prisma cloud

Final Report on Privacy and Anonymization Techniques (TOPOCERT)

Deliverable D5.7

Editor Name	Thomas Groß (UNEW)
Type	Report
Dissem. Level	PU
Release Date	M30
Version	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644962.

More information available at https://prismacloud.eu.

Copyright Statement

Executive Summary

PRISMACLOUD implements novel cryptographic concepts and methods to lift them into practical application and improve the security and privacy of cloud based services, while at the same time make the services accessible to providers and end users.

The purpose of this report is to outline the architecture and design of the cryptographic tool TOPOCERT. The TOPOCERT tool aims to facilitate the certification and verification of cloud infrastructures. The deliverable describes the design paradigms of the tool, put in context of its terms and definitions. It outlines the component model and static architecture, determining scope and responsibilities of components and roles in cryptographic protocols. It summarizes the background on the confidentiality preserving security assurance, provides an overview on the geo-location separation and presents research regarding the application of the TOPOCERT tool beyond the application in the e-Government use-case within PRISMACLOUD.

Table of Contents

Document information

Project Context

Work Package	WP5 Efficient and Secure Implementations
Task	T5.3 Secure and privacy preserving processing of authenticated data
Dependencies	D4.6, D4.7

Author List

Organization	Name	E-mail
UNEW	Thomas Groß	thomas.gross@newcastle.ac.uk
UNEW	Ioannis Sfyrakis	ioannis.sfyrakis@newcastle.ac.uk

Reviewer List

Organization	Name	E-mail
TU Graz	Daniel Slamanig	daniel.slamanig@tugraz.at

Version History

Version	Date	Reason/Change	Editor
0.1	2017-05-04	1st Draft	Thomas Groß
0.2	2017-07-05	updated services paragraphs	Ioannis Sfyrakis
0.3	2017-07-10	terms, definitions, abstract description	Thomas Groß
0.4	2017-07-17	added appendix, papers, background, updated services	Ioannis Sfyrakis
0.5	2017-07-24	parameter specification	Thomas Groß
0.6	2017-07-25	detailed spec of algorithms, I/O	Thomas Groß
0.7	2017-07-27	detailed spec of TOPOCERT algorithms, I/O	Thomas Groß
0.8	2017-07-27	Interface finalization, sync with D6.6	Thomas Groß
0.9	2017-07-27	Integrated geo-separation spec	Thomas Groß
1.0	2017-07-30	Integrated UML diagrams, ZKPoK, background	Thomas Groß

1 Introduction

1.1 Scope of the document

The TOPOCERT tool offers an approach for integrity and privacy for clouds and is a component of the PRISMACLOUD architecture as illustrated in Figure 1. TOPOCERT yields the possibility to certify and prove properties of cloud infrastructures without disclosing the layout of the infrastructure. In this document, we describe the architecture and design of the tool, starting from terms and definitions, over design paradigms and component model, to the static software architecture. Subsequently, this document outlines the background on confidentiality-preserving security assurance, discusses our contribution on geo-location separation and presents research directions investigating the use of the TOPOCERT tool beyond the application context within the PRISMACLOUD pilots.

TOPOCERT realizes what could be described as a credential system on cloud topologies and thereby follows the design paradigms of established anonymous credential schemes.

This report presents the final architecture and design for the set of libraries that define the TOPOCERT tool. Details on the implementation are already presented in deliverable D6.6.

1.2 Relation to other project work

This deliverable relates to the WP 4 and the corresponding research into cryptographic primitives in that it offers a concrete design for the realization of theses primitives into software. This deliverable is thereby related to the deliverables D4.6 and D4.7 representing *First Year Research and Progress Report on Privacy-Enhancing Cryptography* respectively. The deliverable D4.8 is the next iteration representing the *Report on Privacy-Enhancing Cryptography*. Furthermore, the cryptographic techniques used in this report are also related to D4.4 *Overview of Functional and Malleable Signature Schemes*

This deliverable lays the foundation for the TOPOCERT tool design as well as for the services using the tool (cf. deliverables D7.3, D7.4, D7.5, D7.6, D7.8)

1.3 Structure of the document

This deliverable contains the description for the TOPOCERT tool. First, we give a high-level overview. Second, we define terms and definitions important to set the stage for the description of the tool. Third, we describe the component model of the tool. Fourth, we describe distinct components including the static and possibly dynamic software architecture. Fifth, discuss recommendations to be adhered for a secure implementation of the tool. Sixth, we outline the background on confidentiality-preserving security assurance. Seventh, we discuss our geo-location separation framework. Finally, we position the TOPOCERT tool inside the e-Government pilot of PRISMACLOUD, and present our research on additional applications beyond the use cases of PRISMACLOUD.

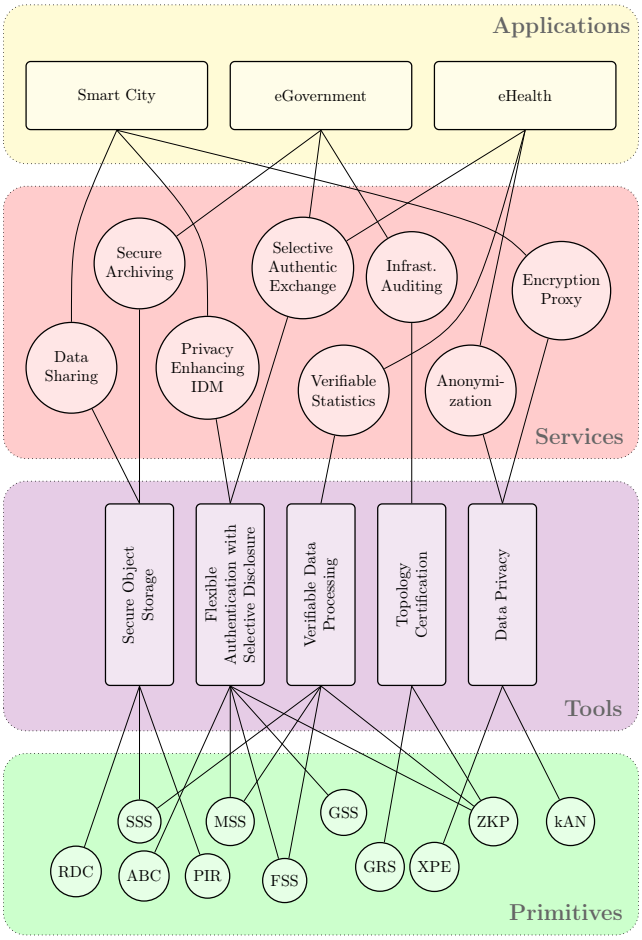

Figure 1: PRISMACLOUD architecture

2 The TOPOCERT Tool

2.1 Overview

2.1.1 Scope Definition

Goals

- During setup time, the auditor will specify a language for the graph signatures on topologies.
- At setup time, the auditor will determine the largest graph size to be certified in number of vertices n and of edges m, the alphabets and representatives for vertices ($\mathcal{V}$ and $\Xi_{\mathcal{V}}$) as well as for labels ($\mathcal{L}$ and $\Xi_{\mathcal{L}}$).
- Upon receiving a graph representation G the auditor will issue a graph signature thereon. This is facilitated in an interactive protocol with the provider.

- Holding a graph signature for G and having received a policy predicate $\mathfrak{P}$ of a tenant, the provider will compute a zero-knowledge proof of knowledge that shows the predicate to be fulfilled for the signed G.
- The tenant is enabled to declare a policy predicate $\mathfrak{P}$. Upon the interaction with the provider, the tenant can verify with respect to the public key of the auditor that the provider indeed holds a valid topology certificate of the auditor and that the policy predicate $\mathfrak{P}$ holds on the certified graph.

Non-Goals

- It is out of scope of this specification how the graph representation of the infrastructure is derived. In the architecture there shall be a method Inspect(), which acts as placeholder for the derivation.
- It is out of scope of this specification how the auditor verifies that the obtained graph representation is indeed a faithful rendering of the current state of the cloud infrastructure.

2.1.2 Tool Architecture

The TOPOCERT tool operates on top of a graph signature library, where the TOPOCERT tool is responsible to organize the setup, issuing and proof processes pertaining to topologies, while the graph signature library is responsible to setup, issue and prove graph signatures on general graphs.

The architecture of the graph signature library, in turn, is influenced by the established design and realization of anonymous credential systems, especially, the IBM Identity Mixer Library [IBM13]. The underlying reason for that is that Identity Mixer as well as the graph signatures are based on the Camenisch-Lysyanskaya signature scheme [CL02].

Notably, the design of the graph signature library does not follow the JCE/JCA provider paradigms. Instead, the graph signature library consists of a number of issuer, prover and verifier algorithms that specialize in handling graph messages and in proving predicates over graphs. One possible realization can build on-top of the IBM Identity Mixer Library [IBM13] to draw upon efficient tried and tested implementations of low-level algorithms on Camenisch-Lysyanskaya signatures.

2.1.3 Services Based on TOPOCERT

Infrastructure Auditing (IA) service. For this service, we have separated the main functionalities into three services. In the following we outline the services:

- *Audit-Profiles Management service*: this service provides a REST interface to auditors that can specify and certify audit profiles. Upon obtaining a graph representation using the REST API the service will issue a partial graph signature.

- *Infrastructure Auditing Management service*: cloud providers use this service to search and select audit profiles according to their requirements. The service also provides a REST API for providers that want to sign their infrastructure and provide a graph representation to auditors.
- *Geo-Separation service:* provides a REST API that offers policy predicates. Tenants can use this service to ask the provider for proofs on topologies.

2.1.4 Software Implementation

The software implementation will be in Java. It will not follow the Java Cryptography Architecture. Instead it will be based on design paradigms commonly used in anonymous credential systems, such as the realization of the IBM Identity Mixer Library [IBM13].

2.2 Terms and Definitions

2.2.1 Roles

The TOPOCERT implementation considers high-level and low-level roles.

- Auditor – Signer
- Provider – Recipient/Prover
- Tenant – Verifier

2.2.2 Auditor

An auditor is a party responsible for setting up a graph signature scheme for topologies and for the certification of graphs obtained from the provider. It is the certification authority of the TOPOCERT system.

2.2.3 Signer

The signer is a low-level role of the graph signature library that is responsible for the low-level key setup and the signing of committed graphs.

2.2.4 Provider

The provider acts as topology certificate/graph signature recipient towards the auditor and as prover of security properties towards tenants.

2.2.5 Recipient

The (signature) recipient is a low-level role of the graph signature library that is responsible for initiating the provider side of the signing process and to complete the signature on the provider's side.

2.2.6 Prover

The prover is a low-level role of the graph signature library that is responsible for the proof of possession and vertex and edge decomposition of the graphs.

2.2.7 Tenant

The tenant role is enabled to communicate to the provider a policy predicate $\mathfrak{P}$, to which the provider responds with a Zero-Knowledge Proof of Knowledge verifiable by the tenant with respect to the public key of the auditor.

2.2.8 Verifier

The verifier is a low-level role of the graph signature scheme, poised to verify zero-knowledge proofs of knowledge coming from the prover with respect to the signer public key and the given policy predicate $\mathfrak{P}$.

2.2.9 Graph

Graphs are defined over finite vertex sets with undirected edges and finite sets of vertex and edge labels.

$\mathcal{V}$	Finite set of vertices
$\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$	Finite set of edges
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, t_{\mathcal{V}}, t_{\mathcal{E}})$	Graph
$\mathcal{L}_{\mathcal{V}}, \mathcal{L}_{\mathcal{E}}$	Finite sets of vertex and edge labels
$f_{\mathcal{V}} : \mathcal{V} \to \mathcal{P}(\mathcal{L}_{\mathcal{V}})$	Labels of a given vertex
$f_{\mathcal{E}} : \mathcal{E} \to \mathcal{P}(\mathcal{L}_{\mathcal{E}})$	Labels of a given edge
$n = \lvert\mathcal{V}\rvert, m = \lvert\mathcal{E}\rvert$	Number of vertices and edges

In terms of architecture, graphs are represented by a standard library, such as JGraph, that support a serialization and deserialization with respect to standard graph data formats, such as GML or GEFX.

2.2.10 Vertex

A vertex is a node in a graph or topology. In TOPOCERT, a vertex can represent a physical or virtual device, for instance a virtual machine, a network or storage device.

2.2.11 Edge

In a graph an edge is a pair of two vertices, representing the connection between them. In TOPOCERT, an edge can represent a physical connection, a hierarchical connection (e.g., a VM being on-top of a physical machine), or a virtual connection (e.g., a VLAN connection).

2.2.12 Label

A label is an associated datum from a pre-defined finite set, which annotate the graph. In TOPOCERT, a vertex label can, for instance, represent the geo-location (country) of a component.

2.2.13 Realization Model

The realization model [BGSE11, BVG14, BVGM15] is a graph/topology representation of a virtualized infrastructure, which encodes physical and virtual components as well as the connections between them.

2.2.14 Message

A message is a piece of information being signed. With respect to the underlying Camenisch-Lysyanskaya signature scheme the message is a bit-string or an integer number. For the graph signature scheme the high-level message is a graph, whose components (vertex and edge descriptions) are then encoded into low-level messages over the integers.

2.2.15 Topology Certification

Topology certification [Gro14] is the process of the auditor to obtain a faithful graph representation (realization model) of a cloud infrastructure representing its topology and to sign this representation. The topology representation can be obtained from graph-based cloud security analysis tools [BGSE11, BVG14, BVGM15].

2.2.16 Topology Certificate

A topology certificate is a signed representation of a provider's topology. It consists of a graph signature on the realization model and auxiliary information for the provider's use of the topology certificate in subsequent proofs.

2.2.17 Graph Signature

A graph signature is a digital signature on a graph data structure that enables a prover to prove statements over the signed graph in zero-knowledge proofs of knowledge.

2.2.18 Zero-Knowledge Proof of Knowledge

A zero-knowledge proof of knowledge is a cryptographic methods that enables a prover to convince another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true. Per default, a proof of knowledge (represented as PK) is an interactive protocol between a prover and a verifier, in which the verifier contributes a random challenge.

We represent honest-verifier zero-knowledge protocols in the Camenisch-Stadler notation [CS97]. For example, the statement

$$\begin{aligned} &PK\{(x, r) : \\ &\quad C \equiv \pm R^x S^r \bmod N \\ &\} \end{aligned}$$

denotes that a Prover and a Verifier execute an interactive honest-verifier Σ-proof, in which the Prover proves the representation of the Integer commitment $C = R^x S^r \bmod N$. The papers in the appendix contain additional explanation on how the notation is used.

2.2.19 Signature Proof of Knowledge

A signature proof of knowledge (represented as SPK) is a zero-knowledge proof which is made non-interactive with the Fiat-Shamir heuristic, in that the prover computes the challenge himself by hashing context and previous information of the protocol run. Only a single message is non-interactively to the verifier.

2.2.20 Commitment Scheme

A commitment scheme is a cryptographic primitive that allows a committer to commit to a chosen value, while keeping this value secret (hiding). Commitment schemes prevent the committer from changing the value after having committed to it (binding). The

graph signature scheme and the underlying Camenisch-Lysyanskaya signatures operate on committed values.

2.2.21 Public Key

The TOPOCERT scheme is governed by the public key of the auditor, a key distributed widely with integrity, especially to providers and tenants.

2.2.22 Private Key

In TOPOCERT, the private key refers to the private signing key of the auditor, under which graph signatures and topology certificates are issued.

2.2.23 Issuing

Issuing is the process of creating a new graph signature or topology certificate. The issuing process is an interactive protocol between provider (prover) and auditor (signer), operates on committed values and establishes the signature without the auditor gaining linkable information in the process.

2.2.24 Graph Signature Scheme

A graph signature scheme [Gro15] is a cryptographic primitive and corresponding protocol suite, which enables the signing of committed graphs and proofs of knowledge of predicates over signed graphs.

2.2.25 Anonymous Credential Scheme

An anonymous credentials scheme is a cryptographic primitive and corresponding protocol suite, which enables the signing of committed attributes and proofs of knowledge thereon.

2.2.26 Policy Predicate

A policy predicate, such as defined by Bleikertz and Groß [BG11] is a statement over secret values, here representing a graph, that may be evaluated to true or false.

2.2.27 Vertex/Label Identifier

A vertex or label identifier is a number which represents the vertex or label.

2.2.28 Prime Representative

A prime representative is a prime number that encodes a vertex or label in a prime encoding.

2.2.29 Prime Encoding

A particular encoding for enumerations, binary values or finite sets into anonymous credential schemes [CG08, CG12].

2.2.30 Camenisch-Groß Encoding

Another name of the prime encoding [CG08, CG12].

2.2.31 Geo-Location

A label from pre-specified finite set that represents a location in the physical world [Gro17].

2.2.32 Proof of Representation

A proof of representation is a zero-knowledge proof of knowledge that proves a proof predicate which shows that certain public values are well-formed with respect to a specified representation and that the prover knows the secrets to compute the representation.

2.2.33 Proof of Possession

A proof of possession is a zero-knowledge proof of knowledge that convinces a verifier that a prover possesses an anonymous credential or, in this domain, a graph signature or a topology certificate.

2.2.34 Proof of Vertex/Edge Composition

A proof of a vertex or edge composition is a zero-knowledge proof of knowledge that convinces the verifier that a graph signature can be decomposed into commitments which encode the graph signature's parts (vertex, edges and their labels).

2.2.35 Proof of Separation

A proof of separation (especially with respect to geo-location) is a zero-knowledge proof of knowledge that convinces the verifier that the vertices of the graph can be classified

into a k-partition, for which holds that the partition's subsets are disjoint with respect to a class of vertex labels.

2.2.36 Proof of Isolation

A proof of isolation is a zero-knowledge proof of knowledge which convinces the verifier that there exists a partition on a graph signature for which holds that there are no edges bridging the subsets of the partition.

2.2.37 Proof of Connectivity

A proof of connectivity is a zero-knowledge proof of knowledge, which convinces the verifier that there exists a path in a signed graph that connects two specified vertices with at most k hops.

2.2.38 Partition

A k-partition is a way of separating of a set into k non-empty subsets, such that every element of the overall set is contained in one and only one subset.

2.2.39 Disjointness

Sets are called pair-wise disjoint if the the intersection of any two distinct sets is empty.

2.3 Component Model of the TOPOCERT Tool

In the component model of anonymous credential schemes, there are signer, recipient, prover and verifier components, which collaborate in interactive cryptographic protocols to facilitate the signing and proof functions. We consider design paradigms first.

2.3.1 Design Paradigms

For each signer component there is a matching recipient component, for each prover component, there is a matching verifier component.

Prover and verifier components offer methods for two stages: a commitment stage (Step 1) and a response stage (Step 2). The commitment stage requires as input the policy predicate $\mathfrak{P}$ as well as the setup parameters (public key, language specification, group setup). The commitment stage is responsible for storing the protocol state. The response stage requires as input the verifier's or Fiat-Shamir challenge and draws upon the commitment stage state.

All prover components of a party are operating on the same proof context. In particular, all provers operate on the same challenge. They also have access to the randomness representing secrets in the proofs and all commitments computed in the vertex and edge decomposition.

2.3.2 Auditor

The auditor component is responsible for the key setup and certification of a graph encoding scheme. The auditor inspects a provider's infrastructure configuration, thereby obtains a trustworthy graph representation of the infrastructure and signs an encoding of this representation. The auditor draws upon the signer role of the graph signature scheme.

2.3.3 Provider

The provider interacts with the auditor to receive a graph signature on his infrastructure. Subsequently, the provider is enabled to act as prover towards a tenant and prove in zero-knowledge that required properties of the graph signature are fulfilled. The provider can draw upon the recipient and the prover role of the graph signature scheme.

2.4 Tenant

The tenant acts specifies a policy predicate $\mathfrak{P}$, which is subsequently proven by the provider. The tenant acts as a verifier on the proof made by the providers. The tenant draws upon the verifier role of the graph signature scheme.

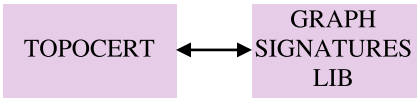

Figure 2: Library components of the TOPOCERT tool.

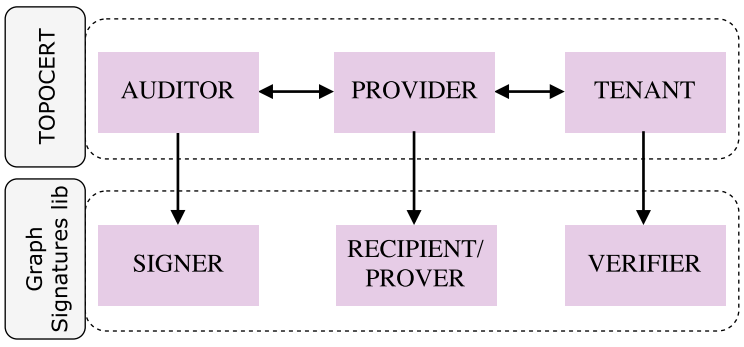

Figure 3: Abstract components of the TOPOCERT tool.

2.5 TOPOCERT Tool

2.5.1 Auditor

Setup. The auditor is responsible for setting up a certification environment for the topology certification, which includes the certification of the topology language. As such, the auditor is responsible for the selection of vertex identifiers and label identifiers. As such, the auditor specifies the message space that can be governed by TOPOCERT, which entails the pre-determination of the maximal graph size the system can handle. It delegates the generation of further key material to the signer role.

Inspection. The auditor is responsible for the inspection of the provider's cloud configuration. It is out of scope for this architecture how the auditor facilitates this inspection and how the auditor convinces himself that the actual configuration is faithfully represented in the graph representation, the realization model. For the purpose of this architecture, the auditor obtains graph representation G.

Issuing. The issuing is an interactive protocol between provider and auditor. At the start of the protocol, it is assumed that the auditor has completed the inspection and holds a known graph representation G. The actual issuing protocol is initiated by the provider and leads to the auditor computing a partial graph signature handed over to the provider (recipient)

2.5.2 Provider

Inspection Assurance. The provider is responsible for delivering a graph representation of the topology to the auditor or to enable the auditor to inspect the infrastructure's configuration directly. This can happen through enabling the auditor with direct access to the infrastructure, to management hosts, or through assurance evidence supporting the graph representation.

Issuing Reception. The provider starts the issuing protocol by committing to his own master secret key and possibly hidden values. He communicates this commitment to the auditor along with a proof of representation. Once the auditor has issued the partial graph signature, the provider will complete it to its full form.

Proving. In the proof protocol, the provider acts reactively to a policy predicate $\mathfrak{P}$ received from the tenant. Having received that, the provider enters the prover role, draws on the graph signature on the topology and the graph representation to create a zero-knowledge proof of knowledge on the policy predicate $\mathfrak{P}$. This is then communicated to the tenant.

2.5.3 Tenant

Verifying. The tenant aims at verifying the cryptographic evidence on the policy predicate $\mathfrak{P}$ with respect to the public key of the auditor. The tenant starts the protocol by sending the policy predicate $\mathfrak{P}$ along with a nonce. In the interactive version (PK) of the verification, the tenant responds with a random challenge. In the non-interactive version (SPK), the provider computes the challenge himself with the Fiat-Shamir heuristic, based on the context acquired up to that point. Hence, in this case the tenant will receive witness commitments, along with the responses that conclude the protocol.

2.5.4 Abstract Description

The TOPOCERT tool draws upon the operations $\mathsf{Keygen}(1^\lambda, gs\_params)$, $\mathsf{Commit}(\mathcal{G}; R)$, $\mathsf{HiddenSign}(C, \mathcal{V}_\mathsf{R}, \mathcal{V}_\mathsf{S}, pk_{\mathsf{S,E}})$, and $\mathsf{Verify}(pk_{\mathsf{S,E}}, C, R', \sigma)$ from the underlying graph signature scheme defined in Section 2.6.6 below.

Core Interface. The TOPOCERT tool realizes operations for the topology certification.

Definition 2.1 (TOPOCERT Tool)**.** *The tool consists of the following algorithms:*

$((pk_\mathsf{S}, sk_\mathsf{S}), \sigma_{\mathsf{S,kg}}) \leftarrow \mathsf{Keygen}(1^\lambda, gs\_params)$ *A probabilistic polynomial-time algorithm which computes the key setup of the TOPOCERT tool, delegating to the key generation* Keygen() *of the underlying graph signature scheme.*

$((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,me}}, \mathcal{M}_\mathsf{E}) \leftarrow \mathsf{GraphEncodingSetup}((pk_\mathsf{S}, sk_\mathsf{S}), \sigma_{\mathsf{S,kg}}, \mathcal{V}, \mathcal{L}, enc\_params)$ *A probabilistic polynomial-time algorithm which creates and certifies the encoding scheme for the topology certification. This process entails the systematic allocation of prime representatives for all possible vertices and labels in a mapping $\mathcal{M}_\mathsf{E}$ as well as the creation of generators to act as message based for the graph signature scheme. The latter is delegated to the graph signature library.*

$(\mathcal{G}; \epsilon) \leftarrow \mathsf{Inspect}(\mathcal{G})$ *An interactive algorithm by which the auditor inspects the provider infrastructure operation and obtains a trustworthy graph representation, the realization model.*

$(\sigma; \epsilon) \leftarrow \mathsf{HiddenSign}(pk_{\mathsf{S,E}}, \mathcal{M}_\mathsf{E}, C, V_\mathsf{R}, V_\mathsf{S})$ *An interactive probabilistic polynomial-time algorithm between auditor and provider which offers a topology signature on a issuer-known graph $\mathcal{G}$ obtained during* Inspect(), *delegates to the* HiddenSign() *operation of the graph signature scheme.* Private inputs: *Recipient* R*: $\mathcal{G}_\mathsf{R}$, commitment randomness R; Signer* S*: $sk_{\mathsf{S,E}}$, $\mathcal{G}_\mathsf{S}$.*

0 or 1 $\leftarrow \mathsf{Verify}(pk_\mathsf{S}, C, R', \sigma)$ *A verification algorithm on graph commitment C and signature σ. Delegates to the corresponding method of the graph signature scheme.*

The proof of policy predicates $\mathfrak{P}$ is realized with specified Σ-proofs in the graph signature.

Algorithm Input/Output Specification. We define the inputs and outputs for the abstract interface as follows.

Definition 2.2 $(((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}) \leftarrow \mathsf{Keygen}(1^{\lambda}, gs\_params))$**.** *A probabilistic polynomial-time algorithm which computes the key setup of the TOPOCERT tool, delegating to the key generation* Keygen() *of the underlying graph signature scheme.*

The specification of the inputs and outputs is identical with Definition 2.7, p. 24 of the low-level Keygen().

