, "Pripare: Integrating privacy best practices into a privacy engineering methodology," in *Security and Privacy Workshops (SPW), 2015 IEEE*, May 2015, pp. 151–158.

[6] T. Lorünser, T. Länger, and D. Slamanig, *Cloud Security and Privacy by Design*. Cham: Springer International Publishing, 2015, pp. 202–206.

[7] D. Slamanig and C. Hanser, "On cloud storage and the cloud of clouds approach," in *ICITST 2012*, 2012, pp. 649–655.

[8] T. Loruenser, A. Happe, and D. Slamanig, "ARCHISTAR: Towards Secure and Robust Cloud Based Data Sharing," in *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, nov 2015, pp. 371–378.

[9] L. Lamport, R. E. Shostak, and M. C. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[10] A. Happe, S. Krenn, and T. Loruenser, "PBFT and Secret-Sharing in Storage Settings," in *International Workshop on Security Protocols, Brno, Czech Republic â 7-8 April 2016*. IEEE, 2016.

[11] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, 1985, pp. 383–395.

[12] J. Buchmann, D. Demirel, D. Derler, L. Schabhüser, and D. Slamanig, "PRISMACLOUD D5.8: Overview of Verifiable Computing Techniques Providing Private and Public Verifiability," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.

[13] T. Groß, "Efficient certification and zero-knowledge proofs of knowledge on infrastructure topology graphs," in *ACM Workshop on Cloud Computing Security (CCSW 2014)*. ACM, 2014, pp. 69–80.

[14] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, nov 1979.

[15] M. Backes, A. Datta, and A. Kate, "Asynchronous computational VSS with reduced communication complexity," in *CT-RSA*, 2013.

[16] J. Buchmann, D. Demirel, A. Happe, S. Krenn, T. Lorünser, and G. Traverso, "PRISMACLOUD D4.1: Secret Sharing Protocols for Various Adversary Models," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.

[17] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *ACM CCS*, 2007, pp. 584–597.

[18] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, 2011.

[19] D. Demirel, S. Krenn, T. Loruenser, and G. Traverso, "Efficient Thrid Party Auditing for a Distributed Storage System," in *submitted*, 2016.

[20] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *FOCS*, 1995, pp. 41–50.

[21] I. Goldberg, "Improving the robustness of private information retrieval," in *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, 2007, pp. 131–148.

[22] R. Henry, Y. Huang, and I. Goldberg, "One (Block) Size Fits All: PIR and SPIR Over Arbitrary-Length Records via Multi-block PIR Queries ," in *NDSS*, 2013.

[23] R. Steinfeld, L. Bull, and Y. Zheng, "Content Extraction Signatures," in *ICISC*, ser. LNCS, vol. 2288. Springer, 2001, pp. 285–304.

[24] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Topics in Cryptology—CT-RSA 2002*. Springer, 2002, pp. 244–262.

[25] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig, "A General Framework for Redactable Signatures and New Constructions," in *Information Security and Cryptology - ICISC 2015*, ser. LNCS, vol. 9558. Springer, 2015, pp. 3–19.

[26] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable Signatures," in *ESORICS'05*, ser. LNCS, vol. 3679, 2005.

[27] H. C. Pöhls and K. Samelin, "Accountable redactable signatures," in *ARES 2015*. IEEE, Aug. 2015.

[28] H. C. Pöhls and K. Samelin, "On Updatable Redactable Signatures," in *ACNS*, 2014, pp. 457–475.

[29] C. Hanser and D. Slamanig, "Blank Digital Signatures," in *8th ACM SIGSAC Symposium on Information, Computer and Communications Security (AsiaCCS). Full Version: Cryptology ePrint Archive, Report 2013/130*. ACM, 2013, pp. 95–106.

[30] D. Derler and D. Slamanig, "Rethinking Privacy for Extended Sanitizable Signatures and a Black-Box Construction of Strongly Private Schemes," in *Provable Security, ProvSec 2015*, ser. LNCS, vol. 9451. Springer, 2015, pp. 455–474.

[31] D. Demirel, D. Derler, C. Hanser, H. C. Pöhls, D. Slamanig, and G. Traverso, "PRISMACLOUD D4.4: Overview of Functional and Malleable Signature Schemes," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.

[32] E. Boyle, S. Goldwasser, and I. Ivan, "Functional Signatures and Pseudorandom Functions," in *Public-Key Cryptography - PKC 2014*, ser. LNCS, vol. 8383. Springer, 2014, pp. 501–519.

[33] M. Bellare and G. Fuchsbauer, "Policy-Based Signatures," in *PKC 2014*, ser. LNCS. Springer, 2014.

[34] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology - EUROCRYPT 2001*, 2001, pp. 93–118.

[35] T. Groß and D. Slamanig, "PRISMACLOUD D4.6: First Year Research Report on Privacy-Enhancing Cryptography," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.

[36] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, 1991, pp. 257–265.

[37] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology - CRYPTO 2000*, 2000, pp. 255–270.

[38] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems (extended abstract)," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, 1985, pp. 291–304.

[39] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, 2008, pp. 415–432.

[40] T. Groß, "Signatures and efficient proofs on committed graphs and NP-statements," in *19th International Conference on Financial Cryptography and Data Security (FC 2015)*, 2015, pp. 293–314.

[41] J. Black and P. Rogaway, "Ciphers with arbitrary finite domains," in *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, 2002, pp. 114–130.

[42] B. Rozenberg, "PRISMACLOUD D4.9: Analysis of the State of the Art of FPE, OPE and Tokenization schemes," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.

[43] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[44] M. Moffie, "PRISMACLOUD D5.5: Analysis of the requirements for and the state of the art for privacy and anonymisation techniques," H2020 Prismacloud, www.prismacloud.eu, Tech. Rep., 2015.