Definition 2.3 $(((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,me}}, \mathcal{M}_{\mathsf{E}}) \leftarrow$ $\mathsf{GraphEncodingSetup}((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}, \mathcal{V}, \mathcal{L}, enc\_params))$**.** *A probabilistic polynomial-time algorithm which creates and certifies the encoding scheme for the topology certification. This process entails the systematic allocation of prime representatives for all possible vertices and labels in a mapping* $\mathcal{M}_{\mathsf{E}}$ *as well as the creation of generators to act as message based for the graph signature scheme. The latter is delegated to the graph signature library.*

In addition to the inputs and outputs specified for the low-level GraphEncodingSetup() defined in Definition 2.8 on p. 24, the TOPOCERT encoding setup accepts as inputs sets of vertices and labels that constitute the message space. Note that the label set $\mathcal{L}$ can distinguish vertex and edge label sets, $\mathcal{L}_{\mathcal{V}}$ and $\mathcal{L}_{\mathcal{E}}$.

As default strategy for the TOPOCERT, labels $\mathcal{L}$are allocated low prime representatives. The vertices $\mathcal{V}$are allocated prime representatives greater than the dedicated bit length for the label encoding $\ell'_{\mathcal{L}}$. The reason for that default setting is the efficient encoding of multi-labeled graphs.

The TOPOCERTGraphEncodingSetup() may call the graph signature library for its GraphEncodingSetup() or draw upon stored outputs of previous runs of the graph signature GraphEncodingSetup(). The reuse has the following interface:

$$((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,me}}, \mathcal{M}_{\mathsf{E}}) \leftarrow \mathsf{GraphEncodingSetup}((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,es}}, \mathcal{V}, \mathcal{L}, enc\_params)$$

As output the algorithm produces a mapping between prime representatives and graph elements $\mathcal{M}_{\mathsf{E}}$. This mapping constitutes the alphabet that defines the message space for a particular encoding scheme.

The algorithm outputs an instance of the extended public key $(pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}})$ from the low-level GraphEncodingSetup(), which is amended with the established mapping $\mathcal{M}_{\mathsf{E}}$, as well as a signature $\sigma_{\mathsf{S,me}}$. The signature $\sigma_{\mathsf{S,me}}$ incorporates the signature $\sigma_{\mathsf{S,es}}$ on the encoding bases and certifies in addition mapping $\mathcal{M}_{\mathsf{E}}$.

Definition 2.4 $((\sigma; \epsilon) \leftarrow \mathsf{HiddenSign}(pk_{\mathsf{S,E},\mathcal{M}_{\mathsf{E}}}, C, V_{\mathsf{R}}, V_{\mathsf{S}}))$**.** *An interactive probabilistic polynomial-time algorithm between auditor and provider which offers a topology signature on a issuer-known graph* $\mathcal{G}$ *obtained during* Inspect()*, delegates to the* HiddenSign() *operation of the graph signature scheme.*

The TOPOCERT version of the HiddenSign() algorithm establishes a graph signature

for a specified encoding mapping $\mathcal{M}_{\mathsf{E}}$. *Private inputs:* Recipient R: $\mathcal{G}_{\mathsf{R}}$, commitment randomness R; Signer S: $sk_{\mathsf{S,E}}$, $\mathcal{G}_{\mathsf{S}}$. Otherwise, we refer to the interface of the graph-signature library HiddenSign() (Definition 2.10, p. 25) for the specification of other inputs and outputs.

Definition 2.5 (0 or $1 \leftarrow \mathsf{Verify}(pk_{\mathsf{S}}, C, R', \sigma)$)**.** *A verification algorithm on graph commitment C and signature σ. Delegates to the corresponding method of the graph signature scheme.*

The input/output profile is identical with Definition 2.11, p. 26 of the low-level Verify().

2.5.5 Static Architecture and Design of the TOPOCERT Tool

The TOPOCERT tool is designed such that foundational operations are delegated to the graph signature library. At the same time, the TOPOCERT tool offers methods to govern the graph encoding setup suitable for topologies.

The TOPOCERT tool implements specialized prover and verifier pairs that focus on proving particular policy predicates. For that, they drawn upon the same proof context as the graph signature library and on the values held therein (secrets, randomness, commitments).

As specialized prover, for instance, the TOPOCERT tool includes a geo-location separation proof, which depends on the graph signature library's vertex decomposition.

2.5.6 Dynamic Architecture and Design of the TOPOCERT Tool

Issuing. Figure 4 shows the dynamic flow between auditor and provider during issuing. Note that the inspection of the provider infrastructure/configuration is modeled as a synchronous communication.

Proving. Figure 5 shows the proof process between the provider and the tenant, in which the provider offers an interactive proof of knowledge (PK) on the policy predicate of the tenant. Note that the communication is asynchronous, where the nonce communicated in the first message acts as reference for the session context between activations.

Figure 6 shows a proof process, executed non-interactively. Here the proof is done as signature proof of knowledge (SPK) where a single response message is created with the Fiat-Shamir Heuristic, constituting a signature on the nonce sent in the policy predicate message.

2.6 Graph Signature Library

The graph signature library implements the corresponding signature scheme (GRS) specified by Groß [Gro15]. The library realizes the interactions between a signer and a recipient,

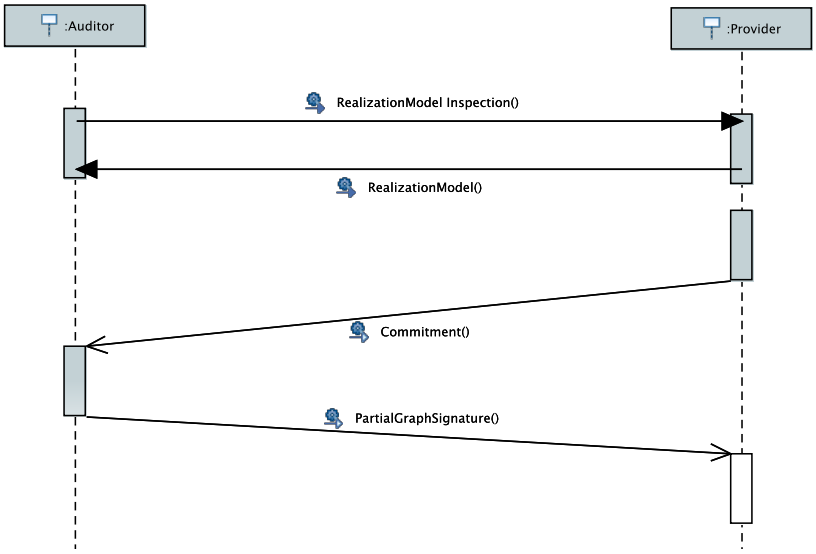

Figure 4: Sequence diagram of the issuing process between auditor and provider roles.

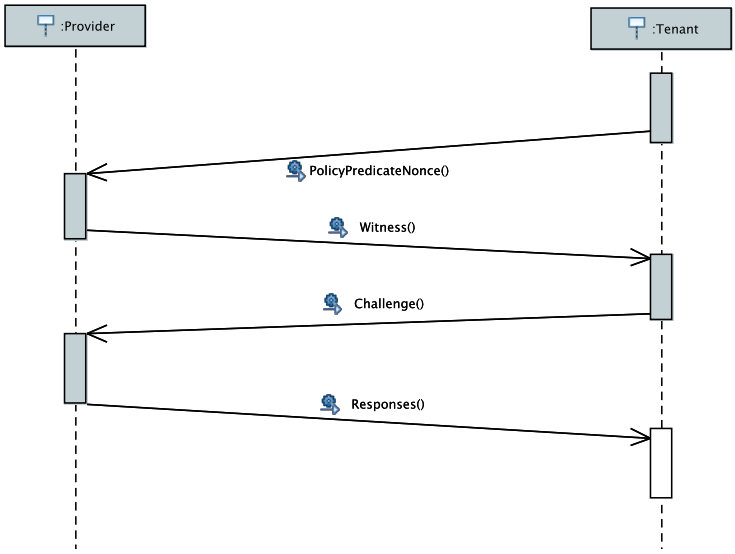

Figure 5: Sequence diagram of the proof process between provider and tenant.

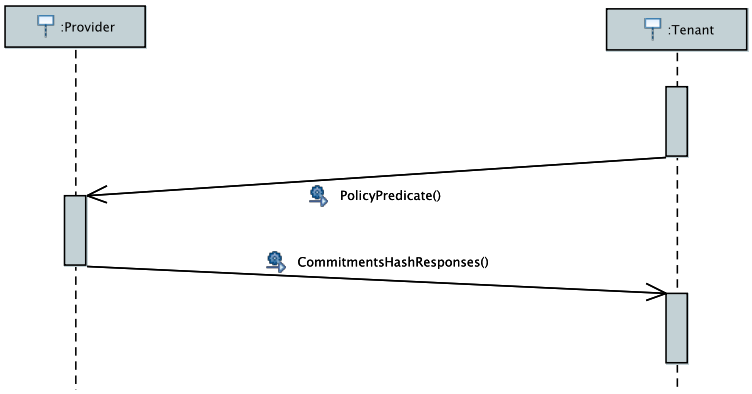

Figure 6: Sequence diagram of the proof process between provider and tenant.

meant to create a signature on a hidden committed graph, and the interactions between a prover and a verifier, meant to prove properties of a graph signature in zero-knowledge.

Graph signatures can be formed by combining a committed (hidden) sub-graph from the recipient and a issuer-known sub-graph from the signer. For the sake of the PRISMACLOUD project, it is sufficient to realize issuer-known graphs, as the graphs will be known by the signer (auditor).

2.6.1 Signer S

The signer is responsible to generate an appropriate key setup, to certify an encoding scheme, and to sign graphs. In the HiddenSign() protocol the signer accepts a graph commitment from the recipient, adds an issuer-known sub-graph and completes the signature with his secret key sk_{S}. The signer outputs a partial graph signature, subsequently completed by the recipient.

2.6.2 Recipient R

The recipient initializes the HiddenSign() protocol by creating a graph commitment and retaining randomness R, possibly only containing his master secret key, but no sub-graph. In this case, it is assumed that the signer knows the graph to be signed. Once the signer sends his partial signature, the recipient completes the signature with his randomness R.

2.6.3 Prover P

The prover role computes zero-knowledge proofs of knowledge with a policy predicate $\mathfrak{P}$ on graph signatures. These proofs can either be interactive or non-interactive.

2.6.4 Verifier V

The verifier role interacts with a prover to verify a policy predicate $\mathfrak{P}$. The verifier initializes the interaction, sending the policy predicate $\mathfrak{P}$ as well as a nonce that binds the session context.

2.6.5 Proof Context

The different prover and verifier algorithms co-create, amend and draw upon a joint proof context. The proof context is specific for a session of a zero-knowledge proof. It contains the entire proof state, that is,

- Integer and graph commitments,
- witness commitments,
- challenge,
- responses.

The prover's proof context contains additional secrets:

- The randomness of integer and graph commitments,
- the randomness corresponding to the secrets of the ZKPoK, and
- the secrets themselves (especially the actual graph and its encoding).

2.6.6 Abstract Description

Parameters. We offer the description of the parameters used for the graph signature scheme in Table 1. We use the same notation as the Identity Mixer credential system, the standard realization of the Camenisch-Lysyanskaya signature scheme [IBM13].

Core Interface. The graph signature library draws upon an interface with multiple operations. We first specify the abstract interface itself in Definition 2.6 and then discuss the inputs and outputs subsequently.

Definition 2.6 (Graph Signature Scheme)**.** *The graph signature scheme consists of the following algorithms:*

$((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}) \leftarrow \mathsf{Keygen}(1^{\lambda}, gs\_params)$ *A probabilistic polynomial-time algorithm which computes the key setup of the graph signature scheme and corresponding commitment scheme.*

$((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,es}}) \leftarrow \mathsf{GraphEncodingSetup}((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}, enc\_params)$ *A probabilistic polynomial-time algorithm which computes the setup of the graph encoding, especially, a reserved certified set of bases which are meant to hold the vertex and edge messages.*

$C \leftarrow \mathsf{Commit}(\mathcal{G}; R)$ *A probabilistic polynomial-time algorithm computing an Integer commitment on a graph.*

$(\sigma; \epsilon) \leftarrow \mathsf{HiddenSign}(pk_{\mathsf{S,E}}, C, V_{\mathsf{R}}, V_{\mathsf{S}})$ *An interactive probabilistic polynomial-time algorithm between a recipient and a signer which signs a committed graph. We note that both parties can have common inputs, namely a commitment* $C = \mathsf{Commit}(\mathcal{G}_{\mathsf{R}}; R)$ *encoding the recipient's graph and disclosed connections points* $V_{\mathsf{R}}, V_{\mathsf{S}}$*, and the Signer's extended public key* $pk_{\mathsf{S,E}}$*. Hence, the signer and the recipient can contribute subgraphs to be combined.* Private inputs: *Recipient* R*:* $\mathcal{G}_{\mathsf{R}}$*, commitment randomness* R*; Signer* S*:* $sk_{\mathsf{S,E}}$*,* $\mathcal{G}_{\mathsf{S}}$*.*

0 or 1 $\leftarrow \mathsf{Verify}(pk_{\mathsf{S}}, C, R', \sigma)$ *A verification algorithm on graph commitment* C *and signature* σ*.*

Algorithm Input/Output Specification. We define the inputs and outputs for the abstract interface as follows.

Definition 2.7 $(((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}) \leftarrow \mathsf{Keygen}(1^{\lambda}, gs\_params))$**.** *A probabilistic polynomial-time algorithm which computes the key setup of the graph signature scheme and corresponding commitment scheme.*

The key generation algorithm takes as input the general security parameter 1^{λ} and the key generation parameters of the graph signature scheme *gs_params*. The parameters are described in detail in Table 1a.

The key generation outputs a secret key sk_{S}, including the factorization of a special RSA group with modulus bit length ℓ_n and the corresponding public key pk_{S}. The both keys contain the group setups of the special RSA group as well as the group setups of the commitment group Γ.

The key generation outputs as part of the group setups a foundational generator S for the Quadratic Residues under the given Special RSA modulus QR_N. The key generation outputs a dedicated base for the Recipient's master ket R_0.

The key generation digitally signs the given public outputs and makes the signature $\sigma_{\mathsf{S,kg}}$ public.

The parameters specified in *gs_params* are stored for this instantiation of the Signer S.

Definition 2.8 $(((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,es}}) \leftarrow \mathsf{GraphEncodingSetup}((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}, enc\_params))$**.** *A probabilistic polynomial-time algorithm which computes the setup of the graph encoding, especially, a reserved certified set of bases which are meant to hold the vertex and edge messages.*

The graph encoding setup takes as input the Signer S's secret key sk_{S}, public key pk_{S} as well as the encoding setup parameters *enc_params*. Table 1b offers a details on the corresponding parameters.

As public output, the algorithm produces a number of bases reserved to hold vertex and edge encodings, $R_i|i \in \{1, \ldots, \ell_{\mathcal{V}}\}$ for vertices and $R_j|j \in \{1, \ldots, \ell_{\mathcal{E}}\}$ for edges.

The algorithm digitally signs the generators, proving knowledge of their representation and binding them to public key pk_{S}. It outputs signature $\sigma_{\mathsf{S,es}}$.

We call the certified combination of original public key pk_{S} and the vertex and edge encoding bases R_i, R_j the Signer's *extended public key* $pk_{\mathsf{S,E}}$.

As private output, the algorithm returns the discrete logarithms of all produced bases with respect to generator S, $\mathsf{log}_S(R_k)$. The discrete logarithms stored securely persistently and retained for the graph signing process.

Corresponding to the definition of the public key, we call the combination of original secret key sk_{S} and the discrete logarithms $\mathsf{log}_S(R_k)$ the Signer's *extended secret key* $sk_{\mathsf{S,E}}$

Definition 2.9 $(C \leftarrow \mathsf{Commit}(\mathcal{G}; R))$**.** *A probabilistic polynomial-time algorithm computing an Integer commitment on a graph.*

The $\mathsf{Commit}()$ algorithm takes a graph $\mathcal{G}$ and randomness R as input.

It commits to the graph in an appropriate encoding, that is, holding vertex and edge representations in different bases. As specified in the graph signature definition [Gro15], the algorithm will establish a commitment as follows:

$$C = \underbrace{\cdots R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k}}_{\forall \text{ vertices } i} \underbrace{\cdots\cdots R_{\pi(i,j)}^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k} \cdots}_{\forall \text{ edges } (i,j)} S^r \bmod N,$$

where e_i and e_j are vertex representatives. The label representatives e_k are obtained with the vertex mappings $f_{\mathcal{V}}(i)$ and edge mappings $f_{\mathcal{E}}(i, j)$.

The public output C is the computed commitment. The Committer retains the randomness R for future commitment opening or proofs of representation.

Definition 2.10 $((\epsilon; \sigma) \leftarrow \mathsf{HiddenSign}(pk_{\mathsf{S,E}}, C, V_{\mathsf{R}}, V_{\mathsf{S}}))$**.** *An interactive probabilistic polynomial-time algorithm between a recipient and a signer which signs a committed graph. We note that both parties can have common inputs, namely a commitment* $C = \mathsf{Commit}(\mathcal{G}_{\mathsf{R}}; R)$ *encoding the recipient's graph and disclosed connections points* $V_{\mathsf{R}}, V_{\mathsf{S}}$*, and the Signer's extended public key* $pk_{\mathsf{S,E}}$*. Hence, the signer and the recipient can contribute sub-graphs to be combined.* Private inputs: *Recipient* R*:* $\mathcal{G}_{\mathsf{R}}$*, commitment randomness* R*; Signer* S*:* $sk_{\mathsf{S,E}}$*,* $\mathcal{G}_{\mathsf{S}}$*.*

The abstract interface specification for the interactive algorithm decomposes into two interfaces for Signer S and Recipient R.

$$\mathsf{Signer.HiddenSign}(pk_{\mathsf{S,E}}, C, V_{\mathsf{R}}, V_{\mathsf{S}}; sk_{\mathsf{S,E}}, \mathcal{G}_{\mathsf{S}}), \text{and}$$

$$\mathsf{Recipient.HiddenSign}(pk_{\mathsf{S,E}}, C, V_{\mathsf{R}}, V_{\mathsf{S}}; \mathcal{G}_{\mathsf{R}}, R).$$

Let us first discuss public and private inputs. While the Integer commitment C is publicly known, it is usually computed by the Recipient R for the the given $\mathsf{HiddenSign}()$ operation with the corresponding randomness R with $\mathsf{Commit}()$. The Recipient will be required to offer a proof of representation of the commitment as part of the interactive protocol.

Note that the key-pair inputs $(pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}})$ refer to extended keys, that is, public and private keys that contain the information on encoding bases of $\mathsf{GraphEncodingSetup}()$.

While the $\mathsf{HiddenSign}()$ algorithm allows for either both private input graphs $\mathcal{G}_{\mathsf{R}}$ and $\mathcal{G}_{\mathsf{S}}$ to be present or absent, the standard case realized in TOPOCERT is that we have a signer-known graph $\mathcal{G}_{\mathsf{S}}$, but no hidden/committed graph of the Recipient R.

In most cases the connection points V_{R} and V_{S} will be equal, but that is not necessary.[1]

As output on the Recipient side, R obtains a graph signature $\sigma_{\mathsf{S},\mathcal{G}}$ (or short σ) on the combined graph $\mathcal{G} = \mathcal{G}_{\mathsf{R}} \cup \mathcal{G}_{\mathsf{S}}$ valid with respect to $pk_{\mathsf{S,E}}$.

The Signer S does not produce an output.

Definition 2.11 $(0 \text{ or } 1 \leftarrow \mathsf{Verify}(pk_{\mathsf{S,E}}, C, R', \sigma))$**.** *A verification algorithm on graph commitment C and signature σ.*

The algorithm $\mathsf{Verify}()$ takes as inputs the Signer's extended public key $pk_{\mathsf{S,E}}$, a signed graph commitment C and its randomness R' and the graph signature σ.

The algorithm outputs either 0 or 1, signifying that σ is either invalid or valid as graph signature on C.

We note that usually graph signatures, such as σ are used in proof of possessions and further zero-knowledge proofs of knowledge to show certain properties. In such cases, we can use $\mathsf{Verify}()$ to signify a proof of representation that the secrets encoded in commitment C are equal to the secret messages of the graph signature σ.

Then we have a zero-knowledge proof of knowledge defined as follows:

$$\begin{aligned}
&PK\{(e_i, e_j, e_k, e, v, r) : \\
&\quad Z \equiv \pm \underbrace{\cdots R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} \cdots\cdots}_{\forall \text{ vertices } i} \underbrace{R_{\pi(i,j)}^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k} \cdots}_{\forall \text{ edges } (i,j)} A^e S^v \bmod N \\
&\quad C \equiv \pm \underbrace{\cdots R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} \cdots\cdots}_{\forall \text{ vertices } i} \underbrace{R_{\pi(i,j)}^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k} \cdots}_{\forall \text{ edges } (i,j)} S^r \bmod N \\
&\}.
\end{aligned}$$

Here the first equation proves the representation of the graph signature σ and the second

[1] The first graph signature [Gro15] proposal referred to the connection points as $\mathcal{V}_{\mathsf{R}}$, $\mathcal{V}_{\mathsf{S}}$, which would mean the set of all vertices and not a subset.

equation proves the representation of the commitment C, yielding equality over the secrets e_i, e_j and e_k.

We note that the proofs of knowledge on graph properties between prover and verifier are specified as standard Σ-proofs.

2.6.7 Static Architecture and Design of the Library

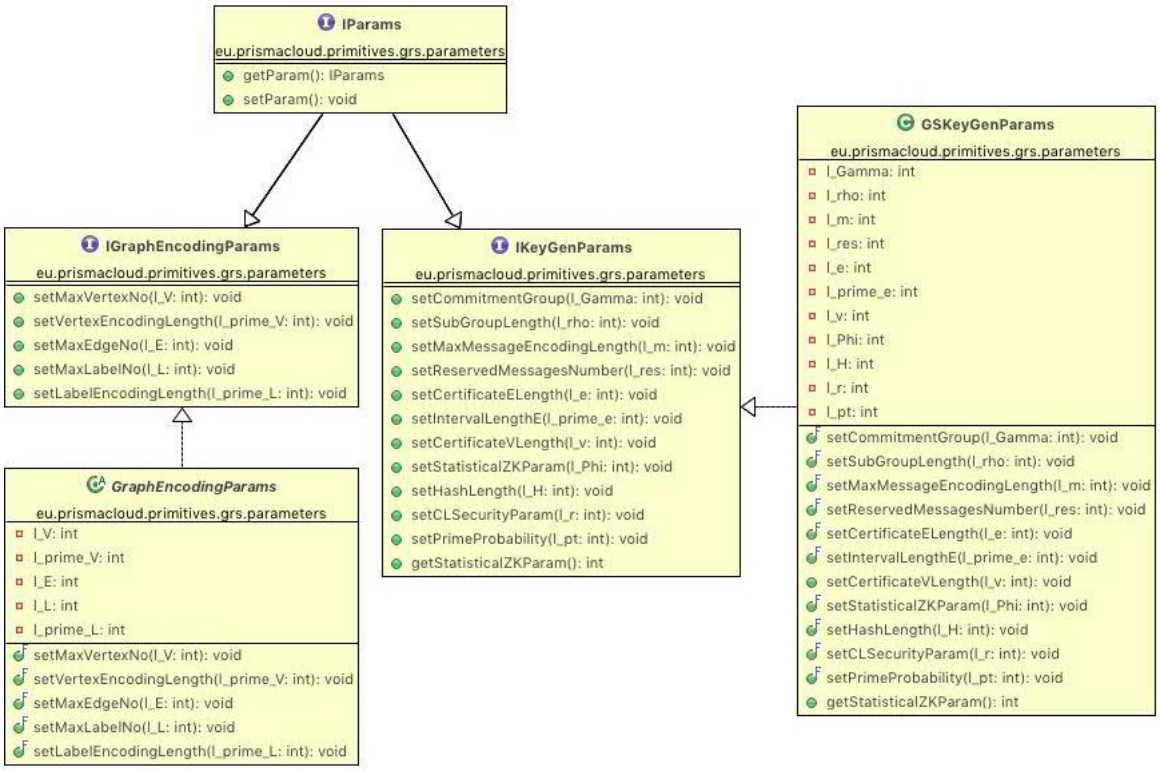

Figure 7: Parameters Class Diagram

In this section, we present the architecture of the graph signature library related to the above abstract description. The following class diagrams present the main interfaces and classes that comprise the graph signature library.

Figure 7 presents the interface hierarchy of the parameters used for the key generation and the graph encoding. First, we create a generic interface called `IParams` that can used to create and get parameters required for the library. Second, we use two different interfaces for the key generation and encoding parameters. Third, the `GSKeyGenParams` and `GraphEncodingParams` concrete classes in the diagram implement the creation of the parameters for the graph encoding and for the key generation.

Figure 8 shows the key generation hierarchy for the library. First, we create an interface for the key generation and one for the extended key pairs. Each of the classes implementing a key pair interface acts as a wrapper encapsulating the private and public keys

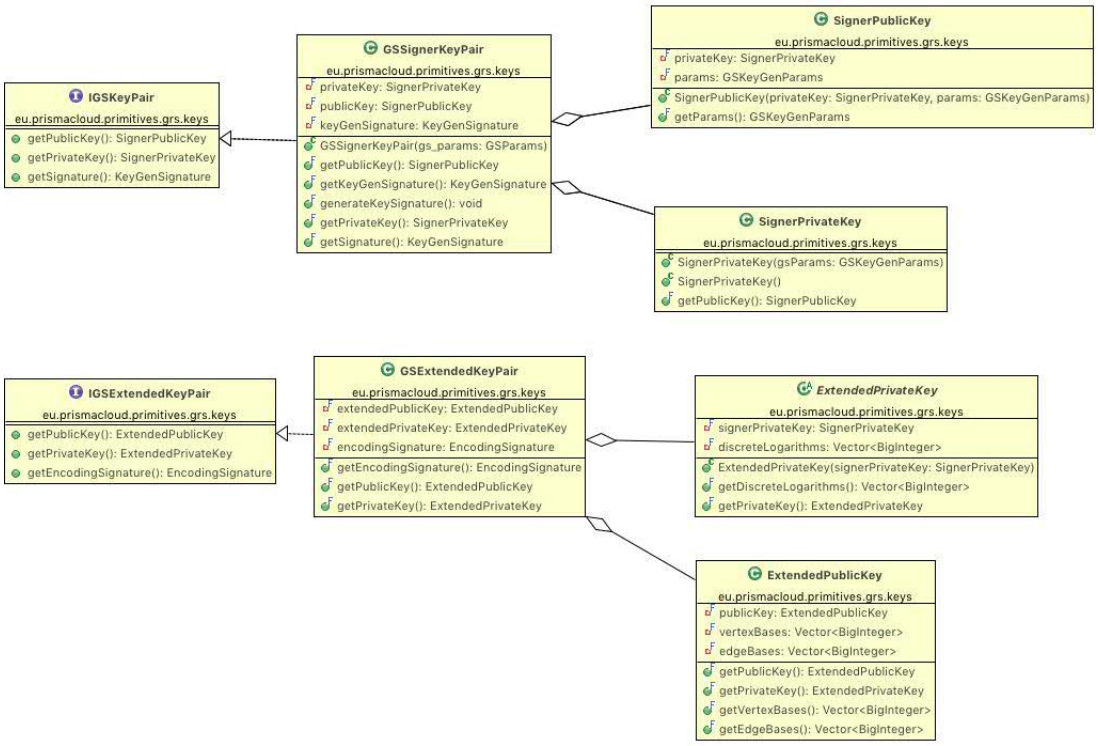

Figure 8: Key pairs Class Diagram

as shown in the classes `SignerPublicKey`, `SignerPrivateKey`, `ExtendedPublicKey` and `ExtendedPrivateKey` and their respective private and public outputs following the abstract description of the graph signature library.

Figure 9 illustrates the architecture for the recipient and the signer. The recipient uses first an interface (`IRecipient`) where we outline the main methods a concrete class must implement. `GSRecipient` is the concrete class that implements the required methods and includes the graph methods. The `GSGraph` and `GSVertex` classes act as wrappers for the JGrapht library for the creation and manipulation of graphs.

The signer part of the architecture uses a similar hierarchy. First an interface called `ISigner`) is used where we outline the main methods a concrete class must implement. `GSSigner` is the concrete class that implements the required methods and includes the graph methods in the same fashion as the recipient.

The library is designed such that there are dedicated pairs of provers and verifiers for the purposes of the graph signature protocol. All provers and verifiers draw upon and enrich the proof context with the values they have derived. Subsequent proofs that rely on computed values of provers/verifiers in the dependency tree call upon the proof context to obtain these values. Consequently, the proof context is the lynchpin to hold session state and tie different component provers together.

The *Proof of Possession* prover only shows that the prover indeed owns a graph signature

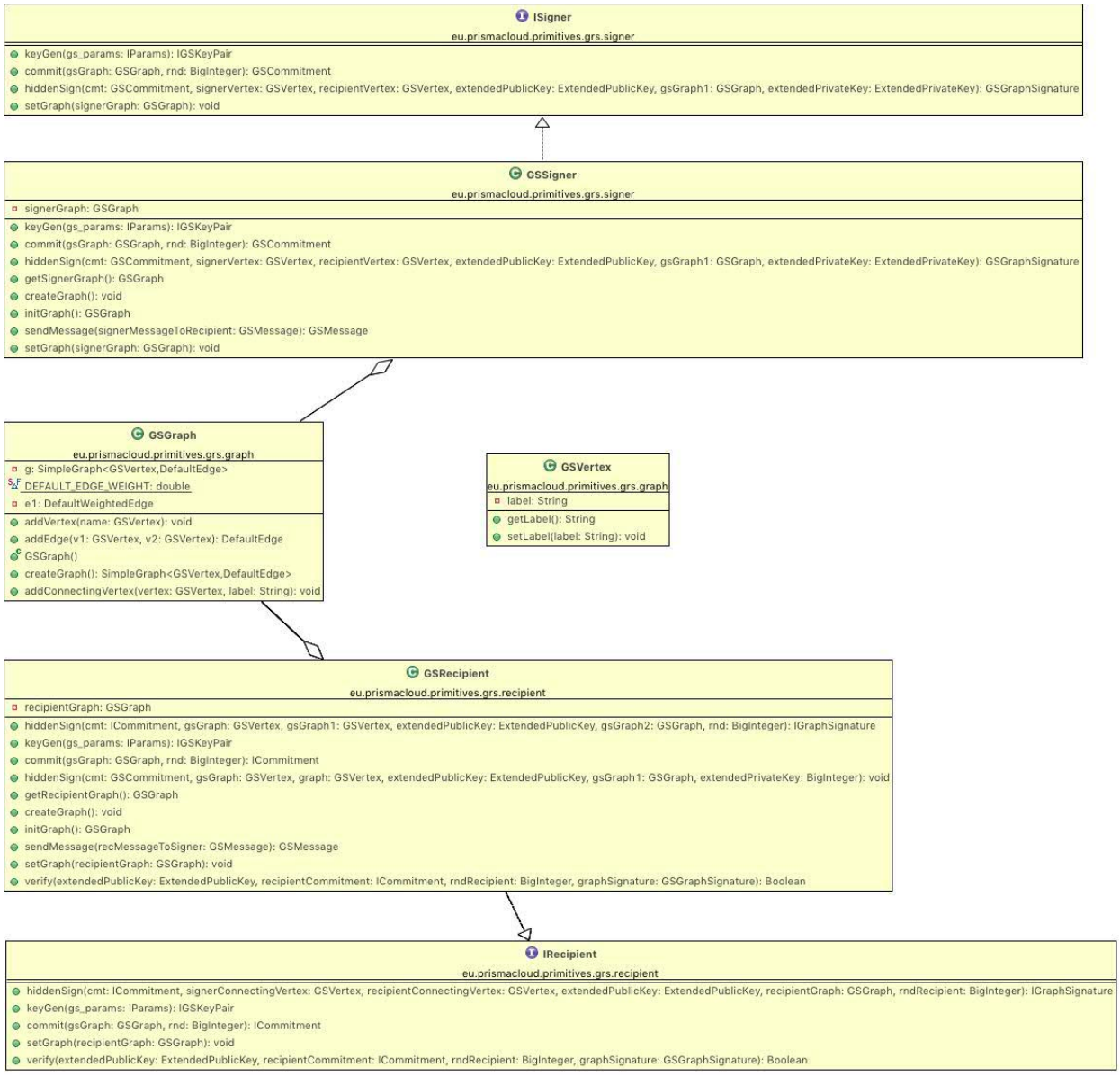

Figure 9: Recipient and Signer Class Diagram

and knows all corresponding secrets. As an operationalization of the graph signature, the prover can compute and transfer integer commitments on all message attributes of the graph signature. A *Proof of Vertex Composition* prover computes integer commitments that separate vertex identifiers and labels and proves their correct decomposition. A *Proof of Edge Composition* prover computes integer commitments that separate the edges' vertex identifiers and labels and proves their correct decomposition.

2.7 Recommendations

To securely use the graph signature scheme and the TOPOCERT tool, it is necessary to follow key size requirements specified for the Identity Mixer cryptographic library [IBM13]. Here we note that the TOPOCERT tool is meant to operate in an implementation with a 2048-bits key strength and appropriately selected parameters, which implies parameters as defined in Table 1. For a detailed specification of the parameter selection for the underlying Camenisch-Lysyanskaya signature scheme, we refer to Tables 2 and 3 of the Specification of the Identity Mixer Cryptographic Library, Version 2.3.40, on p. 43 [IBM13].

Remark 1 (Security Parameter). The security parameters, especially bit length for the group setups, flow from the specification of the bit length of the special RSA modulus ℓ_n and the message space ℓ_m. The constraints placed on the respective bit lengths are crucial to maintain the soundness of the security proof of the underlying Camenisch-Lysyanskaya signature scheme (cf. Table 3 of the Specification of the Identity Mixer Cryptographic Library, Version 2.3.40, on p. 43 [IBM13]).

Remark 2 (Encoding Parameters). We consider the choices made for the graph encoding scheme.

Encoding Defaults The bit length parameters for the prime encoding $\ell'_{\mathcal{V}}$ and $\ell'_{\mathcal{L}}$ follow from the available message bit length, assuming that the labels are encoded as the lowest prime representatives. However, the given defaults for number of vertices, edges, labels to be encoded $\ell_{\mathcal{V}}$, $\ell_{\mathcal{E}}$, and $\ell_{\mathcal{L}}$ are not the theoretical maxima.

Maximal Number of Labels For a single-labeled graph with $\ell'_{\mathcal{L}} = 16$, the maximal encodable number of labels is 6542. The restrictions of the number of labels is in place to allow for multi-labeled graphs, in which case the product of the label identifiers occupies the reserved space.

Maximal Number of Vertices The maximal number of vertices for the reserved bit length $\ell'_{\mathcal{V}} = 120$ is 1.59810^{34}. The limiting factor for the number of encoded vertices, however, is *not* the reserved bit length of the message space, but the space required to store the corresponding based dedicated vertex and edge encoding. For each possibly encodable vertex and edge the graph signature scheme needs to reserve a group element with an bit length of $\ell_n = 2048$. A encoding for fully connected graphs with $\ell_{\mathcal{V}} = 1000$ and $\ell_{\mathcal{E}} = \ell_{\mathcal{V}}(\ell_{\mathcal{V}} - 1) = 999000$ would consume 244.28 kBytes for vertices and 243.89 MBytes for the edges.

Remark 3 (Signature Size). A signature of the graph signature scheme consists of one group element and two exponents A, e, v). A single signature has the following bit length for the default parameters in Table 1:

$$|(A, e, v)|_2 = \ell_n + \ell_v + \ell_e = 5369 \text{ bits.}$$

Remark 4 (Base Randomization). We note here that the graph signature scheme proposed by Groß [Gro15] requires a base randomization for multi-use confidentiality of graph elements. This is because the bases referenced in the ZKPoK are public knowledge and each proof reveals which exponents are harbored by which base.

Table 1: Parameters of the graph signature scheme *gs_params* and encoding setup *enc_params*. Parameters for the underlying Camenisch-Lysyanskaya signature scheme are largely adapted from the Identity Mixer Specification [IBM13]. In the implementation, this table is referred to as `table:params`.

(a) Parameters of Keygen()

Parameter	Description	Bit-length
ℓ_n	Bit length of the special RSA modulus	2048
ℓ_Γ	Bit length of the commitment group	1632
ℓ_ρ	Bit length of the prime order of the subgroup of Γ	256
ℓ_m	Maximal bit length of messages encoding vertices and edges	256
ℓ_{res}	Number of reserved messages	1†
ℓ_e	Bit length of the certificate component e	597
ℓ'_e	Bit length of the interval the e values are taken from	120
ℓ_v	Bit length of the certificate component v	2724
$\ell_\varnothing$	Security parameter for statistical zero-knowledge	80
ℓ_H	Bit length of the cryptographic hash function used for the Fiat-Shamir Heuristic	256
ℓ_r	Security parameter for the security proof of the CL-scheme	80
ℓ_{pt}	The prime number generation to have an error probability to return a composite of $1 - 1/2^{\ell_{pt}}$	80†

Note: † refers to numbers that are integers, not bit lengths.

(b) Parameters of GraphEncodingSetup()

$\ell_\mathcal{V}$	Maximal number of vertices to be encoded	1000†‡
$\ell'_\mathcal{V}$	Reserved bit length for vertex encoding (bit length of the largest encodable prime representative)	120
$\ell_\mathcal{E}$	Maximal number of edges to be encoded	50.000†‡
$\ell_\mathcal{L}$	Maximal number of labels to be encoded	256†‡
$\ell'_\mathcal{L}$	Reserved bit length for label encoding	16

Note: † refers to numbers that are integers, not bit lengths; ‡ refers to the default parameter, not the theoretical maximum.

The base randomization asks that random permutations $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{E}}$ be applied to the vertex and edge bases respectively. A space-efficient solution for that requirement could use keyed pseudorandom permutations.

Let an appropriate family of pseudorandom permutations $\mathcal{F}$ on group elements in QR_N be given, where pseudorandom permutations (PRPs) are defined as by Katz and Lindell [KL14]. Theoretical work on constructions of pseudorandom permutations from pseudorandom functions was spawned by the seminal work of Luby and Rackoff [LR88]. As an alternative approach, we also refer to constructions of Verifiable Secret Shuffles, such as Neff [Nef01], which allow a Prover in a honest-verifier zero-knowledge proof scenario to convince a Verifier that a secret shuffled was computed correctly.

1. During the Signer's round of HiddenSign(), S chooses a uniformly random permutation key k with appropriate bit length.
2. S applies the pseudorandom permutations $(\pi_{\mathcal{V}}, \pi_{\mathcal{E}})$ with common key k to the certified base sets, obtaining permuted base sets.
3. Signer S then encodes graph $\mathcal{G}$ on the derived base sets.
4. Signer S shares permutation key k with the Recipient together with the corresponding signature $\sigma_k = (A, e, v)_k$ along with a proof of representation that σ_k indeed fulfills the CL-equation on the derived bases.

Hence, the Signer will issue multiple signatures, one for each permutation. Each signature has a size of one group element, two exponents, and one permutation key.

3 Geo-Separation

3.1 Overview

The geo-location separation [Gro17] first specified in deliverable D4.7 aims at certifying a topology graph in such a way that a Provider can convince a Tenant with the TOPOCERT tool that the resources of a Tenant are distributed over different geo-locations, for example multiple European countries.

The geo-separation scenario is built upon the Abstract Interface of the TOPOCERT tool, as defined in Section 2.5.4 on p. 18.

3.1.1 Our Contribution

We are the first to propose a geo-separation proof on graphs realized with the graph signature scheme proposed by Groß [Gro15]. The geo-separation scenario demonstrates the utility of the graph signature scheme and the TOPOCERT tool in the wider context of PRISMACLOUD.

Our contribution contains the specification of an encoding scheme for TOPOCERT as well as of the corresponding zero-knowledge proofs of knowledge between Provider and Tenant.

3.1.2 State-of-the-Art

Previous proposals on graph signatures offered predicates to prove Graph 3-Colorability [Gro15] or predicates such as connectivity and isolation [Gro14]. They all use prime numbers as representatives to encode graphs such that their constituent components (vertices, edges and labels) remain accessible to discrete-logarithm-based zero-knowledge proofs of knowledge. The underlying graph signature scheme [Gro15] is based on the Strong-RSA version of the Camenisch-Lysyanskaya signature scheme [CL02].

3.2 Preliminaries and Building Blocks

The geo-separation protocol builds directly upon the TOPOCERT tool defined in Section 2.5.4. It relies on a special mapping $\mathcal{M}_{\mathsf{E}}$ suitable for a geo-location encoding of a finite list of countries. In this scenario, the TOPOCERT Auditor is responsible to map GPC coordinates into the appropriate United Nations country code.

Furthermore, the system is based on discrete-logarithm-based proofs of knowledge that prove pair-wise co-primality of committed exponents without disclosing information about those exponents. While these techniques are well-known in the field, they have been notably employed in the NOT-proofs of the Camenisch-Groß encoding scheme for anonymous credentials [CG08, CG12] as well as in work on Credential Authenticate Key Exchange and Identification (CAKE/CAID) [CCGS10].

3.2.1 Our Framework

Setup We assume that the key generation of the TOPOCERT tool Keygen() has already been completed successfully, yielding a certified graph signature key pair $((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}})$.

We continue to specify the message space which will eventually yield the graph encoding mapping $\mathcal{M}_{\mathsf{E,geo}}$. As the labels set $\mathcal{L}$ we specify the set of 249 United Nations country codes as defined in `ISO 3166-1 alpha-2` [ISO06] in alphabetical order.

The country codes are encoded into label prime representatives in the low-bit range, occupying the prime numbers from 2 to 1579 inclusive, at a maximal bit length of 11 bits. $\ell_{\mathcal{L}}$ is set to 249. Note that this can be realized with the default setup of TOPOCERT as outlined in Table 1 on p. 31.

The vertex set $\mathcal{V}$ and its maximal size $\ell_{\mathcal{V}}$ is set to fit the needs of the application scenario, yet not exceeding the bit length $\ell'_{\mathcal{V}}$. Otherwise the standard parameters of TOPOCERT are adopted.

The graph encoding is produced with a call to TOPOCERT algorithm:

$$((pk_{\mathsf{S,E}}, sk_{\mathsf{S,E}}), \sigma_{\mathsf{S,me}}, \mathcal{M}_{\mathsf{E,geo}}) \leftarrow \mathsf{GraphEncodingSetup}((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}}, \mathcal{V}, \mathcal{L}, enc\_params),$$

taking as inputs the described parameters and the given certified key pair $((pk_{\mathsf{S}}, sk_{\mathsf{S}}), \sigma_{\mathsf{S,kg}})$. The encoding mapping $\mathcal{M}_{\mathsf{E,geo}}$ is part of the certified extended public key $pk_{\mathsf{S,E}}$, which is then distributed to Provider and Tenant roles with integrity.

Certification For the certification operation by the Auditor, we assume that we are only dealing with a Signer-known graph $\mathcal{G}$, which is obtained through the Auditor's inspection. The Auditor determines for each physical server in the virtualized infrastructure represented in $\mathcal{G}$ the geo-location as per trusted GPS coordinates and systematically maps this geo-location into `ISO 3166-1 alpha-2` [ISO06] country codes.

To obtain a graph signature, the Provider initiates the interactive protocol by creating a commitment C on the Provider's master secret key and inputting that into the Provider's side of the TOPOCERT.HiddenSign() algorithm.

$$\mathsf{Recipient.HiddenSign}(pk_{\mathsf{S,E}}, C, \emptyset, \emptyset; \epsilon, R)$$

Given this protocol initiation, commitment C will be transferred to the Auditor with a zero-knowledge proof of representation. Receiving that, the Auditor initiates its part of the protocol:

$$\mathsf{Signer.HiddenSign}(pk_{\mathsf{S,E}}, C, \emptyset, \emptyset; sk_{\mathsf{S,E}}, \mathcal{G}_{\mathsf{S}})$$

The Auditor then uses the encoding mapping $\mathcal{M}_{\mathsf{E,geo}}$ to look-up the prime identifier for each country label and associates the labels with the vertices of all physical machines.

The Auditor encodes this information in message blocks of the graph signature scheme along with the unlabeled edges in $\mathcal{G}$. Once, the encoding is complete, the Auditor derives

the pre-signature, which is then securely communicated back to the Provider along with a proof of representation and the encoding of $\mathcal{G}$.

Remark 5 (Simplifying Assumptions). For the proof of geo-separation as demonstrated in PRISMACLOUD, we assume that the Auditor enforces that vertex identifiers uniquely identify resources. For instance, if there is a particular server identified as e_i, there could not be another server of the same name. A violation of this rule will only yield false positive alarms, i.e., a TOPOCERT proof outcome yielding that the geo-separation is not fulfilled, even if it is. There will be no false negatives. Making this assumption allows for streamlined computations, saving on a complex vertex-decomposition into a tree of commitments.

Proof of Geo-Separation The proof of geo-separation is initiated by a Tenant asking for a proof for a geo-separation predicate while submitting a nonce.

Upon receiving such a predicate, the Provider computes commitments on all vertex messages:

$$C_i = R^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} S_i^r \bmod N$$

. The provider, then, establishes (either interactively or non-interactively) a zero-knowledge proof of knowledge as follows:

$$\begin{aligned}
&PK\{(e_i, e_j, e_k, e, v, r_i, a_{i,j}, b_{i,j}, r_{i,j}) : \\
&\quad Z \equiv \pm \underbrace{\cdots R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k}}_{\forall \text{ vertices } i} \underbrace{\cdots\cdots R_{\pi(i,j)}^{e_i e_j} \cdots}_{\forall \text{ edges } (i,j)} A^e S^v \bmod N \\
&\quad C_i \equiv \pm \underbrace{R^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} S_i^r \bmod N}_{\forall \text{ vertices } i} \\
&\quad R \equiv \pm \underbrace{C_i^{a_{i,j}} C_j^{b_{i,j}} S^{r_{i,j}} \bmod N}_{\forall \text{ lower-triangle pairs } i,j} \\
&\quad \}.
\end{aligned}$$

The first equation proves the possession of the graph signature. The second equation proves the representation of the commitments C_i as well as the equality with the message exponents in the graph signature. Finally, the third equation yields a proof of co-primality between the different vertex commitments, through Bézout's identity:

$$\begin{aligned}
1 &= a_{i,j} m_i + b_{i,j} m_j \\
&= a_{i,j} (e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k) + b_{i,j} (e_i \Pi_{k \in f_{\mathcal{V}}(j)} e_k)
\end{aligned}$$

Factors $a_{i,j}$ and $b_{i,j}$ only exist if m_i and m_j are co-prime, which further entails that their labels must be co-prime.

Consequently, after having verified this zero-knowledge proof of knowledge, the Tenant will be convinced that the physical servers hosting the Tenant's resources will be in separate geo-locations, without learning which countries harbor the servers.

4 The TOPOCERT Tool in the Application Context

In this section, we first briefly set our architecture in the context of the pilot applications within PRISMACLOUD. Then we present our research on additional applications and also discuss how those applications would comply with the architecture of our framework.

4.1 The e-Goverment Pilot

In the e-Government use case a regional government IT company called LISPA delivers IaaS services to customers such as municipalities and other local public bodies. Customers can use this service to setup their infrastructure. We aim to use the TOPOCERT tool in this use case to allow the regional government IT company to certify its infrastructure and to prove policy predicates to public bodies without disclosing the layout of its infrastructure. The policy predicate supported by TOPOCERT tool is the geo-location separation. Using this policy predicate a public body can be assured for instance, that its physical servers and virtual machines reside in different countries.

Further implementation details reside in deliverables D7.6 and D7.8 for the architecture of the Infrastructure Auditing (IA) service that is used in the e-Goverment use case. The TOPOCERT tool plays a central role to the IA service as it supports all the crucial low-level operations for certification of virtualized infrastructures and proving geo-location separation.

4.2 Research on Additional Applications

4.2.1 Geo-location

While we have specified a scheme for geo-location separation in Section 3 meant for virtualized infrastructures, we have also investigated graph signatures and corresponding proofs on graphs of geo-locations and their routes. Furthermore, we have investigated the use of attribute-based credentials on GPS-based geo-locations to enable a direct binding between the geo-location coordinates and the graph signatures.

4.2.2 Future Work

Future research directions include:

- certification of software-defined networks,
- provenance graphs (e.g., in the Open Provenance Model [MFF+08]), and
- causality representations (e.g. Structured Occurrence Nets [KR09]).

Such future applications will be investigated in the European Research Council (ERC) Starting Grant Confidentiality-Preserving Security Assurance (CASCAde, GA n°716980).

5 Conclusion

In this deliverable, we have presented our final design for the TOPOCERT tool. We have established a flexible design with an easy to use API which is compatible with all the requirements imposed by the Prismacloud pilots. While the design itself represents a final iteration of the design already presented in D5.6, we have presented additional research on applications going beyond the use cases within the Prismacloud pilots, and assessed the possibilities of an integration of the primitives required by those extended application scenarios.

6 Appendix

In the following we have attached the publications that relate to Sections 2 and 3 of this deliverable.

Signatures and Efficient Proofs on Committed Graphs and NP-Statements

Thomas Groß
School of Computing Science, Newcastle University, UK
thomas.gross@ncl.ac.uk

No Institute Given

Abstract. Digital signature schemes are a foundational building block enabling integrity and non-repudiation. We propose a graph signature scheme and corresponding proofs that allow a prover (1) to obtain a signature on a committed graph and (2) to subsequently prove to a verifier knowledge of such a graph signature. The graph signature scheme and proofs are a building block for certification systems that need to establish graph properties in zero-knowledge, as encountered in cloud security assurance or provenance. We extend the Camenisch-Lysyanskaya (CL) signature scheme to graphs and enable efficient zero-knowledge proofs of knowledge on graph signatures, notably supporting complex statements on graph elements. Our method is based on honest-verifier proofs and the strong RSA assumption. In addition, we explore the capabilities of graph signatures by establishing a proof system on graph 3-colorability (G3C). As G3C is NP-complete, we conclude that there exist Camenisch-Lysyanskaya proof systems for statements of NP languages.

1 Introduction

Digital signature schemes are foundational cryptographic primitives; they are useful in themselves to ensure integrity and non-repudiation and as building block of other systems. From their first construction by Rivest, Shamir and Adleman [26], digital signatures have been on bit-strings or group elements, on a committed sequence of bit-strings [10] or structure-preserved group elements [1]. In this work, we establish a signature scheme and corresponding proof system for committed graphs.

The basis for this work is the Camenisch-Lysyanskaya proof system: a collection of distributed algorithms that allow an issuer, a prover and a verifier to prove knowledge of committed values, issue a Camenisch-Lysyanskaya (CL) signature [9,10] on committed values, and prove knowledge of such a signature in zero-knowledge, while selectively disclosing values or proving statements about them. It uses honest-verifier Σ-proofs (Schnorr proofs [27]) and has the advantage that it keeps all attributes in the exponent. It thereby allows us to access attributes with known discrete-logarithm-based zero-knowledge proofs of knowledge [27,16,18,11,4,13]. The attributes that could be signed are, however, limited by the message space of the CL-signature scheme: a sequence of small bit-strings.

We study how to extend the Camenisch-Lysyanskaya proof system to establish signatures on committed graphs and, by extension, on committed statements from NP

languages. Zero-knowledge proofs of certified or committed graphs with the capability of selective disclosure of graph elements or complex statements over graph attributes have many significant applications beyond classical graph proof techniques [21,3] or the more recent proposal of transitive signatures [23]. The key difference to earlier work is that the graph encoding is universal, enables direct access to graph elements, and allows a prover to be flexible in the statements proven after the graph is certified. Such graph proofs are instrumental in foundational techniques, such as the zero-knowledge proof of knowledge of certified Petri nets as well as in various application scenarios, such as for the certification of audited cloud topologies for which we proposed a dedicated framework for topology proofs [2]. including coverage, disjointness and partitions as well as connectivity and isolation.

First, we establish a new encoding of undirected graphs into the message space of CL-Signatures. The encoding allows for unlabeled, vertex- or edge-labeled graphs. The graph encoding is universal and operational in the sense that it supports efficient proofs over graph elements (vertices, edges, labels) and their relations.

Second, we extend the Camenisch-Lysyanskaya proof system to graphs by integrating the graph encoding into integer commitments and the CL-Signature bootstrapping process. This allows prover and issuer to sign committed graphs with sub-graphs contributed by both parties and to prove knowledge of graph signatures in honest-verifier Σ-proofs. The obtained graph proof system in itself allows for efficient zero-knowledge proofs of interesting graph properties, such as partitions, connectivity and isolation [2], already demonstrated in an application scenario of topology certification and proofs in virtualized infrastructures. Graph proofs with a level of indirection between the authority on the graph (the issuer) and the verifier, established by a graph signature and with access to a wide range of graph properties, have not been covered by existing zero-knowledge graph proofs, such as [21,3,20], or transitive signatures [23]. While the former graph proofs are powerful constructions allowing for NP statements, e.g., graph 3-colorability or directed Hamiltonian cycle, their encoding does not cater for proving relations over graph elements in zero-knowledge. The latter is focused on the transitive closure along graph edges.

Third, we establish a proof system for graph 3-colorability (G3C) that allows us to obtain CL-Signatures on committed instances of 3-colorable graphs and to prove knowledge thereof to a verifier in zero-knowledge. Given that graph 3-colorability is NP-complete, we can lift the Camenisch-Lysyanskaya proof system to NP statements. Based on the 3-colorability proof system in a special RSA group and under the Strong RSA assumption, we show that there exists a Camenisch-Lysyanskaya proof system for any NP language, that is, the proof is capable of issuing CL-Signatures on committed statements from the NP language and to prove knowledge of such signatures in honest-verifier Σ-proofs. Whereas the G3C-reduction does not offer particularly efficient constructions for graph proofs, it shows the theoretical expressiveness of the graph credential system.

In effect, this work extends the reach of the Camenisch-Lysyanskaya proof system to signatures and proofs on structures of entire systems. To our knowledge, it is the first work to enable signatures on committed graphs. Notably, the graph elements are

present in the exponents and, thereby, accessible to known discrete-logarithm-based zero-knowledge proofs on a wide range graph properties in honest-verifier proofs.

1.1 Outline

In §2, we discuss the preliminaries of our graph proof construction: Camenisch-Lysyanskaya signatures and Camenisch-Groß encoding. Based on the Camenisch-Groß encoding, we establish a canonical encoding for vertex- and edge-labeled graphs in §3. §4 establishes how integer commitments and CL-Signature are extended with the graph encoding. In §5 we show how graph 3-colorability can be expressed in the graph proof system as proof of the encoding's theoretical reach. §7 considers earlier work on zero-knowledge proofs and signatures on graphs, while §8 draws conclusions of this work's properties.

2 Preliminaries

2.1 Assumptions

Special RSA Modulus A *special RSA modulus* has the form $N = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, the corresponding group is called *special RSA group*. *Strong RSA Assumption [26,18]:* Given an RSA modulus N and a random element $g \in \mathbb{Z}_N^*$, it is hard to compute $h \in \mathbb{Z}_N^*$ and integer $e > 1$ such that $h^e \equiv g \bmod N$. The modulus N is of a special form pq, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. *Quadratic Residues* The set QR_N is the set of Quadratic Residues of a special RSA group with modulus N.

2.2 Integer Commitments

Damgård and Fujisaki [16] showed for the Pedersen commitment scheme [24] that if it operates in a special RSA group and the committer is not privy to the factorization of the modulus, then the commitment scheme can be used to commit to *integers* of arbitrary size. The commitment scheme is information-theoretically hiding and computationally binding. The security parameter is ℓ. The public parameters are a group G with special RSA modulus N, and generators $(g_0, \ldots, g_m)$ of the cyclic subgroup QR_N. In order to commit to the values $(V_1, \ldots, V_l) \in (\mathbb{Z}_n^*)^l$, pick a random $R \in \{0,1\}^\ell$ and set $C = g_0^R \prod_{i=1}^{l} g_i^{v_i}$.

2.3 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [27] or a composite [16,18], (2) proof of knowledge of equality of representation modulo two (possibly different) composite [11] moduli, (3) proof that a commitment opens to the product of two other committed values [5,11], (4) proof that a committed value lies in a given integer interval [4,11], and also (5) proof of the

disjunction or conjunction of any two of the previous [15]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption.

Proofs as described above can be expressed in the notation introduced by Camenisch and Stadler [12]. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^{\alpha} h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta} \wedge (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers* α*,* β*, and* δ *such that* $y = g^{\alpha} h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta}$ *holds, where* $u \leq \alpha \leq v$*,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities of which knowledge is being proven, while all other values are known to the verifier. We apply the Fiat-Shamir heuristic [17] to turn such proofs of knowledge into signatures on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^{\alpha}\}(m)$. Given a protocol in this notation, it is straightforward to derive an actual protocol implementing the proof.

2.4 Camenisch-Lysyanskaya Signatures

Let us introduce Camenisch-Lysyanskaya (CL) signatures in a Strong RSA setting [10].

Let $\ell_{\mathcal{M}}$, ℓ_e, ℓ_N, ℓ_r and L be system parameters; ℓ_r is a security parameter, $\ell_{\mathcal{M}}$ the message length, ℓ_e the length of the Strong RSA problem instance prime exponent, ℓ_N the size of the special RSA modulus. The scheme operates with a ℓ_N-bit special RSA modulus. Choose, uniformly at random, $R_0, \ldots, R_{L-1}, S, Z \in \mathrm{QR}_N$. The public key $\mathsf{pk}(\mathsf{I})$ is $(N, R_0, \ldots, R_{L-1}, S, Z)$, the private key $\mathsf{sk}(\mathsf{I})$ the factorization of the special RSA modulus. The *message space* is the set $\{(m_0, \ldots, m_{L-1}) : m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}\}$.

Signing hidden messages. On input $m_0, \ldots, m_{L-1}$, choose a random prime number e of length $\ell_e > \ell_{\mathcal{M}} + 2$, and a random number v of length $\ell_v = \ell_N + \ell_{\mathcal{M}} + \ell_r$. To sign hidden messages, user U commits to values V in an integer commitment C and proves knowledge of the representation of the commitment. The issuer I verifies the structure of C and signs the commitment:

$$A = \left(\frac{Z}{C R_l^{m_l} \ldots R_{L-1}^{m_{L-1}} S^{v'}} \right)^{1/e} \bmod N.$$

The user completes the signature as follows: $\sigma = (e, A, v) = (e, A, (v' + R))$.

To verify that the tuple (e, A, v) is a signature on message $(m_0, \ldots, m_{L-1})$, check that the following statements hold: $Z \equiv A^e R_0^{m_0} \ldots R_{L-1}^{m_{L-1}} S^v \pmod N$, $m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}$, and $2^{\ell_e} > e > 2^{\ell_e - 1}$ holds.

Theorem 1. *[10] The signature scheme is secure against adaptive chosen message attacks under the strong RSA assumption.*

Proving Knowledge of a Signature. The prover randomizes A: Given a signature (A, e, v), the tuple $(A' := AS^{-r} \bmod N, e, v' := v + er)$ is also a valid signature as well. Now, provided that $A \in \langle S \rangle$ and that r is chosen uniformly at random from $\{0,1\}^{\ell_N + \ell_\varnothing}$, the value A' is distributed statistically close to uniform over $\mathbb{Z}_N^*$. Thus, the user could compute a fresh A' each time, reveal it, and then run the protocol

$$\begin{aligned} &PK\{(\varepsilon, \nu', \mu_0, \ldots, \mu_{L-1}) : \\ &\quad Z \equiv \pm R_0^{\mu_0} \cdots R_{L-1}^{\mu_{L-1}} A'^{\varepsilon} S^{\nu'} \pmod N \ \wedge \\ &\quad \mu_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}} \ \wedge \ \varepsilon \in [2^{\ell_e - 1} + 1, 2^{\ell_e} - 1]\} \end{aligned}$$

2.5 Set Membership from CL-Signatures

Set membership proofs can be constructed from CL-Signatures following a method proposed by Camenisch, Chaabouni and shelat [7]. For a set $\mathcal{S} = \{m_0, \ldots, m_i, \ldots, m_l\}$, the issuer signs all set members m_i in CL-Signatures $\sigma_i = (A, e, v)$ and publishes the set of message-signature pairs $\{(m_i, \sigma_i)\}$ with integrity. To prove set membership of a value committed in C, the prover shows knowledge of the blinded signature σ_i' corresponding to the message m_i and equality of exponents with C. We explain this technique in detail in the extended version of this paper and denote a set membership proof $\mu[C] \in \mathcal{S}$, which reads μ encoded in commitment C is member of set $\mathcal{S}$.

2.6 Camenisch-Groß Encoding

The Camenisch-Groß (CG) Encoding [8] establishes structure on the CL message space by encoding multiple binary and finite-set values into a single message, and we will use a similar paradigm to encode graphs efficiently. We explain the key principles briefly and give more details in the extended version of this paper.

The core principle of the CG-Encoding is to represent binary and finite-set attribute values as prime numbers. It uses divisibility and coprimality to show whether an attribute value is present in or absent from a credential. The attribute values certified in a credential, say e_i, e_j, and e_l, are represented in a single message of the CL-Signature, by signing the product of their prime representative $E = e_i \cdot e_j \cdot e_l$ in an Integer attribute. The association between the value and the prime number of the encoding is certified by the credential issuer.

Divisibility/AND-Proof. To prove that a disclosed prime representative e_i is present in E, we prove that e_i divides the committed product E, we show that we know a secret μ' that completes the product:

$$PK\{(\mu', \rho) : \qquad D \equiv \pm(g^{e_i})^{\mu'} h^{\rho} \pmod N\}.$$

Coprimality/NOT-Proof. We show that one or multiple prime representatives are not present in a credential, we show coprimality. To prove that two values E and F are coprime, i.e., $\mathsf{gcd}(E, F) = 1$, we prove there exist integers a and b such that Bézout's Identity equals 1, where a and b for this equation do not exist, if $\mathsf{gcd}(E, F) > 1$.

$$PK\{(\mu, \rho, \alpha, \beta, \rho') : \qquad D \equiv \pm g^{\mu} h^{\rho} \pmod N \ \wedge \ g \equiv \pm D^{\alpha} (g^F)^{\beta} h^{\rho'} \pmod N\}.$$

OR-Proof To show that a credential contains an attribute e that is contained in an OR-list, we show there exists an integer a such that $ae = \prod_i^\ell e_i$; if e is not in the list, then there is no such integer a as e does not divide the product. We use the notation $\alpha \subseteq \Xi$ for an OR-proof that α contains one or more values of Ξ.

3 Graph Encoding

We consider graphs over finite vertex sets, with undirected edges or directed arcs, and finite sets of vertex and edge labels. Vertices and edges may be associated with multiple labels. We leave the encoding of directed arcs to the extended version of this paper.

$\mathcal{V}$	Finite set of vertices
$\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$	Finite set of edges
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, t_\mathcal{V}, t_\mathcal{E})$	Graph
$\mathcal{L}_\mathcal{V}, \mathcal{L}_\mathcal{E}$	Finite sets of vertex and edge labels
$f_\mathcal{V} : \mathcal{V} \to \mathcal{P}(\mathcal{L}_\mathcal{V})$	Labels of a given vertex
$f_\mathcal{E} : \mathcal{E} \to \mathcal{P}(\mathcal{L}_\mathcal{E})$	Labels of a given edge
$n = \lvert\mathcal{V}\rvert, m = \lvert\mathcal{E}\rvert$	Number of vertices and edges

For each vertex i in $\mathcal{V}$, we introduce a vertex identifier, a prime e_i, which represents this vertex in credential and proofs. The symbol $\perp$, associated with identifier $e_\perp$ represents that a vertex is not present. All vertex identifiers are pair-wise different. We call the set of all vertex identifiers $\Xi_\mathcal{V}$, their product $\chi_\mathcal{V} = \Pi\Xi_\mathcal{V}$. For each label k in the label sets $\mathcal{L}_\mathcal{V}$ and in $\mathcal{L}_\mathcal{E}$, we introduce a prime representative e_k. All label representatives are pair-wise different. We call the set of all label representatives $\Xi_\mathcal{L}$, their product $\chi_\mathcal{L} = \Pi\Xi_\mathcal{L}$. Vertex identifiers and label representatives are disjoint:

$$\Xi_\mathcal{V} \cap \Xi_\mathcal{L} = \emptyset \quad \Leftrightarrow \quad \mathsf{gcd}(\chi_\mathcal{V}, \chi_\mathcal{L}) = 1.$$

Random Base Association We encode vertices and edges into the exponents of integer commitments and CL-Signatures and make them therefore accessible to proofs of linear equations over exponents. We randomize the base association to vertices and edges: For a vertex index set $\mathcal{V}$= 0,...,i,n-1 with vertex identifiers e_i, we choose a uniformly random permutation $\pi_\mathcal{V}$ of set $\mathcal{V}$ to determine the base $R_{\pi(i)}$ to encode vertex i. Edge bases $R_{\pi(i,j)}$ are chosen analogously with a random permutation $\pi_\mathcal{E}$.

Encoding Vertices To encode a vertex and its associated labels into a graph commitment or CL-Signature, we encode the product of the vertex identifier $e_i \in \Xi_\mathcal{V}$ and the prime representatives $e_k \in \Xi_\mathcal{L}$ for $k \in f_\mathcal{V}(i)$ of the labels into a single of the signature message. The product of prime representatives is encoded as exponent of dedicated vertex bases $R \in G_\mathcal{V}$.

Encoding Edges To get a compact encoding and efficient proofs thereon, the encoding needs to maintain the graph structure and to allow us to access it to proof higher-level properties, such as connectivity and isolation. The proposal we make in this paper after

Table 1. Interface of the graph signature scheme.

$\mathsf{Commit}(\mathcal{G}; R)$	A PPT algorithm computing an Integer commitment on a graph.
$\mathsf{Keygen}(1^\ell, params)$	A PPT algorithm computing the key setup.
$\mathsf{HiddenSign}(C, \mathcal{V}_\mathsf{U}, \mathcal{V}_\mathsf{I}, pk_\mathsf{I})$	An interactive PPT algorithm signing a committed graph.
Private inputs:	User U: $\mathcal{G}_\mathsf{U}$, commitment randomness R; Issuer I: $\mathcal{G}_\mathsf{I}$, sk_I.
$\mathsf{Verify}(pk_\mathsf{I}, C, R', \sigma)$	A verification algorithm on graph commitment C and signature σ.

evaluating multiple approaches is to use divisibility and coprimality similar to the CG-Encoding to afford us these efficient operations over the graph structure, while offering a compact encoding of edges.

Recall that each vertex is certified with an vertex identifier from $\Xi_\mathcal{V}$, e.g., e_i or e_j. For each edge $(i, j) \in \mathcal{E}$, we include an edge attribute as exponent of a random edge base $R_{\pi(i,j)} \in G_\mathcal{E}$, containing the product of the vertex identifiers and the associated label representatives $e_k \in \Xi_\mathcal{L}$ for $k \in f_\mathcal{E}(i, j)$ of the edge:

$$E_{(i,j)} := e_i \cdot e_j \cdot \Pi_{k \in f_\mathcal{E}(i,j)} e_k.$$

Whereas we usually consider simple graphs, specialties such as multigraphs, loops (i, i) encoded as e_i^2 or half-edges encoded as $(e_j, e_\perp)$ can be included.

Well-formed Graphs

Definition 1 (Well-formed graph). *We call a graph encoding* well-formed *iff 1. the encoding only contains prime representatives $e \in \Xi_\mathcal{V} \cup \Xi_\mathcal{L}$ in the exponents of designated vertex and edge bases $R \in G_\mathcal{V} \cup G_\mathcal{E}$, 2. each vertex base $R \in G_\mathcal{V}$ contains exactly one vertex identifier $e_i \in \Xi_\mathcal{V}$, pair-wise different from other vertex identifiers and zero or more label representatives $e_k \in \Xi_\mathcal{L}$, and 3. each edge base $R \in G_\mathcal{E}$ contains exactly two vertex identifiers $e_i, e_i \in \Xi_\mathcal{V}$ and zero or more label representatives $e_k \in \Xi_\mathcal{L}$.*

Theorem 2 (Unambiguous encoding and decoding). *A well-formed graph encoding on the integers is unambiguous modulo the base association. [Proof A.1]*

4 Signatures on Committed Graphs

CL-signatures are signatures on committed messages, where messages can be contributed by issuer and user. This translates to a user committing to a hidden partial graph $\mathcal{G}_\mathsf{U}$, which is then completed by the issuer $\mathcal{G}_\mathsf{I}$, as outline in the interface in Table 1. We establish the setup for the construction first, explain the proof of representation second, and the issuing third. We discuss notions of secrecy and imperfections of this construction in §4.1.

As a point of reference, we give the structure of the graph signatures first. We have bases $R_{\pi(i)} \in G_\mathcal{V}$, which store attributes encoding vertices, and bases $R_{\pi(i,j)} \in G_\mathcal{E}$,

which store attributes encoding edges. Observe that which base stores which vertex or edge is randomized by permutations $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{E}}$.

$$Z = \underbrace{\cdots R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} \cdots}_{\forall \text{ vertices } i} \underbrace{\cdots R_{\pi(i,j)}^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k} \cdots}_{\forall \text{ edges } (i,j)} A^e S^v \bmod N$$

4.1 Secrecy Notion

In a *known-graph* proof, the structure of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an auxiliary input to the verifier. Such a proof occurs if the prover needs to prove knowledge of a (NP-hard) property of the entire graph, e.g., a proper coloring in graph 3-colorability (cf. §5.1).

A *hidden-graph* proof keeps the structure of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ secret. For instance, there are graph proofs in which a local property is proven and the graph structure itself kept secret, e.g., when proving that disclosed vertices of the graph are connected by a hidden path.

The number of bases from $\mathcal{G}_{\mathcal{V}}$ and $\mathcal{G}_{\mathcal{E}}$ in a CL-Signature reveals an upper-bound on the number of vertices n and edges m of the signed graph. A suitable padding can be introduced by encoding nil-vertices $e_\perp$ and nil-edges $(e_\perp, e_\perp)$.

Proving properties over multiple attributes reveals which bases were involved in the proof. Characteristic patterns over said bases may interfere with the CL-Signature's multi-use unlinkability. For instance, if the prover shows that vertices i and j are connected by an edge (i, j) along with properties on the vertices themselves, the verifier will learn that the bases for the vertex identifiers e_i and e_j are related to the base for the encoding of edge (i, j). To overcome this linking, the prover can obtain a collection of CL-Signatures on the same graph, each with a randomized association between bases and vertices/edges, that is, using different random permutations $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{E}}$. When proving a property over the graph the prover chooses a CL-Signature from the collection uniformly at random and proves possession over that instance.

4.2 Proof of Representation

For a full proof of representation, we need to establish that the encoded graph in a graph commitment or CL-Signature is indeed well-formed (Def. 1). Given a graph commitment C the prover and verifier engage in the following proof of representation (the proof for a CL credential work analogously). We show that vertex bases contain a bi-partition of one and only one vertex identifier $e_i \in \Xi_{\mathcal{V}}$ and a set of labels $e_l \in \Xi_{\mathcal{L}}$. Edge bases contain a bi-partition of a product of exactly two vertex identifiers $(e_i \cdot e_j)$ and a set of labels $e_l \in \Xi_{\mathcal{L}}$. To prove that the representation contains exactly one vertex identifier for a vertex base and two vertex identifiers for an edge base, we establish a set membership proof.

1. Commitments The prover computes Integer commitments on the exponents of all vertex and edge bases. For each vertex i and for each edge (i, j), the prover computes commitments on vertex attribute and identifier (all $\bmod N$)::

$$C_i = R^{e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k} S^r \quad \text{and} \quad \breve{C}_i = R^{e_i} S^{\breve{r}};$$

$$C_{(i,j)} = R^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k} S^{r}, \quad \breve{C}_{(i,j)} = R^{e_i e_j} S^{\breve{r}} \quad \text{and} \quad \dot{C}_i = R^{e_i} S^{\dot{r}}.$$

2. Proof of knowledge. We build up the proof of possession and well-formedness step by step, where it is understood the proofs will be done in one compound proof of knowledge with *referential integrity between the secret exponents*. Let us consider a proof fragment for vertices i, j and an edge (i, j) committed in a graph commitment C (the same proof structure is used for CL-Signatures).

2.1 Proof of representation. We prove that commitment C can be decomposed into commitments C_i, C_j, one for each vertex i, j and one commitment $C_{(i,j)}$ for each edge (i, j):

$$\begin{aligned} PK\{&(\mu_i, \mu_j, \mu_{(i,j)}, \rho, \rho_i, \rho_j, \rho_{(i,j)}) : \\ & C \equiv \pm \cdots R_{\pi(i)}^{\mu_i} \cdots R_{\pi(j)}^{\mu_j} \cdots R_{\pi(i,j)}^{\mu_{(i,j)}} \cdots S^{\rho} \pmod N \ \wedge && (1) \\ & C_i \equiv \pm R^{\mu_i} S^{\rho_i} \pmod N \ \wedge \ C_j \equiv \pm R^{\mu_j} S^{\rho_j} \pmod N \ \wedge && (2) \\ & C_{(i,j)} \equiv \pm R^{\mu_{(i,j)}} S^{\rho_{(i,j)}} \pmod N\}. && (3) \end{aligned}$$

2.2 Vertex composition. Second, we need to show properties of the vertex composition, that the encoding for each vertex i contains exactly one vertex identifier $e_i \in \Xi_{\mathcal{V}}$ and zero or multiple label representatives $e_k \in \Xi_{\mathcal{L}}$. We show this structure with help of the commitments $\breve{C}_i$ and set membership and prime-encoding OR proofs. This proof is executed for all vertices.

$$\begin{aligned} PK\{&(\varepsilon_i, \breve{\rho}_i, \gamma_i, \rho_i') : \\ & \breve{C}_i \equiv \pm R^{\varepsilon_i} S^{\breve{\rho}_i} \pmod N \ \wedge \ C_i \equiv \pm \breve{C}^{\gamma_i} S^{\rho_i'} \pmod N \ \wedge && (4) \\ & \gamma_i[C_i] \subseteq \Xi_{\mathcal{L}} \ \wedge \ \varepsilon_i[\breve{C}_i] \in \Xi_{\mathcal{V}}\}. && (5) \end{aligned}$$

2.3 Edge composition. Third, we prove the structure of each edge (i, j) over the commitments $C_{(i,j)}$, showing that each commitment contains exactly two vertex identifiers $e_i, e_j \in \Xi_{\mathcal{V}}$ as well as zero or more label representative $e_k \in \Xi_{\mathcal{L}}$:

$$\begin{aligned} PK\{&(\varepsilon_j, \rho_{(i,j)}, \gamma_{(i,j)}, \rho'_{(i,j)}) : \\ & \breve{C}_{(i,j)} \equiv \pm \dot{C}_i^{\varepsilon_j} S^{\rho_{(i,j)}} \pmod N \ \wedge && (6) \\ & C_{(i,j)} \equiv \pm \breve{C}_{(i,j)}^{\gamma_{(i,j)}} S^{\rho'_{(i,j)}} \pmod N \ \wedge && (7) \\ & \gamma_{i,j} \subseteq \Xi_{\mathcal{L}}\}. && (8) \end{aligned}$$

2.4 Pair-wise difference. Finally, we prove pair-wise difference of vertices by showing that the vertex representatives are pair-wise co-prime over the commitments $\breve{C}_i$ and $\breve{C}_j$.

$$PK\{(\forall i, j : \alpha_{i,j}, \beta_{i,j}, \rho_{i,j}) : \quad R \equiv \pm \breve{C}_i^{\alpha_{i,j}} \breve{C}_j^{\beta_{i,j}} S^{\rho_{i,j}} \pmod N\}. \qquad (9)$$

Theorem 3 (Proof of Well-formedness). *The compound proof of knowledge establishes the well-formedness of an encoded graph according to Def. 1. [Proof B]*

4.3 Joint Graph Issuing

To jointly issue a graph CL-signature, a user commits to a hidden partial graph and the issuer adds further elements to the graph (cf. §2.4)

In the setup, the issuer establishes a user vertex space and issuer vertex space, i.e., a bi-partition on vertex and edge bases, $G_{\mathcal{V}}$ and $G_{\mathcal{E}}$ and on vertex identifiers $\Xi_{\mathcal{V}}$. Thus, user and issuer can encode partial graphs without interfering with each other.

In the joint graph issuing, user and issuer designate and disclose connection points (vertex identifiers) that allow the user and the issuer to connect their sub-graphs deliberately. The user constructs a graph representation by choosing two uniformly random permutation $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{E}}$ for the base association on the user bases and commits to his sub-graph in a graph commitment. The user interacts with the issuer in a proof of representation of his committed sub-graph. The issuer verifies this proof, chooses uniformly random permutations for his graph elements and encodes them into his base range. The issuer creates the pre-signature of the CL-Signature scheme on the entire graph, proving that the added sub-graph is well-formed. The user completes the CL-Signature with his own randomness.

Theorem 4 (Security of graph signatures). *The graph signature scheme maintains confidentiality and integrity of the encoded graphs and offers existential unforgeability against adaptive chosen message attacks under the strong RSA assumption.[Proof A.1]*

5 Graph 3-Colorability and NP Statements

5.1 Graph 3-Colorability

We adapt the following definition from Goldreich, Micali and Wigderson [21].

Definition 2 (Graph 3-Colorability). *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be* 3-colorable *if there exists a vertex label mapping $f_{\mathcal{V}} : \mathcal{V} \to \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$ called proper coloring such that every two adjacent vertices are assigned different color labels. This means that for each edge $(i, j) \in \mathcal{E}$ $f_{\mathcal{V}}(i) \neq f_{\mathcal{V}}(j)$. The language graph 3-colorability, denoted G3C, consists of the set of undirected graphs that are 3-colorable. Graph 3-Colorability is known to be NP-complete. [19]*

We adapt the graph 3-colorability problem to show in honest-verifier zero-knowledge that the prover knows an CL signature on an instance of a proper coloring of a given graph $\mathcal{G}$.

Without loss of generality, we assume that graph $\mathcal{G}$ is simple and connected. The three color labels $\mathcal{L} = \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$ are encoded with three primes $\Xi_{\mathcal{L}} = \{e_{\mathsf{R}}, e_{\mathsf{G}}, e_{\mathsf{B}}\}$. The graph is encoded with vertex identifiers $\Xi_{\mathcal{V}}$ and these vertex labels. In addition to the conditions for a well-formed graph (Def. 1), we require that each vertex base contains exactly one label representative from $\Xi_{\mathcal{L}}$, which we show with a set membership proof on the secret vertex label.

The prover shows knowledge of a proper graph coloring by showing that the product of vertex identifiers and label representatives for each pair of adjacent vertices (i, j) are coprime.

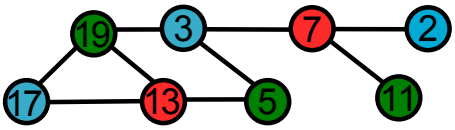

Fig. 1. Example of a 3-colored graph, where the vertex identifiers are prime numbers, and the labels R,G, and B are represented as colors.

Common inputs: Graph $\mathcal{G}$, public-key of the CL-issuer.
Prover input: CL-Signature on proper coloring for G3C.

1. Credential randomization and commitments. The prover computes randomizations for the graph signature as well as for all occurrences of set membership proofs. The prover computes Integer commitments on the exponents of all vertex and edge bases. For each vertex i, the prover computes two commitments on the vertex attribute and the vertex identifier:

$$C_i = R^{e_i e_{f_\mathcal{V}(i)}} S^r \bmod N \quad \text{and} \quad \breve{C}_i = R^{e_i} S^r \bmod N.$$

For each edge (i, j), the prover computes the commitment:

$$\breve{C}_{i,j} = R^{e_i e_j} S^r \bmod N.$$

2. Proof of knowledge. The prover sends the commitments to the verifier. Then, prover and verifier engage in the following proof of possession over the graph signature and vertices i and j and all edges (i, j). We build upon the proof of representation and well-formedness presented in §4.2 with the following differences: Instead of proving that a vertex contains zero or multiple labels, we prove that the vertex contains *exactly one label*. Further, the proof is simplified because the edges do not contain labels. Again, we explain the proofs step by step, while it is understood that the proofs are executed as compound proof of knowledge *with referential integrity between the secret exponents*.

2.1 Possession of CL-Signature. First, we prove of possession of the graph signature and representation of the commitments. Clause 1 proves possession of the CL-Signature on the graph. The clauses 2 and 3 prove the representation on the integer commitments on signed attributes for vertices j, j and edges (i, j), and, thereby, make the attributes accessible for the analysis of the exponents.

$$\begin{aligned}
&PK\{(\mu_i, \mu_j, \mu_{(i,j)}, \varepsilon, \nu', \rho_i, \rho_j, \rho_{(i,j)}) : \\
&\quad Z \equiv \pm \cdots R_{\pi(i)}^{\mu_i} \cdots R_{\pi(j)}^{\mu_j} \cdots R_{\pi(i,j)}^{\mu_{(i,j)}} \cdots (A')^{\varepsilon} S^{\nu'} \pmod N \ \wedge && (1)\\
&\quad C_i \equiv \pm R^{\mu_i} S^{\rho_i} \pmod N \ \wedge \ C_j \equiv \pm R^{\mu_j} S^{\rho_j} \pmod N \ \wedge && (2)\\
&\quad C_{(i,j)} \equiv \pm R^{\mu_{(i,j)}} S^{\rho_{(i,j)}} \pmod N \ \wedge && (3)\\
&\quad \mu_i, \mu_j, \mu_{((i,j))} \in \pm\{0,1\}^{\ell_\mathcal{M}} \ \wedge \ \varepsilon \in [2^{\ell_e - 1} + 1, 2^{\ell_e} - 1]\}
\end{aligned}$$

2.2 Well-formedness. Second, we establish that the vertex attributes are well-formed: Clause 4 establishes the relation between C_i and $\breve{C}_i$ and, thereby, shows that a

vertex attribute is bi-partitioned onto a vertex identifier and a label representative part. Clause 5 establishes that they contain exactly one vertex identifier and label representative of the certified sets $\Xi_{\mathcal{V}}$ and $\Xi_{\mathcal{L}}$.

$$PK\{(\varepsilon_i, \rho_i, \gamma_i, \breve{\rho}_i) :$$
$$\breve{C}_i \equiv \pm R^{\varepsilon_i} S^{\rho_i} \pmod{N} \ \wedge \ C_i \equiv \pm \breve{C}^{\gamma_i} S^{\breve{\rho}_i} \pmod{N} \ \wedge \tag{4}$$
$$\gamma_i[C_i] \in \Xi_{\mathcal{L}} \ \wedge \ \varepsilon_i[\breve{C}_i] \in \Xi_{\mathcal{V}}\}. \tag{5}$$

Clause 5 is different from a proof of well-formedness as introduced in §4.2, as it enforces that that vertex i contains exactly one label.

2.3 Proper coloring. Third, clauses 6 and 7 complete the statement by establishing that there is a proper coloring for the adjacent vertices i and j: Clause 6 shows that commitment $C_{(i,j)}$ is on an edge (i, j). Finally, Clause 7 establishes that the attributes for vertex i and j are coprime, by proving that Bézout's Identity equals 1. It follows that the labels of both vertices must be different.

$$PK\{(\varepsilon_i, \rho'_{(i,j)}, \alpha_{(i,j)}, \beta_{(i,j)}, \rho_{(i,j)''}) :$$
$$\breve{C}_{(i,j)} \equiv \pm \breve{C}_j^{\varepsilon_i} S^{\rho'_{(i,j)}} \pmod{N} \ \wedge \tag{6}$$
$$R \equiv \pm C_i^{\alpha_{(i,j)}} C_j^{\beta_{(i,j)}} S^{\rho''_{(i,j)}} \pmod{N}\}. \tag{7}$$

3. Verification. The verifier outputs *accept* if the proof of knowledge checks out; *reject* otherwise.

Lemma 1 (Knowledge of a CL-Signature of G3C). *The prover convinces the verifier in zero-knowledge that the prover knows a proper graph 3-coloring for known graph $\mathcal{G}$. [Proof B.1]*

Lemma 2. *The proof has an asymptotic computation complexity of $O(n+m)$ exponentiations and a communication complexity of $O(n+m)$ group elements and is thereby a polynomial time proof. [Proof B.1]*

5.2 Proofs Systems for Languages in NP

Having established a proof for certified graph 3-colorability, we can use the fact that G3C is NP-complete to establish that such Camenisch-Lysyanskaya proof systems exist for statements from other NP languages.

Definition 3. *We call a* Camenisch-Lysyanskaya proof system *a set of PPT machines Prover* P*, Verifier* V *and Issuer* I *that engage in the following protocols:*

Proof of representation $\mathsf{P} \rightarrow \mathsf{I}$: *Proof of representation on committed values V.*
Issuing $\mathsf{I} \rightarrow \mathsf{P}$: *Issuing of CL-Signature σ on hidden committed values V.*
Proof of possession $\mathsf{P} \rightarrow \mathsf{V}$: *Proof of possession of CL-Signature σ.*

The issuer I *can act in the role of the verifier* V *and thereby allow the bootstrapping of further CL-Signatures from the hidden v alues of existing CL-Signatures.*

Compared to a zero-knowledge proof system for an NP language, this construction offers a level of indirection: The issuer acts as auditor with authority to decide whether the statement of an NP language is fulfilled in a certain environment, and its signature binds this statement to that environment. The instance of the NP language can either be provided by the issuer or provided by the prover and verified by the issuer.

The proof follows the same strategy as one of the initial results that all languages in NP have zero-knowledge proof systems, by Goldreich, Micali and Widgerson [21]: Given a CL proof system for G3C, we use the existing poly-time NP reductions to transform any NP language statement into an instance of G3C. This instance is then encoded as a graph in a CL-Signature and knowledge of the signature proven to a verifier. Lemma 1 shows that this is a zero-knowledge proof of knowledge of a proper coloring.

Theorem 5. *Statements of languages in NP can efficiently be proven in a Camenisch-Lysyanskaya proof system based in honest-verifier zero-knowledge.* *[Proof B.2]*

6 Efficiency Analysis

We display the efficiency analysis for the proof predicates in Table 2, where vertex and edge composition proofs show the overhead over the basic proof of possession (cf. topology proofs [2]). We measure computational complexity in multi-base exponentiations. The communication complexity is dominated by the transmitted group elements from $\mathbb{Z}_N^*$, which is equal to the number of multi-base exponentiations (one for each Integer and Schnorr proof commitment). The most expensive proof is the complete graph representation established in the issuing, where the set membership proofs (4 MExps) and the OR-based subset proofs (6 MExps) constitute significant overhead. The square-complexity is introduced by the final disjointness proof to establish that the graph is indeed well-formed. In the down-stream proofs, the verifier trusts the issuer to only certify well-formed graphs, which allows us to reduce complexity by only the computing the proof of possession and the statement proven.

The modular exponentiations for message bases R_i are with small exponents of size of $\ell_{\mathcal{M}} \ll \ell_N$, where the parameter $\ell_{\mathcal{M}}$ can be chosen similarly small as in Direct Anonymous Attestation (DAA) [6].

In addition, the Σ-proofs employed in this work benefit from batch-proof techniques, such as [25]. The graph proofs are likely to be transformed to signature proofs of knowledge with the Fiat-Shamir heuristic [17] and can thereby be computed offline.

We have evaluated the system experimentally in [2], in computations using components of the Identity Mixer Library [22] with modulus length $\ell_n = 2048$ bits and default system parameters (ℓ_v, etc.). The performance analysis is executed on 64-bit Java JDK 1.7.13 on a Windows 7 SP 1 Thinkpad X220 Tablet, on Intel CPU i5-2520 with 2.5 GHz, 8 GB RAM, where all computations are performed on a *single processor core only*, a very conservative setup. Figure 2 contains the results of a prototypical implementation of computations of the graph signature scheme, on representative computations of commitments and a proof of knowledge thereof. Based on uniform random bit-strings of the prescribed length and number (as in the actual Schnorr proof witnesses), we compute: $C := R_0^{m_0} \cdots R_\ell^{m_\ell} S^v \bmod N$,

Table 2. Efficiency of proofs of predicates in multi-base exponentiations (MultiExps) dependent on the number of vertices n and of edges m. For a simple graph holds $m \leq \frac{n(n-1)}{2}$.

Predicate	**Basis**	**Commitments**	**MultiExps**	
		#	#	O
Possession		$n+m$	$2n+2m+1$	$O(n+m)$
Vertex Composition	Possession	n	$3n$	$O(n)$
Edge Composition	Possession	$2m$	$4m$	$O(m)$
Total Well-formed Graph		$2n+3m$	$n^2+8n+8m+1$	$O(n^2)$
Graph-3 Colorability (§5)		$n+m$	$6n+4m+1$	$O(n+m)$

The simulation uses random graphs with specified number of vertices n and a derived number of edges $m := 2n$ as major independent variable (on the x-axis), the dependent variable is computation time in milliseconds (in log-scale on the y-axis).

7 Related Work

Establishing zero-knowledge proofs on graphs and their properties is a classic area of research. Such proofs were instrumental in showing that there exist zero-knowledge proof systems for all NP languages. We discuss their graph modeling: Goldreich, Micali and Wigderson [21] offered such a construction with $O(m^2)$ rounds and $O(n)$ messages each. Based on the existence of a non-uniformly secure encryption function, they explored graph isomorphism and non-isomorphism as well as graph 3-colorability (G3C). Blum's proof [3] shows directed Hamiltonian cycles (DHC) in graphs. Both proofs use a metaphor of locked boxes to formulate the proof. Goldreich et al.'s G3C proof encodes the colors of adjacent vertices in boxes. Blum's proof of Hamiltonian cycles encodes the graph's adjacency matrix randomly in $n + \binom{n}{2}$ such boxes, giving the verifier the choice to either verify the correct graph representation or the knowledge of the Hamiltonian cycle. Blum offers an alternative construction for G3C with a similar methodology, encoding the graph representation and the coloring of each vertex in separate yet related boxes and operating on an adjacency matrix lifted to the labeling. Goldreich and Kahan [20] offered a constant-round construction based on the existence of collections of claw-free functions, also using G3C as NP-problem. We observe that these constructions are specific to the statement to be proven and do not cater for a level of indirection through a signature scheme.

A related notion to full graph signatures is transitive signature schemes, e.g., as proposed by Micali and Rivest [23]. They are concerned with the transitive closure of signatures on graph elements, where vertices and edges are signed individually; however, they do not offer zero-knowledge proofs of knowledge on graph properties.

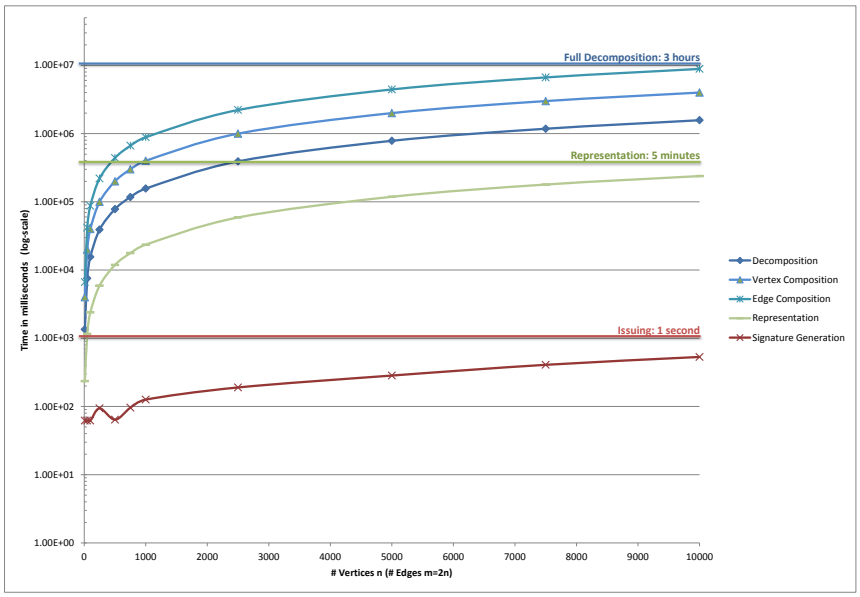

Fig. 2. Experimental performance analysis with a secure modulus length of 2048 bits, in the worst case of a non-parallelized computation on a single processor core (adapted from [2]). x-axis contains the number of vertices n and the y-axis a log-scale of computation time in milliseconds. Blue colors denote provider computations to prove properties of a committed graph, where the green line shows a proof of representation of a graph signature. Red colors denote auditing system/issuer computations to sign the graph.

8 Conclusion

We have introduced a practical construction of signatures on committed graphs and zero-knowledge proofs over their structure. The scheme is special in that it enables proofs over the entire graph structure, including statements such as isolation (two vertices are not connected by any sequence of edges). The construction derives its security from the properties of the Camenisch-Lysyanskaya (CL) signature scheme under the Strong RSA assumption. The interactive proofs are honest-verifier zero-knowledge if executed with multiple rounds with small challenges. While we have established a framework for graph topology proofs separately [2], this work focuses on the foundations of graph encoding in CL-signatures itself. We show its theoretical expressiveness by proving that the scheme is capable of signing committed NP statements and proving properties thereof, via reduction to graph 3-colorability. The presented scheme is efficient and practical because once the issuer has established graph well-formedness in $O(n^2)$, the prover can resort to proofs over the graph structure in linear time. The used Σ-proofs can be handled efficiently with batch processing techniques [25]. As future work, we aim at establishing a differential graph signature scheme, which can be employed for large-scale graph topologies as found in virtualized infrastructures.

References

1. ABE, M., FUCHSBAUER, G., GROTH, J., HARALAMBIEV, K., AND OHKUBO, M. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 209–236.
2. ANONYMIZED FOR REVIEW. Anonymized for review. In *conference proceedings to appear* (Nov. 2014).
3. BLUM, M. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians* (1986), vol. 1, p. 2.
4. BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology — EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 431–444.
5. BRANDS, S. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology — EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 318–333.
6. BRICKELL, E., CAMENISCH, J., AND CHEN, L. Direct anonymous attestation. In *Proc. 11th ACM Conference on Computer and Communications Security* (2004), acm press, pp. 225–234.
7. CAMENISCH, J., CHAABOUNI, R., AND SHELAT, A. Efficient protocols for set membership and range proofs. In *Advances in Cryptology-ASIACRYPT 2008* (2008), Springer, pp. 234–252.
8. CAMENISCH, J., AND GROSS, T. Efficient attributes for anonymous credentials. *ACM Transactions on Information and System Security (TISSEC) 15*, 1 (2012), 4.
9. CAMENISCH, J., AND LYSYANSKAYA, A. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT 2001* (2001), B. Pfitzmann, Ed., vol. 2045 of *LNCS*, Springer Verlag, pp. 93–118.
10. CAMENISCH, J., AND LYSYANSKAYA, A. A signature scheme with efficient protocols. In *Security in Communication Networks SCN 2002* (2003), vol. 2576 of *LNCS*, Springer Verlag, pp. 268–289.
11. CAMENISCH, J., AND MICHELS, M. Proving in zero-knowledge that a number n is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 107–122.
12. CAMENISCH, J., AND STADLER, M. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1296 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 410–424.
13. CHAN, A., FRANKEL, Y., AND TSIOUNIS, Y. Easy come – easy go divisible cash. In *Advances in Cryptology — EUROCRYPT '98* (1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 561–575.
14. COOK, S. A. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing* (1971), ACM, pp. 151–158.
15. CRAMER, R., DAMGÅRD, I., AND SCHOENMAKERS, B. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94* (1994), Y. G. Desmedt, Ed., vol. 839 of *LNCS*, Springer Verlag, pp. 174–187.
16. DAMGÅRD, I., AND FUJISAKI, E. An integer commitment scheme based on groups with hidden order. `http://eprint.iacr.org/2001`, 2001.
17. FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 186–194.

18. FUJISAKI, E., AND OKAMOTO, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 16–30.
19. GAREY, M. R., JOHNSON, D. S., AND STOCKMEYER, L. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing* (1974), ACM, pp. 47–63.
20. GOLDREICH, O., AND KAHAN, A. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology 9*, 3 (1996), 167–190.
21. GOLDREICH, O., MICALI, S., AND WIGDERSON, A. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM) 38*, 3 (1991), 690–728.
22. IBM. Specification of the Identity Mixer cryptographic library, v. 2.3.40. Specification, IBM Research, Jan. 2013. `http://prime.inf.tu-dresden.de/idemix/`.
23. MICALI, S., AND RIVEST, R. L. Transitive signature schemes. In *Topics in Cryptology-CT–RSA 2002*. Springer, 2002, pp. 236–243.
24. PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91* (1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 129–140.
25. PENG, K., BOYD, C., AND DAWSON, E. Batch zero-knowledge proof and verification and its applications. *ACM Transactions on Information and System Security (TISSEC) 10*, 2 (2007), 6.
26. RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (Feb. 1978), 120–126.
27. SCHNORR, C. P. Efficient signature generation for smart cards. *Journal of Cryptology 4*, 3 (1991), 239–252.

A Proofs

A.1 Well-formed Encoding and Security

Proof (Unambiguous encoding and decoding: Theorem 2). We show that there is a bijection between encoding and graph.
Graph $\rightarrow$ encoding: For each graph there exits a unique encoding modulo base association. For all vertices $i \in \mathcal{V}$ choose the vertex identifier $e_i \in \Xi_{\mathcal{V}}$, for the labels $k \in f_{\mathcal{V}}(i)$ choose the prime representative $e_k \in \Xi_{\mathcal{L}}$ and compute their product. As said factors are prime, it follows from the fundamental theorem of arithmetic that the $e_i \Pi_{k \in f_{\mathcal{V}}(i)} e_k$ represents a unique integer. Given that the user is not privy to the discrete logarithm between one base and another (guaranteed by the CL-Signature setup), the bases unambiguously separate the exponents. Thus, apart from the random permutation of the base association, the encoding is unambiguous.
Encoding $\rightarrow$ graph: With knowledge of the elements of $\Xi_{\mathcal{V}}$ and $\Xi_{\mathcal{L}}$, an encoded product can be decoded efficiently and unambiguously into the elements of the graph. That the parties are not privy to the discrete logarithm between base and another guarantees attribute separation. The base designates unambiguously whether a vertex or an edge is encoded. Given that all representatives of the encoding are prime, the product can be decomposed into a unique factorization by the fundamental theorem of arithmetic. Each representative unambiguously represents either a vertex identifier in $\Xi_{\mathcal{V}}$ or a label in $\Xi_{\mathcal{L}}$, as both sets are disjoint. □

Proof (Security of graph signatures: Theorem 4). The security of the scheme is directly derived from the unambiguous embedding into Integer commitments and Camenisch-Lysyanskaya signatures and their security properties. Theorem 2 establishes that the graph encoding encodes

graphs unambiguously into the CL-message space. The graph structure is encoded in the exponents of the Integer commitment and CL-signature schemes. Confidentiality is derived from the information-theoretical hiding property of the Integer commitment scheme and the hiding properties of CL-signatures on committed messages. Under the condition that the adversary is not privy to the group-order of the commitment and the CL signature scheme, we obtain that integrity for both schemes holds over the integers and thereby the graph encoding (cf. [16]). We obtain existential unforgeability against chosen message attacks directly from the CL-signature scheme in Theorem 1 [10].

B Well-formedness Proof

The following proof is representative for the argument structure of the proofs for different predicates; others use the same tools.

Proof (Wellformedness proofs, Theorem 3).
The Schnorr proofs used in the construction are honest-verifier zero-knowledge if executed repeatedly with small challenges, otherwise witness-indistinguishable. It is standard to extract from a successful prover knowledge on the secrets ranging over $\forall i, j$:

$$\mu_i, \mu_{(i,j)}, \rho, \rho_i, \rho_{(i,j)}, \varepsilon_i, \breve{\rho}_i, \gamma_i, \rho_i', \dot{\varepsilon}_i, \gamma_{(i,j)}, \rho_{(i,j)}', \alpha_{i,j}, \beta_{i,j}, \rho_{i,j}$$

such that all equations of the CS-notation hold for some t, where t must be ± 1 as modulus N is a product of two safe primes [16]. As CL-signatures are existentially unforgeable [10], we obtain that the messages μ_i and $\mu_{(i,j)}$ are indeed signed, and that the membership proofs for ε_i establish that $\varepsilon_i \in \Xi_{\mathcal{V}}$, i.e., are certified vertex identifiers (the CL multi-show unlinkability ensures that the verifier learns no other information about ε_i). The CG-OR proofs [8] yield that γ_i and $\gamma_{(i,j)}$ must encode valid vertex label identifiers (but yield no further information on the labels). Therefore, we have fixes the roots $\mu_i, \mu_{(i,j)}$ and the leaves $\varepsilon_i, \gamma_i, \gamma_{(i,j)}$ of the proof tree in the CL-notation.

It remains to show what can be derived from the equations that connect the roots to the leaves in the vertex and edge composition statements and from the pairwise difference. The technique used is a standard decomposition of certified messages in Integer commitments to make their components accessible to discrete-logarithm based proofs of knowledge; if the same secret is referenced we have an equality proof, if not there is no further information learned about the relation of the secrets. For the vertices, the equation $C_i \equiv \pm \breve{C}_i^{\gamma_i} S^{\rho_i'}$ (4) establishes that $\mu_i = \varepsilon_i \gamma_i$, given that the prover does not know a multiple of the group order, $\breve{C}_i$ separates out ε_i connected to the membership proof. For edges, the equation $C_{(i,j)} \equiv \pm \breve{C}_{(i,j)}^{\gamma_{(i,j)}} S^{\rho_{(i,j)}'}$ (7) establishes that $\mu_{(i,j)} = \mu_{(i,j)}' \gamma_{(i,j)}$, where $\breve{C}_{(i,j)}$ is shown to contain a product $\dot{\varepsilon}_i \dot{\varepsilon}_j$ in equation (3), which are in turn shown to be valid vertex identifiers (8). By that all variables are bound and the connection between the roots and the leaves established.

Finally, we claim pair-wise difference on vertices from the equation

$$R \equiv \pm \breve{C}_i^{\alpha_{i,j}} \breve{C}_j^{\beta_{i,j}} S^{\rho_{i,j}} \qquad (9).$$

Unless the prover knows a multiple of the group order or the discrete logarithm $\log_R S$, the following equation must hold over the integers:

$$1 = \varepsilon_i \alpha_{i,j} + \varepsilon_j \beta_{i,j}.$$

It is well-known that $\alpha_{i,j}$ and $\beta_{i,j}$ only exist if ε_i and ε_j are coprime, which gives us the pair-wise difference claimed.

B.1 Graph 3-Colorability (G3C)

Proof (Graph 3-Colorability: Lemma 1). **1. Proof of Knowledge.** It is standard to show that there exists a knowledge extractor for all exponents of the proof such that the equality of exponents equations are fulfilled.

We obtain from Clause 1 that the prover knows the representation of a CL-Signature of the given structure. From the existential unforgeability of CL-Signatures, we see that the issuer must have signed the secret attributes μ_i, μ_j and $\mu_{(i,j)}$. Proving equality of exponents with corresponding integer commitments is standard, by which the arguments over the commitments, such as C_i, $\breve{C}_i$ and $C_{(i,j)}$ transfer to the structure of the signed messages.

The Clause 4 shows that a message μ_i consists of two factors known to the prover: $\mu_i = \varepsilon_i\gamma_i$. The following Clause 5 employs a set membership proof to show that $\varepsilon_i \in \Xi_{\mathcal{V}}$ and that $\gamma_i \in \Xi_{\mathcal{L}}$. We use that the set membership from §2.5 guarantees that ε_i and γ_i are exactly one member of the set to conclude that a message μ_i contains exactly one vertex identifier and one label identifier. Thus, μ_i is well-formed. Similarly, Clause 6 establishes the structure $\mu_{(i,j)} = \varepsilon_i\varepsilon$ for the edge (i, j), showing it to be well-formed. Because the prover is not privy to the group order, these statements hold over the integers, by the results of Damgård and Fujisaki [16]. Therefore, with the proof of representation including pair-wise difference, we conclude that the signed graph is well-formed.

Clause 7 shows that the labeling $f_{\mathcal{V}}$ of the signed graph is a proper coloring. Again, we employ Damgård and Fujisaki's [16] result that equations hold over the integers. We have that for each edge (i, j), the corresponding signed messages have the following structure:

$$\mu_i = \varepsilon_i\gamma_i \quad \text{and} \quad \mu_j = \varepsilon_j\gamma_j.$$

We show that the secret labels γ_i and γ_j are different by showing that μ_i and μ_j are coprime, where we use Bézout's Identity:

$$\mathsf{gcd}(\mu_i, \mu_j) = 1 \quad \Leftrightarrow \quad 1 = \alpha_{(i,j)}\mu_i + \beta_{(i,j)}\mu_j.$$

The equality of exponent proof of Clause 7 achieves this as follows

$$\begin{aligned}
R &\equiv \pm C_i^{\alpha_{(i,j)}} C_j^{\beta_{(i,j)}} S^{\rho_{(i,j)}} \pmod N \\
R^1 &\equiv \pm (R_i^{\mu} S^{\rho_i})^{\alpha_{(i,j)}} (R_j^{\mu} S^{\rho_j})^{\beta_{(i,j)}} S^{\rho_{(i,j)}} \pmod N \\
R^1 &\equiv \pm R^{\alpha_{(i,j)}\mu_i} S^{\alpha_{(i,j)}\rho_i} R^{\beta_{(i,j)}\mu_j} S^{\beta_{(i,j)}\rho_j} S^{\rho_{(i,j)}} \pmod N \\
R^1 &\equiv \pm R^{\alpha_{(i,j)}\mu_i + \beta_{(i,j)}\mu_j} S^{\alpha_{(i,j)}\rho_i + \beta_{(i,j)}\rho_j + \rho_{(i,j)}} \pmod N
\end{aligned}$$

From this equation we can conclude that $\mathsf{gcd}(\mu_i, \mu_j) = 1$ and that, therefore, $\gamma_i \neq \gamma_j$, which implies that $f_{\mathcal{V}}(i) \neq f_{\mathcal{V}}(j)$ and that the CL signature indeed contains a proper coloring. □

2. Zero-Knowledge. We claim that proof does not disclose anything else than the statement made that the prover knows a CL-Signature of a proper coloring on known graph $\mathcal{G}$.

The Σ-proofs here are zero-knowledge in an honest verifier setting if performed with multiple rounds and small challenges. It is standard to construct a simulator for all Σ-proofs of representation for the CL-Signature and the commitments as well as for their conjunction [12,15], showing that the verifier does not learn anything else than the relations on exponents shown.

It remains to be shown what the relations disclose. We will argue on the statements made on the secret messages γ_i, which contain the color. Clause 4 establishes that γ_i is part of commitment C_i, but does not disclose further information than the equality of exponents.

Clause 5 proves that γ_i is a member of the set $\Xi_{\mathcal{L}} = \{e_{\mathsf{R}}, e_{\mathsf{G}}, e_{\mathsf{B}}\}$. This statement itself is part of the known problem definition of G3C. The set membership proof is a proof of representation

for an anonymized CL-Signature and a standard proof of equality of exponents, and thereby, does not disclose further information.

Finally, Clause 7 references $\mu_i = \epsilon_i \gamma_i$ to prove that γ_i and γ_j of an adjacent edge are coprime. As the vertex identifiers are pair-wise different by definition and as all representatives are primes, this only establishes that $\gamma_i \neq \gamma_j$ as required by the G3C problem, but nothing else. □

Proof (Polynomial Proof of G3C: Lemma 2). **Precomputation:** The prover computes $2n + 1$ signature randomizations with one exponentiation each and $2n + m$ integer commitments with 2 exponentiations each. The pre-computation phase uses $6n + 2m + 1$ exponentiations, transmits $4n + m + 1$ group elements, and thereby has a computation complexity of $O(n + m)$ and a communication complexity of $O(n + m)$.

Proof of Knowledge: The Schnorr proofs in the proof of knowledge are zero-knowledge if executed with small challenges over multiple rounds and can be connected with techniques from Cramer et al. [15]. The round complexity of the overall protocol is dependent on the proof mode (cf. Brands [5]).

Clause 1 is executed once yielding a Schnorr proof with $n + m + 2$ exponentiations for the prover.

The clauses 2 are executed once for each vertex, such as i and j, Therefore we have n Schnorr proofs with 2 exponentiations each for the prover.

The clauses 3 are executed once for each edge (i, j), making m Schnorr proofs with 2 exponentiations each for the prover.

The clauses 4 are executed once for each vertex, such as i or j. We have $2n$ Schnorr proofs with 2 exponentiations each for the prover.

The set membership proofs of Clauses 5 are executed once for each vertex and its label. Each set membership proof is a proof of representation of a designated CL-Signature for the set member, amounting to 3 exponentiations for the prover. In total, we have $2n$ such proofs of possessions, all done with a single Schnorr proof proving equality of exponents with the corresponding commitment.

Clause 6 proves the edge structure and is executed once per edge, yielding m Schnorr proofs with 2 exponentiations each for the prover. Finally, the proper graph coloring in Clause 7 is shows once for each edge (i, j) amounting to m Schnorr proofs with 3 exponentiations for the prover.

The proof of knowledge of graph coloring thereby requires $5n + 3m + 1 = O(n + m)$ Schnorr proofs with a computational complexity for the prover of $13n + 8m + 2 = O(n + m)$ exponentiations.

The total computational complexity is therefore $O(n + m)$, the communication complexity is $O(n + m)$ group elements. The G3C proof is done in polynomial time. The round complexity depends on the proof mode, where variants with multiple rounds (number of rounds depending on the error probability), with four rounds and initial commitments of the verifier on challenges, and three rounds in a Σ-proof (not zero-knowledge) are possible. □

B.2 CL Proof Systems for NP-Statements

Proof (Sketch NP-Statements: Theorem 5). Let a NP language $\mathfrak{L}$ be given. Let τ be a polynomial-time computable and invertible reduction from $\mathfrak{L}$ to Graph 3-Colorability (G3C): τ can be constructed by composing a polynomial-time reduction of $\mathfrak{L}$ to 3SAT by Cook's proof [14] and a polynomial-time reduction from 3SAT to G3C. We have that $x \in \mathfrak{L}$ iff $\tau(x)$ is 3-colorable.

On common input x, both prover and verifier compute graph $G \leftarrow \tau(x)$. In Goldreich, Micali and Widgerson's work, the proof proceeds to use any interactive zero-knowledge proof system to

prove that G is 3-colorable and thereby show that $x \in \mathfrak{L}$. Our proof continues from this point to show that there exists a Camenisch-Lysyanskaya proof system.

On obtaining $\mathcal{G} = \tau(x)$, the prover constructs a graph commitment C on $\mathcal{G}$ as defined in §3, including a labeling $f_{\mathcal{V}}$ of a proper coloring of $\mathcal{G}$. The known-graph proof transmits $\mathcal{G}$ itself, yet keeps the proper coloring confidential as default.

Proof of representation $\mathsf{P} \rightarrow \mathsf{I}$: The prover interacts with an CL-Signature issuer, proving representation and well-formedness of the commitment C in a known-graph proof, disclosing information to satisfy the verification requirements of the issuer. As $\tau(x)$ is invertible, this proof of representation of G and the proper coloring serves as proof of representation for x and $x \in \mathfrak{L}$.

Issuing $\mathsf{I} \rightarrow \mathsf{P}$: Upon acceptance of the proof, the issuer signs the committed graph $\mathcal{G}$ in a CL-Signature σ. Given the invertibility of τ, this signature holds for x as well. $sigma$ is a CL-Signature on $\tau(x)$ and the proper coloring of $\tau(x)$ iff $x \in \mathfrak{L}$.

Proof of possession $\mathsf{P} \rightarrow \mathsf{V}$: The prover interacts with the verifier to proof knowledge of the CL-Signature σ on a proper coloring on $\mathcal{G}$ and thereby shows graph 3-colorability of $\tau(x)$, which holds iff $x \in \mathfrak{L}$. Thereby, the proof of possession of σ translates to a proof of possession of the statement $x \in \mathfrak{L}$. The proof is zero-knowledge if executed with small challenges over multiple rounds. □

Efficient Certification and Zero-Knowledge Proofs of Knowledge on Infrastructure Topology Graphs

Thomas Groß
School of Computing Science, Newcastle University, UK
thomas.gross@newcastle.ac.uk

ABSTRACT

Digital signature schemes are a foundational cryptographic building block in certification and the projection of trust. Based on a signature scheme on committed graphs, we propose a framework of certification and proof methods to sign topology graphs and to prove properties of their certificates in zero-knowledge. This framework allows an issuer, such as an auditing system, to sign the topology representation of an infrastructure. The prover, such as an infrastructure provider, can then convince a verifier of topology properties including connectivity and isolation without disclosing the blueprint of the topology itself. By that, we can certify the structure of critical systems while still maintaining confidentiality. We offer zero-knowledge proofs of knowledge for a general specification language of security goals for virtualized infrastructures such that high-level security goals can be proven over topology certificates. We offer an efficient and practical construction, built upon the Camenisch-Lysyanskaya signature scheme [11], honest-verifier proofs and the strong RSA assumption.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption—*Public key cryptosystems*

General Terms

Algorithms, Design, Security

Keywords

Cloud; topology; digital signatures; graph signature scheme

1. INTRODUCTION

Digital signature schemes are foundational cryptographic primitives, in particular to ensure the primary security property of integrity. From their conception [33], digital signature schemes have been employed to sign messages or committed messages [11]. Nowadays, they establish the integrity of systems and their components via certification of software or attestation of software stacks. Their use in attestation of a system, as pursued with Direct Anonymous Attestation (DAA) [8], is particularly relevant when a tenant delegates computation, networking or storage to a provider, such as in outsourcing or cloud computing. In this work, we focus on the question how digital signatures can ensure structural integrity of an infrastructure while maintaining its confidentiality.

The tenants will question the integrity of the systems in which their resources are hosted, in particular, as misconfigurations and insider attackers are considered by ENISA [19] and CSA [17] as high risks exposing the tenants to, e.g., isolation failure. Consequently, cloud security assurance sought to establish structural properties of virtual infrastructures for isolation and deployment patterns [4, 3] as well as for hidden dependency graphs [38, 37]. Our own work with an infrastructure cloud and auditing system provider similarly indicates that structural security properties and a projection of trust from an auditing system to a provider are deemed important.

The systems are typically large topologies with flat hierarchies, by which structural properties, inter-connectivity and isolation are important for the security of the tenants' sub-systems and the system at large. The tenants (and collaborating providers in clouds-of-clouds) will naturally expect the provider to convince them that the system is well-structured and that their own resources are properly isolated from other tenants. However, this fundamental integrity requirement of the tenants is at odds with the confidentiality requirement of the provider. The infrastructure provider naturally requires the blue-print of the infrastructure to be secret. The provider also aims to protect the other tenants from exposure and to ensure that the tenants own confidentiality requirements on their sub-systems are fulfilled. Therefore, we ask: *How can a provider convince a verifier that the topology fulfills security properties, such as zone isolation, without disclosing the blueprint of the topology?*

We believe that a signature scheme on committed graphs and efficient zero-knowledge proofs of knowledge thereon offers a building block to solve this problem. It provides us with proofs of knowledge that show elaborate statements on topology security properties, while keeping the topology itself confidential. Our work complements existing results in the tenant-verifiable integrity of infrastructures with host-based monitoring [34] and the attestation of physical hosts and virtual machines [8]: Graph signatures offer the confidential attestation of the system structure and enable a provider to convince a verifier that the system is structured securely, while keeping the blueprint of the system secret.

Contributions: 1) We specify the first framework for efficient zero-knowledge proofs of knowledge over signatures on topology graphs, directly applicable to security goals raised by tenants in the domain of virtualized infrastructures expressed in high-level language. **2)** The graph signatures and proof systems are a generic solution for arbitrary vertex-/edge-labeled undirected graphs in various problem domains, such as dependency and hierarchy graphs,

CCSW'14, November 7, 2014, Scottsdale, Arizona, USA.
ACM 978-1-4503-3239-2/14/11.
http://dx.doi.org/10.1145/2664168.2664175

access control policy and provenance graphs or attack trees and structured occurrence nets of incidents. Thus, the scheme has further applications in the cloud, beyond the infrastructure topology itself. **3)** We establish general efficient zero-knowledge proofs of knowledge for standard statements over topologies, which allow us to make statements over vertex and edge sets, connectivity and isolation. Combined with known discrete-logarithm based proofs of knowledge [35, 18, 21, 15, 12, 6, 14], we obtain an expressive framework. We believe that the presented topology graph signatures are a suitable building block to close the gap between existing attestation of individual components of the infrastructure and statements on the security of the entire infrastructure. **4)** In particular, we propose an efficient general method to prove isolation in linear complexity, while keeping the topology itself confidential. **5)** Our system comes with systematic asymptotical and experimental performance evaluation on a prototypical implementation. We are working with an infrastructure cloud and auditing system provider to ensure that the approach is practicable.

The graph signature scheme allows us to bridge the gap between the integrity requirements of the tenant and the confidentiality requirements of the provider in virtualized infrastructures.

2. SCENARIO AND KEY IDEA

The key idea of this work is to enable a trusted third-party *auditing system* to certify a virtualized infrastructure, such that an *infrastructure provider* can prove to *tenants* (or other infrastructure providers) that security properties are fulfilled, without disclosing the confidential blueprint of the virtualized infrastructure. Whereas we focus this paper on the tenants, yet see applications for cross-provider proofs as well.

Let us consider the scenario in Figure 1 vis-à-vis the nature of the virtualized infrastructure as a large-scale distributed system, which is dynamically changing through self-provisioning, elastic scalability and provider-issued migrations for optimization of utilization. The sheer scale of an infrastructure will require certification to be partitioned while the dynamics of the system will require that the auditing system is continuously present.

Auditing System.

We may imagine the auditing system as a *flight recorder*, which keeps track of the infrastructure state as it observes it. The auditing system analyzes the virtualized infrastructure through the management host interfaces with its own eyes: It employs a topological security analysis tool which comes with its own discovery probes to establish the state of the virtualized infrastructure.

We stress that the audit system can consider arbitrary topology properties such as the dependency graphs investigated by the Structural Reliability Auditor (SRA) [38]. For information flow properties considered here, we find examples of suitable tools in the research domain, e.g., Bleikertz et al. [4, 3], or in industry products, e.g., IBM PowerSC Trusted Surveyor, where both kinds yield a realization model (i.e., a graph representation of the infrastructure's topology). The realization model contains high- and low-level components of compute, network and storage resource as well as their connections as described in the management host's configuration. Research in this space had a considerable focus on ensuring that the realization model is a faithful representation of the actual infrastructure. For instance, such a model would contain PortGroups as vertices with associated VLAN identifiers as labels. The auditing system annotates the graph further, for instance with its own geolocation.

From this graph representation of the infrastructure, the auditor issues a graph signature σ to the provider: The auditing system certifies the topology and configuration as seen by the discovery probes of the security tool. Given the dynamically changing nature of the virtualized infrastructure, the auditing system will sign a continuous sequence of graphs bound to a time index in short succession. Therefore, we will have a requirement that the issuing can be done rapidly.

Infrastructure Provider.

Given such a graph signature, the provider can prove to tenants in zero-knowledge that security properties specified in a high-level language, such as *VALID* [2] or a graph rewriting language, such as *GROOVE*, are fulfilled. Observe that the graph signatures are generic and independent from tenants and the security properties they ask for. In practice, this is driven by tenant requests, by which the infrastructure provider stores a batch of signatures for a retention period.

Tenant.

A tenant auditing his own virtualized infrastructure fixes a time index t and asks the provider questions about the topology of the infrastructure at that time, after the fact. Examples include the following questions:

- "Are my resources isolated from any competitor?" (Zone isolation)
- "Are all my resources geolocated in Europe?" (Deployment correctness)
- "Are all my resources reachable by two independent paths?" (Availability)

Upon receiving such a question, the provider can then look up the graph signature for time index t, and compute a zero-knowledge proof of knowledge that the required property is fulfilled on the signature without disclosing further information about the infrastructure: "Yes, I hold a graph signature for time index t issued by a trusted auditing system, for which the property holds that your resources are isolated from any competitor." The tenant verifies the zero-knowledge proof with respect to the public key of the auditing system and can, thereby, be convinced that the property was fulfilled in the state the auditing system has observed.

Building Blocks and Outline.

To establish the topology certification, we need multiple building blocks spanning a range of conceptual levels. *First*, we need a graph signature scheme that can be employed for a certification by an auditing system. The key idea here is to establish an anonymous credential system that operates on graphs. It must be capable of binding different elements of graphs together (e.g. vertices and their labels) and of making them accessible to zero-knowledge proofs. The preliminaries in §4 are setting the stage: Camenisch-Lysyanskaya signatures and Camenisch-Groß encoding. Having established the preliminaries, §5 introduces the cryptographic interfaces for the graph signature scheme algorithms and the library of proof predicates we construct subsequently.

Second, we need to establish how the signature scheme can be implemented, in particular, how the foundational primitive of a *proof of representation* works. Thus, the next sections focus on the implementation of the graph signature interfaces, where §5.2 establishes the underlying encoding for undirected. §5.3.1 offers the core building blocks of proofs of representations as well as key generation.

Third, we will investigate how to compile zero-knowledge proofs for specific statements, by which §6 offers constructions for a li-

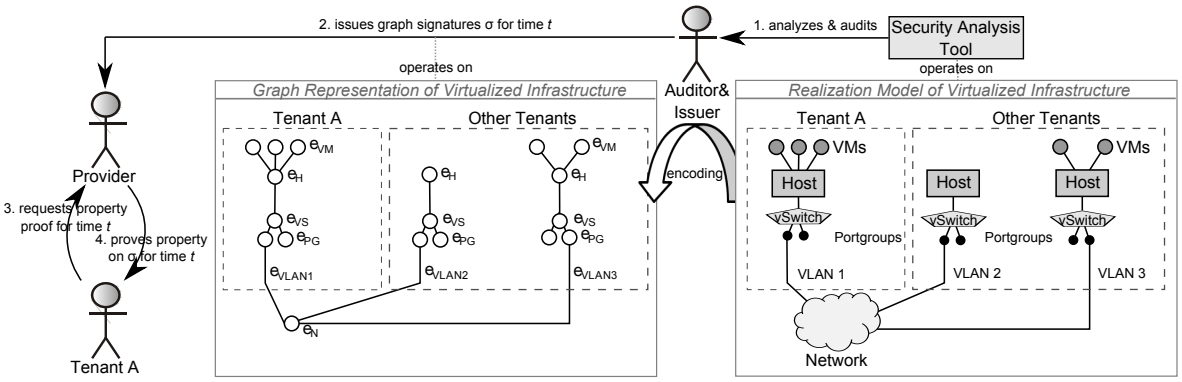

Figure 1: System Model of an auditing system operating on a realization model of a virtualized infrastructure. The auditing system continuously inspects the infrastructure with a security analysis tool at its disposal and certifies graph signatures to the infrastructure provider. The provider, in turn, proves properties of the infrastructure to tenants upon request.

brary of graph proofs, including statements over vertex sets, connectivity and isolation.

Fourth, we need to convince ourselves that a graph signature and proof system can be implemented sufficiently efficiently and scalably, which we investigate in §7 asymptotically as well as experimentally. §8 compares this work with earlier proposals for transitive and homomorphic graph signatures and zero-knowledge proofs on graphs, while §9 discusses future work.

3. SYSTEM MODEL

As system model (Figure 1) we consider a virtualized infrastructure, with compute, network and storage resources, operated via management hosts. A typical example of such an infrastructure is VMware administered with vCenter. We consider three parties: an *infrastructure provider* operating the virtualized infrastructure (acting as recipient of graph signatures and as prover towards tenants), a trusted *auditing system* with read-only access to the management hosts (acting as issuer of graph signatures) and multiple *tenants* (acting as verifiers of zero-knowledge proofs of knowledge on graph signatures).

Trust is asymmetric: The provider trusts the auditor to keep the infrastructure configuration confidential; the tenants trust the auditor to sign only well-formed graphs and only as faithfully obtained from the realization model of the security tool. The faithful representation of virtualized infrastructure configurations (and their changes) in graph models has been studied in the recent years starting from Bleikertz et al. [4] and is assumed as given in this work.

Fault Model: We assume hypervisors and management hosts to be correct: Software failures of hypervisors are out of scope; security failures, in particular, both spawning from misconfigurations (non-malicious human faults) and insider attackers (malicious human faults) are in the scope of this work. We stress that even if the virtualized infrastructure uses secure hypervisors and is protected by TPMs, malicious or non-malicious misconfigurations are a major problem [19, 17] and have been observed in our own work with industrial partners.

4. PRELIMINARIES

4.1 Assumptions

Special RSA Modulus. A *special RSA modulus* has the form $N = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, the corresponding group is called *special RSA group*. *Strong RSA Assumption [33, 21].* Given an RSA modulus N and a random element $g \in \mathbb{Z}_N^*$, it is hard to compute $h \in \mathbb{Z}_N^*$ and integer $e > 1$ such that $h^e \equiv g \bmod N$. The modulus N is of a special form pq, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. *Quadratic Residues.* The set QR_N is the set of Quadratic Residues of a special RSA group with modulus N (cf. [36]).

4.2 Integer Commitments

Damgård and Fujisaki [18] showed for the Pedersen commitment scheme [31] that if it operates in a special RSA group and the committer is not privy to the factorization of the modulus, then the commitment scheme can be used to commit to *integers* of arbitrary size. The commitment scheme is information-theoretically hiding and computationally binding. The security parameter is ℓ. The public parameters are a group G with special RSA modulus N, and generators $(g_0, \ldots, g_m)$. In order to commit to the values $(V_1, \ldots, V_l) \in (\mathbb{Z}_n^*)^l$, pick a random $R \in \{0,1\}^\ell$ and set

$$C = \mathsf{Commit}(R, V_1, \ldots, V_l) = g_0^R \prod_{i=1}^{l} g_i^{V_i}.$$

4.3 Known Discrete-Logarithm-Based Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [35] or a composite [18, 21], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [15] or composite [12] moduli, (3) proof that a commitment opens to the product of two other committed values [7, 12], (4) proof that a committed value lies in a given integer interval [6, 12, 14], and also (5) proof of the disjunction or conjunction of any two of the previous ones [16]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption.

Proofs as described above can be expressed in the notation introduced by Camenisch and Stadler [13]. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta \wedge (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta$ holds, where $u \leq \alpha \leq v$,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities of which knowledge is being proven, while

all other values are known to the verifier. We apply the Fiat-Shamir heuristic [20] to turn such proofs of knowledge into signatures on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^{\alpha}\}(m)$. Given a protocol in this notation, it is straightforward to derive an actual protocol implementing the proof.

We introduce the following short-hands: i. A modulus statement $(\text{mod } N)$ in the PK-header denotes the default modulus for subsequent non-range proof congruences. ii. An all quantifier $\forall i$ denotes that the secrets/terms in its scope are iterated over i. iii. We decompose proofs of knowledge statements in multiple steps and require referential integrity between the secrets of the steps.

4.4 Camenisch-Lysyanskaya Signatures

Let us introduce Camenisch-Lysyanskaya (CL) signatures in a Strong RSA setting [11].

Let $\ell_{\mathcal{M}}$, ℓ_e, ℓ_N, ℓ_r and L be system parameters; ℓ_r is a security parameter, $\ell_{\mathcal{M}}$ the message length, ℓ_e the length of the Strong RSA problem instance prime exponent, ℓ_N the size of the special RSA modulus, L the number of message bases. The scheme operates with a ℓ_N-bit special RSA modulus. Choose, uniformly at random, $R_0, \ldots, R_{L-1}, S, Z \in \text{QR}_N$. The public key pk_{I} is $(N, R_0, \ldots, R_{L-1}, S, Z)$, the private key sk_{I}, the factorization of the special RSA modulus.

The *message space* is $\{(m_0, \ldots, m_{L-1}) : m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}\}$.

Signing algorithm. On input $m_0, \ldots, m_{L-1}$, choose a random prime number e of length $\ell_e > \ell_{\mathcal{M}} + 2$, and a random number v of length $\ell_v = \ell_N + \ell_{\mathcal{M}} + \ell_r$. Compute

$$A = \left(\frac{Z}{R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v} \right)^{1/e} \mod N.$$

The signature consists of (e, A, v).

Verification algorithm. To verify that the tuple (e, A, v) is a signature on message $(m_0, \ldots, m_{L-1})$, check that the following statements hold:

$$Z \equiv A^e R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v \pmod N,$$

with $m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}$, and $2^{\ell_e} > e > 2^{\ell_e - 1}$.

THEOREM 4.1. *[11]*
The signature scheme is secure against adaptive chosen message attacks [23] under the strong RSA assumption.

Proving Knowledge of a Signature. A prover can prove that she possesses a CL-signature without revealing any other information about the signature (as well as use the primitives in §4.3). The prover randomizes A: Given a signature (A, e, v), the tuple $(A' := AS^{-r} \bmod N, e, v' := v + er)$ is a valid signature as well. Now, provided that $A \in \langle S \rangle$ and that r is chosen uniformly at random from $\{0,1\}^{\ell_N + \ell_\varnothing}$, the value A' is distributed statistically close to uniform over QR_N. Thus, the user could compute a fresh A' each time, reveal it, and then run the protocol

$$\begin{aligned} &PK\{(\varepsilon, \nu', \mu_0, \ldots, \mu_{L-1}) : \\ &\quad Z \equiv \pm R_0^{\mu_0} \cdots R_{L-1}^{\mu_{L-1}} A'^{\varepsilon} S^{\nu'} \pmod N \ \wedge \\ &\quad \mu_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}} \ \wedge \ \varepsilon \in [2^{\ell_e - 1} + 1, 2^{\ell_e} - 1]\} \end{aligned}$$

4.5 Set Membership from CL-Signatures

Set membership proofs can be constructed from CL-Signatures following a method proposed by Camenisch et al. [9]. For a set $\mathcal{S} = \{m_0, \ldots, m_i, \ldots, m_l\}$, the issuer signs all set members m_i in CL-Signatures $\sigma_i = (A, e, v)$ and publishes the set of message-signature pairs $\{(m_i, \sigma_i)\}$ with integrity. To prove set membership of a value committed in C, the prover shows knowledge of the blinded signature σ_i' corresponding to the message m_i and equality of exponents with C. We describe the implementation in the extended version [26] and denote a set membership proof $\mu[C] \in \mathcal{S}$, which reads μ encoded in commitment C is member of set $\mathcal{S}$.

4.6 Camenisch-Groß Encoding

The Camenisch-Groß (CG) Encoding [10] gives the CL message space structure by encoding multiple binary and finite-set values into a single message, and we will use a similar paradigm to encode graphs efficiently.

The core principle of the CG-Encoding is to represent binary and finite-set attribute values as prime numbers. It uses divisibility and coprimality to show whether an attribute value is present in or absent from a credential. The attribute values certified in a credential, say e_i, e_j, and e_l, are represented in a single message of the CL-Signature, by signing the product of their prime representative $E = e_i \cdot e_j \cdot e_l$ in an Integer attribute. The association between the value and the prime number of the encoding is certified by the credential issuer.

Divisibility/AND-Proof. To prove that a disclosed prime representative e_i is present in E, we prove that e_i divides the committed product E, we show that we know a secret μ' that completes the product:

$$PK\{(\mu', \rho) : \quad D \equiv \pm(g^{e_i})^{\mu'} h^{\rho} \pmod N\}.$$

Coprimality/NOT-Proof. We show that one or multiple prime representatives are not present in a credential, we show coprimality. To prove that two values E and F are coprime, i.e., $\gcd(E, F) = 1$, we prove that there exist integers a and b such that Bézout's Identity equals 1, where a and b for this equation do not exist, if $\gcd(E, F) > 1$.

$$\begin{aligned} PK\{(\mu, \rho, \alpha, \beta, \rho') : \quad & D \equiv \pm g^{\mu} h^{\rho} \pmod N \ \wedge \\ & g \equiv \pm D^{\alpha} (g^F)^{\beta} h^{\rho'} \pmod N\}. \end{aligned}$$

OR-Proof To show that a credential contains an attribute e that is contained in an OR-list, we show there exists an integer a such that $ae = \prod_i^{\ell} e_i$; if e is not in the list, then no such integer a as e does not divide the product. We use the notation $\alpha \subseteq \Xi$ for an OR-proof that α contains one or more values of Ξ.

5. GRAPH SIGNATURE SCHEME

This section establishes the core graph signature scheme: The key point here is that we offer a graph encoding that lifts the anonymous credential scheme given by the Camenisch-Lysyanskaya signatures (§4) to entire graphs as messages. The encoding is special in that it keeps all elements of the graph (vertices, edges, labels) accessible to efficient zero-knowledge proofs.

5.1 Cryptographic System Interface

Let us first specify the abstract interface of a graph signature scheme and associated proofs over graph properties. The core signature scheme consists of five algorithms:

$$\mathsf{Commit}(), \mathsf{Keygen}(), \mathsf{Sign}(), \mathsf{HiddenSign}(), \text{ and } \mathsf{Verify}(),$$

where we consider $\mathsf{HiddenSign}()$ and $\mathsf{Verify}()$ in the extended version [26], that is, the joint signing of hidden committed graphs.

$\mathsf{Commit}(\mathcal{G}; R)$ is a probabilistic polynomial-time algorithm, providing an Integer commitment lifted to graphs. It takes as input a graph $\mathcal{G}$ encoded with the encoding $\mathsf{encode}(\mathcal{G})$ specified in §5.2 and randomness R.

$\mathsf{Keygen}(1^{\ell}, params)$ establishes the key setup for the graph signature scheme in a probabilistic polynomial-time algorithm. It

takes as input the security parameter ℓ and the public parameters of the commitment scheme Commit. It outputs a key pair (pk, sk), where pk is the public key of the issuer and sk is its secret key.

$\mathsf{Sign}(pk_{\mathsf{I}})$ is a probabilistic polynomial-time algorithm run by issuer I. It signs a graph $\mathcal{G}_{\mathsf{I}}$. The public input is a commitment on the user sub-graph issuer's public key pk_{I}. The issuer's private input is the graph $\mathcal{G}_{\mathsf{I}}$ and his secret key sk_{I}. The output is a signature on the graph $\sigma = \sigma(\mathcal{G}_{\mathsf{I}})$.

In addition, we provide proof predicates for graph signatures (cf. with Table 1): First, we provide predicates for the graph proof of representation possession as well as decompositions to vertex and edge level, both implemented in §5.3.1. Second, we consider sets of graph elements with set coverage cover, pair-wise disjointness disjoint and partition partition. Third, we have predicates on connectivity edge and connected and its complement isolation isolated $= \neg$connected. We establish an implementation for zero-knowledge proofs of knowledge thereof in §6. These proofs can be combined with known proofs of knowledge based on discrete-logarithms and composed to Boolean formulas with logical connectives $\wedge$ and $\vee$.

REMARK 1 (CLOUD SECURITY ASSURANCE).
The predicates introduced in Table 1 correspond to the major predicates of VALID *[2], a formal specification language of cloud security goals suitable for automated model checking.* VALID *expresses goal states as a set (conjunction) of positive and negative facts constrained by a Boolean condition list. It uses terms, such as* edge$(\cdot,\cdot)$ *or* connected$(\cdot,\cdot)$ *to express alarm states on topology graphs.*

Subsequently, we will introduce the implementations for proofs of knowledge for these different functions successively. §5.2 introduces the encoding of undirected graphs itself. §5.3 introduces the key generation Keygen, the proof of representation graph and the issuing and verification algorithms HiddenSign and Verify. Subsequently, §6 contains the constructions for the set and connectivity predicates.

5.2 Graph Encoding

The key idea of the graph encoding is this: we represent vertex identifiers and labels as prime numbers. This allows us to bind different elements of the graph together by signing the product of their representations. In addition, we can have efficient proofs of properties such as connectedness and isolation, by arguing over divisibility and coprimality of the prime products. The proofs can be implemented efficiently with discrete-logarithm zero-knowledge proofs.

We consider strict undirected graphs over finite vertex sets and finite sets of vertex and edge labels, where vertices and edges can have multiple labels. We describe the encoding and proofs for directed graphs in the extended version of this paper [26].

$\mathcal{V}$	Finite set of vertices
$\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$	Finite set of edges
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, t_{\mathcal{V}}, t_{\mathcal{E}})$	Graph
$\mathcal{L}_{\mathcal{V}}, \mathcal{L}_{\mathcal{E}}$	Finite sets labels
$f_{\mathcal{V}} : \mathcal{V} \to \mathcal{P}(\mathcal{L}_{\mathcal{V}})$	labels of a given vertex
$f_{\mathcal{E}} : \mathcal{E} \to \mathcal{P}(\mathcal{L}_{\mathcal{E}})$	labels of a given edge
$n = \lvert\mathcal{V}\rvert, m = \lvert\mathcal{E}\rvert$	number of vertices and edges

We call a *prime representative*, a prime number which denotes an element of a graph. For each vertex i in $\mathcal{V}$, we introduce a *vertex identifier*, a prime e_i, which represents this vertex in credential and proofs. The symbol $\perp$, associated with identifier $e_\perp$ represents that a vertex is not present. All vertex identifiers are pair-wise different. We call the set of all vertex identifiers $\Xi_{\mathcal{V}}$, their product $\chi_{\mathcal{V}} = \Pi\Xi_{\mathcal{V}}$. For each label k in the label sets $\mathcal{L}_{\mathcal{V}}$ and in $\mathcal{L}_{\mathcal{E}}$, we introduce a prime representative e_k. All label representatives are pair-wise different. We call the set of all label representatives $\Xi_{\mathcal{L}}$, their product $\chi_{\mathcal{L}} = \Pi\Xi_{\mathcal{L}}$. Vertex identifiers and label representatives are disjoint:

$$\Xi_{\mathcal{V}} \cap \Xi_{\mathcal{L}} = \emptyset \quad \Leftrightarrow \quad \mathsf{gcd}(\chi_{\mathcal{V}}, \chi_{\mathcal{L}}) = 1.$$

5.2.1 Encoding Vertices and Edges

To encode a vertex and its associated labels into a graph commitment or CL-Signature, we encode the product of the vertex identifier $e_i \in \Xi_{\mathcal{V}}$ and the prime representatives $e_k \in \Xi_{\mathcal{L}}$ for $k \in f_{\mathcal{V}}(i)$ of the labels into a single message of the signature. The product of prime representatives is encoded as exponent of dedicated vertex bases $R \in G_{\mathcal{V}}$.

To get a compact encoding and efficient proofs thereon, the encoding needs to maintain the graph structure and to allow us to access it to prove higher-level properties, such as connectivity and isolation. The proposal we make in this paper after evaluating multiple approaches is to use divisibility and coprimality similar to the CG-Encoding to afford us these efficient operations over the graph structure, while offering a compact encoding of edges.

Recall that each vertex is certified with a vertex identifier from $\Xi_{\mathcal{V}}$, e.g., e_i or e_j. For each edge $(i,j) \in \mathcal{E}$, we include an edge attribute as exponent of a random edge base $R_{\pi(i,j)} \in G_{\mathcal{E}}$, containing the product of the vertex identifiers and the associated label representatives $e_k \in \Xi_{\mathcal{L}}$ for $k \in f_{\mathcal{E}}(i,j)$ of the edge:

$$E_{(i,j)} := e_i \cdot e_j \cdot \Pi_{k \in f_{\mathcal{E}}(i,j)} e_k.$$

DEFINITION 1 (WELL-FORMED GRAPH).
We call a graph encoding well-formed *iff 1. the encoding only contains prime representatives $e \in \Xi_{\mathcal{V}} \cup \Xi_{\mathcal{L}}$ in the exponents of designated vertex and edge bases $R \in G_{\mathcal{V}} \cup G_{\mathcal{E}}$, 2. each vertex base $R \in G_{\mathcal{V}}$ contains exactly one vertex identifier $e_i \in \Xi_{\mathcal{V}}$, pair-wise different from other vertex identifiers and zero or more label representatives $e_k \in \Xi_{\mathcal{L}}$, and 3. each edge base $R \in G_{\mathcal{E}}$ contains exactly two vertex identifiers $e_i, e_i \in \Xi_{\mathcal{V}}$ and zero or more label representatives $e_k \in \Xi_{\mathcal{L}}$.*

5.3 Signing Committed Graphs

Once we have the encoding, the next question is how graphs can be signed. Luckily, the encoding embeds the graph directly into the Camenisch-Lysyanskaya signature scheme, we can employ the signing method given in the preliminaries (§4) directly for the simple case that an audit system inspects a topology and signs the graph it observes.

The full graph signature scheme supports a joint issuing of graph signatures with a hidden committed subgraph contributed by a user and another known subgraph contributed by the issuer. We define the issuing process, including the considerations for hidden graphs being merged in the extended version.

As a point of reference, we give the structure of the graph signatures, as it guides how we prove representation of such signatures. We have seen in the encoding (§5.2) that it operates over bases $R_{\pi(i)} \in G_{\mathcal{V}}$, which store attributes encoding vertices in the exponent, and bases $R_{\pi(i,j)} \in G_{\mathcal{E}}$, which store attributes encoding edges in the exponent. The base association is randomized by

Table 1: Proof of knowledge predicates for graph signatures (cf. *VALID* [2]).

Predicate	Description	
$\mathsf{possession}(\mathcal{G}, \sigma, \mu_i)$	Proof of possession of a graph signature σ	§5.3.1
$\mathsf{vertices}(\mathcal{G}, \varepsilon_i, \gamma_i)$	Proof of composition of graph vertices of a proof of possession	§5.3.1
$\mathsf{edges}(\mathcal{G}, \varepsilon_i, \varepsilon_i, \gamma_{(i,j)})$	Proof of composition of graph edges of a proof of possession	§5.3.1
$\mathsf{graph}(\mathcal{G}, \mu_i)$	Proof of representation and well-formedness (extended version [26])	§5.3.1
$\mathsf{set}(\mathcal{V}, V)$	Representation of a set $V \subseteq \mathcal{V}$	§6.1
$\mathsf{cover}(\mathcal{V}, V_1, \ldots, V_k)$	Vertex set coverage $\bigcup(V_1, \ldots, V_k) = \mathcal{V}$	§6.1
$\mathsf{disjoint}(\mathcal{V}, V_1, \ldots, V_k)$	Vertex set pair-wise disjointness $\bigcap(V_1, \ldots, V_k) = \emptyset$	§6.1
$\mathsf{partition}(\mathcal{V}, V_1, \ldots, V_k)$	Vertex set partition $\bigcup(V_1, \ldots, V_k) = \mathcal{V} \wedge \bigcap(V_1, \ldots, V_k) = \emptyset$	§6.1
$\mathsf{edge}(\mathcal{G}, i, j)$	*VALID*-rule: Adjacency of (i, j)	§6.2
$\mathsf{connected}(\mathcal{G}, i, j, \ell)$	*VALID*-rule: Existence of an ℓ-path between vertex i and vertex j	§6.2.1
$\mathsf{isolated}(\mathcal{G}, i, j)$	*VALID*-rule: Isolation of vertices i and j: There exists no path between i and j	§6.2.2

permutations $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{E}}$.[1] The following congruence holds in $\mathbb{Z}_n^*$:

$$Z \equiv \pm \underbrace{R_{\pi(i)}^{e_i \Pi_{k \in f_{\mathcal{V}}(i) e_k}}}_{\forall \text{ vertices } i} \cdots \underbrace{R_{\pi(i,j)}^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j) e_k}}}_{\forall \text{ edges } (i,j)} A^e S^v$$

5.3.1 Proof of Representation

Establishing zero-knowledge proofs on (graph) signatures usually starts with a *proof of representation*: This proof shows that the prover knows all the secrets contained in the signature equation. It shows that the prover actually *possesses* the graph signature. To talk about further properties of the values in the signature, we need to make these values available in a commitment and thereby accessible to individual treatment. For instance, if a proof is to make statements about the vertices, we need to have commitments on the vertex identifiers and prove that the values in the commitments are equal to the values in the signature. We call this process *decomposition*; it creates a tree of commitments with the graph signature at its root and equality proofs connection the nodes. Whereas a proof of possession of a signature is standard, we expand on it to show how the graph signature is decomposed in commitments on its components. These decompositions implement the predicates possession, vertices, edges, and graph from Table 1 and constitute reusable building blocks for many proofs.

REMARK 2 (FULL GRAPH DECOMPOSITION). *It is crucial to note that the full graph decomposition including a proof of well-formedness and pair-wise difference* graph *(Def. 1) is* only *needed when a user contributes a hidden graph. For proofs considered in this paper, the parties can trust the auditing system that it will only sign well-formed graphs, by which the infrastructure provider can limit its proof computations to the decomposition-level required to answer the tenant's question. For instance, if the tenants asks "Were all my resources geolocated in Europe?" (a property on vertex labels), the provider computes the proof of representation* possession *including vertex decomposition* vertices*, but no edge decomposition.*

1. Commitments.

The zero-knowledge proofs operate on committed values, which requires us to compute commitments for all values referenced in the proof. The prover computes Integer commitments on the exponents of all vertex and edge bases. First, the prover computes commitments on all messages to allow their decomposition into components. $\mathsf{Commit}(\mathsf{possession}(\mathcal{G}, \sigma, \mu_i); r_i, r_{(i,j)})$ are in $\mathbb{Z}_N^*$ with uniformly chosen randomness $r_i, r_{(i,j)} \in \{0,1\}^\ell$:

$$C_i = R^{e_i \Pi_{k \in f_{\mathcal{V}}(i) e_k}} S^{r_i}$$
$$C_{(i,j)} = R^{e_i e_j \Pi_{k \in f_{\mathcal{E}}(i,j) e_k}} S^{r_{(i,j)}}$$

For each vertex i, the prover computes $\mathsf{Commit}(\mathsf{vertices}(\mathcal{G}); \breve{r}_i)$ in $\mathbb{Z}_N^*$ on vertex attribute and identifier using uniformly-chosen randomness $\breve{r}_i \in \{0,1\}^\ell$:

$$\breve{C}_i = R^{e_i} S^{\breve{r}_i}.$$

For edges (i, j), the commitments are in $\mathbb{Z}_N^*$ with uniformly chosen randomness $\breve{r}_{(i,j)}, \dot{r}_{(i,j)} \in \{0,1\}^\ell$:

$$\mathsf{Commit}(\mathsf{edges}(\mathcal{G}); \breve{r}_{(i,j)}, \dot{r}_{(i,j)}) :$$

$$\breve{C}_{(i,j)} = R^{e_i e_j} S^{\breve{r}_{(i,j)}} \quad \text{and} \quad \dot{C}_i = R^{e_i} S^{\dot{r}_{(i,j)}}.$$

2. Proof of knowledge.

We construct the proof of possession and well-formedness step by step, where it is understood the proofs will be done in one compound proof of knowledge with *referential integrity between the secret exponents*. Let us consider a proof fragment for vertices i, j and an edge (i, j) committed in a graph commitment C (the same proof structure is used for CL-Signatures).

2.1 Proof of representation. We prove that commitment C can be decomposed into commitments C_i, C_j, one for each vertex i, j and one commitment $C_{(i,j)}$ for each edge (i, j):

$$PK\{(\mu_i, \mu_j, \mu_{(i,j)}, \rho, \rho_i, \rho_j, \rho_{(i,j)}) : \pmod N$$
$$C \equiv \pm \prod_{i,j} R_{\pi(i)}^{\mu_i} R_{\pi(j)}^{\mu_j} \prod_{(i,j)} R_{\pi(i,j)}^{\mu_{(i,j)}} S^{\rho} \ \wedge \tag{1}$$
$$C_i \equiv \pm R^{\mu_i} S^{\rho_i} \ \wedge \ C_j \equiv \pm R^{\mu_j} S^{\rho_j} \ \wedge \tag{2}$$
$$C_{(i,j)} \equiv \pm R^{\mu_{(i,j)}} S^{\rho_{(i,j)}} \}. \tag{3}$$

The same proof of representation can be applied to graph signatures to prove the predicate possession:

$$Z \equiv \pm \prod_{i,j} R_{\pi(i)}^{\mu_i} R_{\pi(j)}^{\mu_j} \prod_{(i,j)} R_{\pi(i,j)}^{\mu_{(i,j)}} A'^{\varepsilon'} S^{\rho'}$$

2.2 Vertex composition. Second, we need to show properties of the vertex composition that the encoding for each vertex i contains exactly one vertex identifier $e_i \in \Xi_{\mathcal{V}}$ and zero or multiple label representatives $e_k \in \Xi_{\mathcal{L}}$. We show this structure with help of

[1] The randomization is required for multi-use unlinkability once a single graph signature is used in proofs with multiple verifiers. While the extended version specifies this in detail, we keep it her to maintain consistency.

the commitments $\breve{C}_i$ and set membership and prime-encoding OR proofs. This proof is executed for all vertices.

$$PK\{(\forall i : \varepsilon_i, \breve{\rho}_i, \gamma_i, \rho_i') : \pmod N$$
$$\breve{C}_i \equiv \pm R^{\varepsilon_i} S^{\breve{\rho}_i} \ \wedge \ C_i \equiv \pm \breve{C}_i^{\gamma_i} S^{\rho_i'} \ \wedge \quad (4)$$
$$\gamma_i[C_i] \subseteq \Xi_{\mathcal{L}} \ \wedge \ \varepsilon_i[\breve{C}_i] \in \Xi_{\mathcal{V}}\}. \quad (5)$$

Clause 4 establishes the predicate $\mathsf{vertices}(\mathcal{G}, \mu_i, \varepsilon_i, \gamma_i)$, where its second sub-clause links to the possession commitments.

2.3 Edge composition. Third, we prove the structure of each edge (i, j) over the commitments $C_{(i,j)}$, showing that each commitment contains exactly two vertex identifiers $e_i, e_j \in \Xi_{\mathcal{V}}$ as well as zero or more label representative $e_k \in \Xi_{\mathcal{L}}$:

$$PK\{(\dot{\varepsilon}_i, \dot{\varepsilon}_j, \rho_{(i,j)}, \gamma_{(i,j)}, \rho'_{(i,j)}) : \pmod N$$
$$\breve{C}_{(i,j)} \equiv \pm \dot{C}_i^{\dot{\varepsilon}_j} S^{\rho_{(i,j)}} \ \wedge \ \dot{C}_i \equiv \pm R_i^{\dot{\varepsilon}_i} S^{\dot{\rho}_i} \quad (6)$$
$$C_{(i,j)} \equiv \pm \breve{C}_{(i,j)}^{\gamma_{(i,j)}} S^{\rho'_{(i,j)}} \ \wedge \quad (7)$$
$$\gamma_{(i,j)} \subseteq \Xi_{\mathcal{L}} \ \wedge \ \dot{\varepsilon}_j[\breve{C}_{(i,j)}] \in \Xi_{\mathcal{V}}\}\}. \quad (8)$$

Clauses 6 and 7 realize the predicate $\mathsf{edges}(\mathcal{G}, \varepsilon_i, \varepsilon_j, \gamma_{(i,j)})$, for which Clause 7 binds the edges commitments to the possession commitments. Clauses 5 and 8 establish that identifiers and labels are valid.

2.4 Pair-wise difference. We give the proof of pair-wise difference of vertices for completeness; it is not required in the application scenario in which we trust the audition system to only sign well-formed graphs. We show that the vertex representatives are pair-wise co-prime over the commitments $\breve{C}_i$ and $\breve{C}_j$.

$$PK\{(\forall i, j : \alpha_{i,j}, \beta_{i,j}, \rho_{i,j}) :$$
$$R \equiv \pm \breve{C}_i^{\alpha_{i,j}} \breve{C}_j^{\beta_{i,j}} S^{\rho_{i,j}} \pmod N\}. \quad (9)$$

THEOREM 5.1 (PROOF OF WELL-FORMEDNESS).
The compound proof of knowledge establishes the well-formedness of an encoded graph according to Def. 1. [Proof in [26]]

6. PROOFS OF GRAPH PROPERTIES

Having established encoding and foundational bootstrapping cycle of proof of representation and issuing, we continue to establish a library of graph proof predicates. First, we explore with proofs over vertex and edge sets, including coverage and pair-wise disjointness, which are the basis of partition proofs. Second, we discuss different proofs over presence and absence of labels. Third, we establish results on connectivity and isolation in undirected graphs.

6.1 Sets as Cumulative Products

The idea to reach a compact set representation is: given that the elements of the set are all primes, we can safely multiply all set members without losing access to individual values. Hence, we represent a set of vertices $V \subseteq \mathcal{V}$ with ℓ elements by the cumulative product of its vertex identifiers:

$$E_V = \Pi_{i \in V} e_i.$$

The normal set representation only includes the vertex identifiers of the vertex set, the extended set representation also all associated vertex labels. An *edge set* is represented at the cumulative product of all the vertex identifiers of the edges involved. We will use proofs over cumulative products repeatedly and establish a generic interface for those. In the following, we rename the vertex identifier indices to range over $1, \ldots, \ell$, without loss of generality.

1. Commitments. The prover commits to the vertex set representation as the cumulative product E_V. To prepare the proof of representation the prover establishes intermediate commitments $\mathsf{Commit}(\mathsf{set}(\mathcal{V}, V); \breve{r}_1, \ldots, \breve{r}_\ell)$ in $\mathbb{Z}_N^*$ on partial products using uniformly chosen randomness $\breve{r}_1, \ldots, \breve{r}_\ell$:

$$\breve{C}_{V,1} = R^{e_1} S^{\breve{r}_1}, \ldots, \breve{C}_{V,\ell} = R^{\Pi_1^\ell e_i} S^{\breve{r}_\ell}.$$

2. Proof of Representation. To establish a proof of representation of a $\mathsf{set}(\mathcal{V}, V)$ that a cumulative product E_V is composed of the identifiers of certified vertices, the prover engages with the verifier in the following proof of knowledge over the cumulative product:

$$PK\{((\forall i : \mu_i, \varepsilon_i), \breve{\rho}_1, \ldots, \breve{\rho}_\ell) : \pmod N$$
$$\mathsf{possession}(\mathcal{G}, \sigma, \mu_i) \ \wedge \ \mathsf{vertices}(\mathcal{G}, \mu_i, \varepsilon_i)$$
$$\breve{C}_{V,1} \equiv \pm R^{\varepsilon_1} S^{\breve{\rho}_1} \ \wedge$$
$$\breve{C}_{V,2} \equiv \pm \breve{C}_{V,1}^{\varepsilon_2} S^{\breve{\rho}_2} \ \wedge \ldots$$
$$\breve{C}_{V,\ell} \equiv \pm \breve{C}_{V,\ell-1}^{\varepsilon_\ell} S^{\rho_\ell}$$

REMARK 3 (EDGE SETS).
Edge sets can be represented as products of their vertex identifiers, as well. This is a degenerate representation as it does not maintain the edge structure, but only the vertices present, however as we shall see in §6.2.1 it serves its purpose for edge partitions.

6.1.1 Coverage

We want to find out whether a given set of vertex sub-sets completely covers the entire graph. The predicate cover establishes $\bigcup\{V_1, V_2, \ldots, V_k\} \supseteq \mathcal{V}$. As the set is represented as product of vertex identifiers, this equivalent to the product of all vertex identifiers of the graph $\chi_{\mathcal{V}}$ dividing the product representation of the sets:

$$\chi_{\mathcal{V}} | \prod_{i=1}^{k} E_{V_i} \Leftrightarrow \exists a : a\chi_{\mathcal{V}} = \prod_{i=1}^{k} E_{V_i},$$

where $E_{V_i} = \prod_{j \in V_i} e_j$. The product representation of $\Xi_{\mathcal{V}}$ is signed with $C_{\mathcal{V}}$ as part of the issuing. Thus, given proofs for $\mathsf{set}(\mathcal{G}, V_i)$, we compute $\mathsf{set}(\mathcal{G}, \{V_1, \ldots, V_k\})$, which results in a commitment on the cumulative product of all sets:

$$C_{\bar{V}} = R^{\prod_{i=1}^{k} E_{V_i}} S^{\bar{r}}.$$

To complete the proof of coverage, we prove that the cumulative product over all subset in the commitment $C_{\bar{V}}$ divides the product of all vertex identifiers $C_{\mathcal{V}}$.

$$PK\{(\alpha, \rho) :$$
$$\mathsf{set}(\mathcal{G}, V_1) \ \wedge \cdots \wedge \ \mathsf{set}(\mathcal{G}, V_k) \ \wedge$$
$$\mathsf{set}(\mathcal{G}, \{V_1, \ldots, V_k\}) \ \wedge$$
$$C_{\bar{V}} \equiv \pm C_{\mathcal{V}}^{\alpha} S^{\rho} \pmod N$$

6.1.2 Pair-wise Disjointness

The predicate disjoint establishes that vertex sets have no joint vertex, $\bigcap\{V_1, V_2, \ldots, V_k\} = \emptyset$. Recall that the sets are products of primes. Hence, we can use the fact that two vertex sets $V_i \subseteq V$ and $V_j \subseteq \mathcal{V}$ are pair-wise disjoint if their product representations are coprime:

$$V_i \cap V_j = \emptyset \quad \Leftrightarrow \quad \mathsf{gcd}(E_{V_i}, E_{V_j}) = 1.$$

If two sets share a vertex, i.e., a prime factor, this equation cannot be proven.

Based on given commitments C_{V_i} and predicates $\mathsf{set}(\mathcal{G}, V_i)$ and $\mathsf{set}(\mathcal{G}, V_j)$, we establish coprimality and thereby disjointness with Bézout's Identity,

$$\mathsf{gcd}(E_{V_i}, E_{V_j}) = 1 \quad \Leftrightarrow \quad aE_{V_i} + bE_{V_j} = 1,$$

resulting in the following proof:

$$\begin{aligned} PK\{(\forall i,j : \alpha_{i,j}, \beta_{i,j}, \rho_{i,j}) : \quad & (\mathrm{mod}\ N) \\ \mathsf{set}(\mathcal{G}, V_1) \ \wedge \cdots \wedge \ & \mathsf{set}(\mathcal{G}, V_k) \ \wedge \\ \forall i,j : R \equiv \pm C_{V_i}^{\alpha_{i,j}} C_{V_j}^{\beta_{i,j}} S^{\rho_{i,j}} & \end{aligned}$$

6.1.3 Partition

In a partition proof we seek to establish that a several sets of vertices cover the entire graph and that they are all pair-wise disjoint. Hence, we combine both proofs above on coverage and pair-wise disjointness operating on commitments for the predicate $\mathsf{set}(\mathcal{G}, V_i)$.

6.2 Statements over Edges

We now turn our attention to edges and connectivity, where we start with existence of a single before we proceed with connectivity and isolation.

Here we seek to prove that there exists an edge between two secret vertices, that is, the edge predicate. Thus, we need to show that there exists an edge base with an exponent E which encodes the edge (i, j). This, in turn, means that the product of the edge's vertex identifiers divides the exponent.

$$(e_i e_j) | E \Leftrightarrow \exists a : a(e_i e_j) = E.$$

This is realized by the prover engaging with the verifier in the following proof of knowledge:

$$\begin{aligned} PK\{((\forall i : \mu_i), \mu', \rho') : & \\ \mathsf{possession}(\mathcal{G}, \mu_i) \ \wedge & \\ C_{(i,j)} \equiv \pm (R^{e_i e_j})^{\mu'} S^{\rho'} \quad & (\mathrm{mod}\ N)\}. \end{aligned}$$

6.2.1 Connectivity

A major area of interest for hidden-graph proofs is connectivity: How can we show that different vertices of a hidden graph are connected by a chain of ℓ edges? The predicate $\mathsf{connected}(\mathcal{G}, i, j, \ell)$ means that there exists a sequence of at most ℓ edges, such that the end vertex of one edge is the start vertex of the next.

EXAMPLE 1.
Let us consider the following chain of connected edges:

$$(e_i \cdot e_1), (e_1 \cdot e_2), (e_2 \cdot e_3), (e_3 \cdot e_j)$$

We observe that two vertices are connected if and only if there exists a sequence of edge products, such that their the edges match in the joint vertex identifier. By that, we have for instance for the first pair of edges:

$$e_1 | (e_i \cdot e_1) \wedge e_1 | (e_1 \cdot e_2)$$

The idea here is to prove connectivity by showing that the vertex identifier of the connection point between two adjacent edges divides both edge representations. We will chain these divisibility proofs.

Recall that the predicate $\mathsf{edges}(\mathcal{G}, \varepsilon_i, \varepsilon_j, \gamma_{(i,j)})$ from cf. §5.3.1 computes the following commitments in $\mathbb{Z}_N^*$:

$$\breve{C}_{(i,j)} = R^{e_i e_j} S^{\breve{r}_{(i,j)}} \quad \text{and} \quad \dot{C}_i = R^{e_i} S^{\dot{r}_{(i,j)}}$$

for each edge (i, j), allowing us to compose statements over adjacent edges via $\breve{C}_{(i,j)}$ and the subsequent $\dot{C}_j$. We show that a joint factor ε divides adjacent edges (i, j) and (j, k) with the following proof of knowledge:

$$\begin{aligned} PK\{(\varepsilon, \rho, \rho') : \quad & (\mathrm{mod}\ N) \\ \breve{C}_{(i,j)} \equiv \pm \dot{C}_i^{\varepsilon} R^{\rho} \ \wedge \ & \dot{C}_j = R^{\varepsilon} S^{\rho'}\}. \end{aligned}$$

6.2.2 Isolation

Proving isolation of sub-graphs in zero-knowledge efficiently is a daunting task as the prover could "forget" an edge, which normally forces the proof to iterate over all edges (quadratic complexity). We can do better: We can partition the graph into two edge sets, each containing one vertex of the isolation claim. If there is any connection between these two sets, then there must be an edge from one set two the other, which means that they have a joint factor.

For vertices i and j $\mathsf{isolated}(i, j)$ means that there exists no connected path between both vertices i and j. For undirected graphs, isolation means that the vertices i and j are in separate sub-graphs. Two vertices i and j are isolated, if there exists a bi-partition of the edge set $V' \cup V'' = \mathcal{E} \wedge V' \cap V'' = \emptyset$, such that without loss of generality $i \in V'$ and $j \in V''$. Recall that §6.1 represents an edge set in a degenerate form, as the product of the edges' vertex identifiers E' and E''. We obtain commitments and proofs for the predicates $\mathsf{set}(\mathcal{G}, V')$ and $\mathsf{set}(\mathcal{G}, V'')$ which give us two commitments in $\mathbb{Z}_N^*$:

$$\breve{C}_{E'} = R^{E'} S^{\breve{r}'} \quad \text{and} \quad \breve{C}_{E''} = R^{E''} S^{\breve{r}''}.$$

We can derive coverage already from the fact that all commitments of the predicate edges are used to establish the cumulative products for both edge sets. The disjointness of the edge-set bi-partition gives the isolation result, which we prove by showing that both products are coprime:

$$\mathsf{gcd}(E', E'') = 1 \quad \Leftrightarrow \quad \exists a, b : aE' + bE'' = 1.$$

The complete the proof of the predicate isolated, the prover and the verifier engage in the following proof of knowledge:

$$\begin{aligned} PK\{(\alpha, \beta, \rho) : \quad & (\mathrm{mod}\ N) \\ \mathsf{set}(\mathcal{G}, V') \ \wedge \ & \mathsf{set}(\mathcal{G}, V'') \ \wedge \\ R \equiv \pm \breve{C}_{E'}^{\alpha} \breve{C}_{E''}^{\beta} S^{\rho} & \end{aligned}$$

REMARK 4 (ISOLATION ON VLAN IDS).
The isolation *predicate constructed in this section argues over the edges only. In actual topologies, such as infrastructure clouds, it is however the case that graph labels are important to decide upon connectivity and isolation. This holds in particular for the VLAN IDs of virtualized infrastructures, which allow communication if components have matching VLAN IDs. The* isolation *predicate can be easily extended to show pair-wise disjointness for labels as well.*

7. PERFORMANCE EVALUATION

7.1 Asymptotic Evaluation

We display the efficiency analysis for the proof predicates in Table 2, where each row shows the overhead over the basis predicate stated in the first column. We measure computational complexity in multi-base and modular exponentiations. The *communication complexity* is dominated by the transmitted group elements from $\mathbb{Z}_N^*$, which is equal to the number of multi-base exponentiations (one for each Integer and Schnorr proof commitment). In the scenario we consider in this paper, the verifier trusts the issuer to only certify

well-formed graphs. As we do not need to prove well-formedness of a hidden graph, we have *linear complexity* for all proofs.

The modular exponentiations for message bases R_i have small exponents of size $\ell_{\mathcal{M}} \ll \ell_N$, where the parameter $\ell_{\mathcal{M}}$ can be chosen similarly small as in Direct Anonymous Attestation [8]. Implementing the proofs of representation as multi-base exponentiations will reduce the number of multiplications significantly and thereby offer a significant speed-up. In addition, the Σ-proofs employed in this work benefit from batch-proof techniques [32]. The system allows signature proofs of knowledge with the Fiat-Shamir heuristic [20], which can be computed offline.

We note that the public key size (that is the certification of vertex and label identifiers) is linear in the maximal number of vertices and label types, a case in point to partition the infrastructure representation and cover it with multiple (hierarchical) graph signatures.

7.2 Experimental Evaluation

Figure 2 contains an experimental performance analysis, based on the following parameters: The issuer has established the setup with Quadratic Residues QR_N under a special RSA modulus N as specified by the Identity Mixer Library, where the modulus length $\ell_n = 2048$ bits and system parameters (ℓ_v, etc.) are chosen exactly as prescribed in the library setup [27]. Bases for vertex and edge encoding are computed according to the library setup. The performance analysis is executed on 64-bit Java JDK 1.7.13 on a Windows 7 SP 1 Thinkpad X220 Tablet, on Intel CPU i5-2520 with 2.5 GHz, 8 GB RAM, where all computations are performed on a *single processor core only*. This is a very conservative estimate as an infrastructure provider can use multiple cores and parallelize the computations on all levels. The performance analysis uses the math utility functions of the Identity Mixer Library for the computation of randomness and exponentiations, that is, its MultiExp facility.

The experiments performed on a prototypical implementation of computations required for the graph signature scheme, that is, on representative computations of commitments and a proof of knowledge thereof. For a commitment, we would have a structure $C := R_0^{m_0} \cdots R_\ell^{m_\ell} S^v \bmod N$, where the exponents are uniform random bit-strings of the prescribed length and number (as in the actual Schnorr proof witnesses). The simulation uses random graphs with specified number of vertices n and a derived number of edges $m := 2n$ as major independent variable (on the x-axis), the dependent variable is computation time in milliseconds (in log-scale on the y-axis). Again, these computations give a conservative estimate of the expected computation time: issuing and proofs that disclose attributes operate on smaller exponents, by which they are more efficient. Any implementation on actual graphs, with multiple cores or parallelization strategies will perform much better.

Discussion.

If we consider the computations done by the issuer, we see that the issuer can sign large graphs with $10,000$ vertices and $20,000$ edges (a realistic size for an in-house cloud) in 532 milliseconds, if the issuer uses his knowledge of the discrete-logarithms between generator and bases for optimization[2]. This means the issuer can sign large graphs rapidly. The user's proof of representation of a signature with $10,000$ vertices and $20,000$ edges, in turn, can be performed in a time of 238 seconds, ≈ 4 minutes, which is feasible again. We believe that the usual use case will be that the size of the user-contributed subgraph is very small compared to the size of the issuer-contributed subgraph. Such use cases can thereby be implemented efficiently.

[2]An implementation using the standard CL-issuing without this optimization will use 476 seconds, ≈ 8 minutes

EXAMPLE 2 (VIRTUALIZED INFRASTRUCTURE).
Let us consider the scenario of the system model of §2 again. The auditing system certifies the virtualized infrastructure after a security analysis and issues a graph signature to the infrastructure provider. The auditor gets privileged access to the infrastructure, hence, no user-generated proof of well-formedness of a committed graph is needed. The auditing system can issue signatures with a frequency in the order of seconds. Upon tenant request with a time index t, the infrastructure provider looks up the appropriate graph signature for time t to prove to tenants in zero-knowledge that security properties, such as isolation from competitors, are fulfilled. The provider fixes a time of infrastructure state and corresponding signature, and computes the proof of knowledge disclosing only the properties in question, in the order of minutes. We expect the tenants will only query the provider sporadically for proofs over the graph signatures.

REMARK 5 (SCALABILITY).
First, we note that the core algorithms have linear complexity in number of vertices and edges, with a low slope constant. Second, we observe that the decomposition is highly parallelizable: All information (message blocks and random exponents) is known a priori and there are no state inter-dependencies between commitment computations or Schnorr proofs: The commitments can be computed in parallel and once they are computed the Schnorr proofs can be computed in parallel, as well. This requires computing all the witnesses first, then synchronize the computation results to compute the challenge with the Fiat-Shamir heuristic, to finally compute all the responses in parallel again. Thus, we believe graph signatures to be scalable.

8. RELATED WORK

Structural and Topological Cloud Assurance.

Whereas past work was often on hardening hypervisors and preventing isolation breaches in the systems themselves, there are multiple lines of work in the (structural) security assurance of virtualized infrastructures assuming that the hypervisors and management hosts are correct, yet might be misconfigured. Zhai et al. [38] proposed a structural reliability auditor, which discovers dependencies between cloud components. This was later extended by Xiao et al. [37] to achieve this privately in secure multi-party computation (SMPC). Their work argues over the dependency graph of cloud components, a data structure which can be directly implemented in the graph signature scheme presented here. The goal of our works is different, however: Whereas they allow multiple parties (e.g., providers) to compute jointly whether there are hidden dependencies without disclosing secret information, our work proves properties of audited dependency graph, e.g., absence of a single-point-of-failure or isolation from a particular dependency, in zero-knowledge.

SAVE and subsequent works of Bleikertz et al. [4, 3] pursue security analysis of virtualized infrastructures based on a graph representation. Their work employs discovery probes or sensors for virtualized infrastructure change events to maintain a graph representation in sync with the actual topology of the infrastructure. Their work first focused on information flow analysis by graph coloring, yet has been extended subsequently with general purpose model checkers, verifying the graph model against a high-level security policy in abstract language, either the set rewriting dialect

Table 2: Efficiency of proofs of predicates in multi-base and modular exponentiations (MultiExps and ModExps). For a simple graph holds $m \leq \frac{n(n-1)}{2}$. Note that the constant k is the number of sets considered, e.g., $k = 2$ for a bipartition. In practice holds $k \ll \ell \leq n$ and $O(k\ell) = O(n)$, hence, all proofs are linear in n or m.

Predicate	Basis	Commitments	MultiExps	ModExps	
		#	#	#	O
possession$(\mathcal{G}, \sigma, \mu_i)$		$n+m$	$2n+2m+1$	$5n+5m+2$	$O(n+m)$
vertices$(\mathcal{G})$	possession	n	$3n$	$6n$	$O(n)$
edges$(\mathcal{G})$	possession	$2m$	$4m$	$8m$	$O(m)$
set$(\mathcal{V}, V)$, with $\ell = \lvert V \rvert$	vertices	ℓ	2ℓ	4ℓ	$O(\ell) = O(n)$
cover$(\mathcal{V}, V_1, \ldots, V_k)$	vertices	$k(\ell+1)$	$2k(\ell+1)+1$	$4k(\ell+1)+2$	$O(k\ell)$
disjoint$(\mathcal{V}, V_1, \ldots, V_k)$	vertices	$k\ell$	$k^2+2k\ell$	$3k^2+4k\ell$	$O(k^2+k\ell)$
partition$(\mathcal{V}, V_1, \ldots, V_k)$	vertices	$k(\ell+1)$	$k^2+2k(\ell+1)$	$3k^2+4k(\ell+1)+2$	$O(k^2+k\ell)$
edge$(\mathcal{G}, i, j)$	possession	0	1	2	$O(1)$
connected$(\mathcal{G}, i, j, \ell)$	edges	0	2ℓ	4ℓ	$O(\ell) = O(m)$
isolated$(\mathcal{G}, i, j)$	edges	m	$2m+1$	$4m+3$	$O(m)$

VALID [2] or graph rewriting languages. The most recent advances in that line of work include a fast differential analysis on change events. Their graph representation and information flow overlay can serve as basis for this work's auditing system and issuing of graph signatures. At the same time, the tenant can query for zero-knowledge proofs based on the security policies employed. By that, our work could extend theirs by a projection of trust: Whereas Bleikertz et al. have the infrastructure provider specify the security policy for the analysis, run the analysis tool, and have the tenant trust that all this was done correctly, the provider can now delegate running the tool to a trusted auditor and satisfy the tenants own security policies in zero-knowledge.

TPM-protected host-based monitoring of virtualized infrastructures offers an alternative to structural security assurance. *Cloud Verifier* [34], for example, allows a remote tenant to specify integrity criteria for which Cloud Verifier will monitor a node server. This approach employs an integrity verification proxy as well as TPM support for attestation. The monitored policies include VMinfos, the network and host security policies and the host memory, all of which are limited to host properties. Our approach of graph signatures is complementary to host-based auditing, as it allows for proofs over the inter-connectivity of components beyond the host's boundaries. For instance, a host's assurance of a correct VLAN ID configuration is not sufficient for VLAN-based network isolation, as there could be a VLAN ID collision anywhere in the network. Similarly, a host's assurance that it is connected to the correct remote storage partition does not guarantee that there is no other component connected to that partition.

Zero-Knowledge Proofs and Signatures on Graphs.

Zero-knowledge proofs on graphs and their properties is a classic area of research and have been instrumental in showing that there exist zero-knowledge proof systems for all NP languages, e.g., [22, 5] Both proofs use a metaphor of locked boxes to construct known-graph proofs of Graph 3-Colorability (G3C) or Directed Hamiltonian cycles (DHC). The constructions focus on zero-knowledge proofs of knowledge and do not cater for a level of indirection through a signature scheme or proofs of knowledge on graph properties in a hidden-graph setting.

A related notion to full graph signatures is transitive or homomorphic signature schemes, such as [29, 28, 1]. They are concerned with the transitive closure of signatures on graph elements, such that from signatures from edges (i, j) and (j, k) everybody can derive a valid signature on the edge (i, k). These signature schemes have the advantage that one can produce a signature of the transitive path over multiple edges. Therefore, they allow showing signatures equivalent to the connected predicate without disclosing the number of edges on the path and without overhead because of path length. The constructions are not meant to be on committed graphs and consider the signatures as public information. They have limited support for labels and do not have provisions for proofs of isolation as signatures could be withheld.

Authenticated and Verifiable Graph Computations.

Authenticated data structures for graph connectivity have been thoroughly investigated by Goodrich et al. [25], whose approach is restricted to hash-based authentication and does not yield zero-knowledge properties. Subsequent work [24] has been applied to Web-content searching and proposed an authenticated web crawler, whose problem statement also exhibits the highly dynamic nature of the authenticated data structure. The authentication is based on a root-signed Merkle tree, which prevents a zero-knowledge access to the underlying data structure as pursued in this work. The intersection proofs based on the certification of succinct relations presents an interesting avenue for future graph proofs.

One can realize graph proofs with Verifiable Computation (VC), in which a client outsources a computation to an untrusted worker and is subsequently enabled to verify the correctness of the computation result. The recent Pinocchio scheme by Parno et al. [30] employs quadratic programs to achieve a highly efficient verification and constant-size proofs of computation. Whereas the proofs of computation can be made statistically zero-knowledge at low cost, the zero-knowledge certification of the auxiliary worker input (in our case the topology graph) is not considered. However, VC may provide an alternative to compiling Σ-proofs on the graph signature. In parallel to this work, Zhang et al. [39], have proposed ALITHEIA, a verifiable graph processing framework and offered a comprehensive comparison of Verifiable Computing results on graphs. Zhang et al. state that the verifiable execution of BFS on Pinocchio suffered from the need to be expressed as a circuit and the involved quadratic blow-up, which resulted in an implementation scaling up only to 50 vertices. The 15GB-RAM experiment machine was running out of memory when compiling the certifying algorithm code on graphs with more than $10,000$ vertices. Thus, whereas the verification time offered by the ALITHEIA graph VC only grows sub-linearly and can be considerably faster than the compiled Schnorr proofs we use in this approach, this comes at a high storage cost for the server.

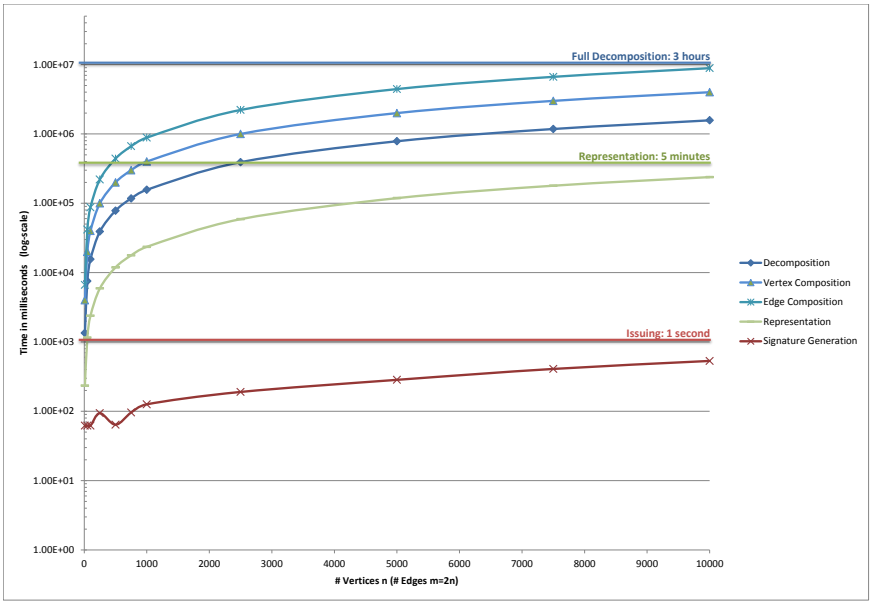

Figure 2: Experimental performance analysis with a secure modulus length of 2048 bits, in the worst case of a non-parallelized computation on a single processor core. x-axis contains the number of vertices n and the y-axis a log-scale of computation time in milliseconds. Blue colors denote provider computations to prove properties of a committed graph, where the green line shows a proof of representation of a graph signature. Red colors denote auditing system/issuer computations to sign the graph.

9. CONCLUSION AND FUTURE WORK

We have introduced a signature scheme on committed graphs together with a framework of proof predicates on sets, connectivity and isolation. The scheme covers undirected, unlabeled and labeled graphs and enables honest-verifier zero-knowledge proofs of knowledge over graph properties, while keeping the graph itself confidential. It constitutes a building block to overcome the requirement gap between the confidentiality requirements of a provider and the integrity requirements of a tenant.

The signature scheme and its proofs are created in a special RSA setting; their security is based on the Strong RSA assumption. The signature scheme is based on the Camenisch-Lysyanskaya (CL) signature scheme [11] and its existential unforgeability directly derived from that. The proofs can be transformed to signature proofs of knowledge with the Fiat-Shamir [20] heuristic. The constructions for the proof predicates are efficient and practical; their performance can be vastly improved with parallelization.

As future work, we see great potential in linking the graph signatures to Direct Anonymous Attestation (DAA) [8]. This allows the combination of attestation results for system components (e.g., physical and virtual machines) with statements over the system topology. Furthermore, we believe that a systematic parallelization and a differential approach will allow graph signatures and corresponding proof systems perform well, by which we will pursue a cloud implementation of the signing and decomposition proofs.

Acknowledgments

The author's research on graph signatures currently supported by the EU FP7 FutureID project (`http://futureid.eu`) under GA n^o 318424. We are grateful for the initial discussions with Jens Groth and Jan Camenisch. This work has benefited from the reviews and insightful comments of the anonymous reviewers of EUROCRYPT 2014, the IEEE Security and Privacy Symposium 2014, ACM CCS 2014 and ACM CCSW 2014.

10. REFERENCES

[1] BELLARE, M., AND NEVEN, G. Transitive signatures based on factoring and rsa. In *Advances in Cryptology — ASIACRYPT 2002*. Springer, 2002, pp. 397–414.

[2] BLEIKERTZ, S., AND GROSS, T. A Virtualization Assurance Language for Isolation and Deployment. In *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY'11)* (Jun 2011), IEEE.

[3] BLEIKERTZ, S., GROSS, T., AND MÖDERSHEIM, S. Automated Verification of Virtualized Infrastructures. In *ACM Cloud Computing Security Workshop (CCSW'11)* (Oct 2011), ACM.

[4] BLEIKERTZ, S., GROSS, T., SCHUNTER, M., AND ERIKSSON, K. Automated Information Flow Analysis of Virtualized Infrastructures. In *16th European Symposium on Research in Computer Security (ESORICS'11)* (Sep 2011), Springer.

[5] BLUM, M. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians* (1986), vol. 1, p. 2.

[6] BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology — EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 431–444.

[7] BRANDS, S. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology — EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 318–333.

[8] BRICKELL, E., CAMENISCH, J., AND CHEN, L. Direct anonymous attestation. In *Proc. 11th ACM Conference on Computer and Communications Security* (2004), acm press, pp. 225–234.

[9] CAMENISCH, J., CHAABOUNI, R., AND SHELAT, A. Efficient protocols for set membership and range proofs. In *Advances in Cryptology-ASIACRYPT 2008* (2008), Springer, pp. 234–252.

[10] CAMENISCH, J., AND GROSS, T. Efficient attributes for anonymous credentials. *ACM Transactions on Information and System Security (TISSEC) 15*, 1 (2012), 4.

[11] CAMENISCH, J., AND LYSYANSKAYA, A. A signature scheme with efficient protocols. In *Security in Communication Networks SCN 2002* (2003), vol. 2576 of *LNCS*, Springer Verlag, pp. 268–289.

[12] CAMENISCH, J., AND MICHELS, M. Proving in zero-knowledge that a number n is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 107–122.

[13] CAMENISCH, J., AND STADLER, M. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1296 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 410–424.

[14] CHAN, A., FRANKEL, Y., AND TSIOUNIS, Y. Easy come – easy go divisible cash. In *Advances in Cryptology — EUROCRYPT '98* (1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 561–575.

[15] CHAUM, D., AND PEDERSEN, T. P. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92* (1993), E. F. Brickell, Ed., vol. 740 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 89–105.

[16] CRAMER, R., DAMGÅRD, I., AND SCHOENMAKERS, B. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94* (1994), Y. G. Desmedt, Ed., vol. 839 of *LNCS*, Springer Verlag, pp. 174–187.

[17] CSA. The Notorious Nine: Cloud Computing Top Threats in 2013. Tech. rep., Cloud Security Alliance (CSA), Feb. 2013.

[18] DAMGÅRD, I., AND FUJISAKI, E. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology — ASIACRYPT 2002* (2002), vol. 2501 of *Lecture Notes in Computer Science*, Springer.

[19] ENISA. Cloud computing: Benefits, risks and recommendations for information security, rev. b. Tech. rep., European Network and Information Security Agency (ENISA), Dec. 2012.

[20] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 186–194.

[21] FUJISAKI, E., AND OKAMOTO, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 16–30.

[22] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM) 38*, 3 (1991), 690–728.

[23] GOLDWASSER, S., MICALI, S., AND RIVEST, R. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing 17*, 2 (Apr. 1988), 281–308.

[24] GOODRICH, M. T., PAPAMANTHOU, C., NGUYEN, D., TAMASSIA, R., LOPES, C. V., OHRIMENKO, O., AND TRIANDOPOULOS, N. Efficient verification of web-content searching through authenticated web crawlers. *Proceedings of the VLDB Endowment 5*, 10 (2012), 920–931.

[25] GOODRICH, M. T., TAMASSIA, R., AND TRIANDOPOULOS, N. Efficient authenticated data structures for graph connectivity and geometric search problems. *Algorithmica 60*, 3 (2011), 505–552.

[26] GROSS, T. Certification and efficient proofs of committed topology graphs. Cryptology ePrint Archive Report 2014/255, IACR, 2014. `http://eprint.iacr.org/`.

[27] IBM. Specification of the Identity Mixer cryptographic library, v. 2.3.40. Specification, IBM Research, Jan. 2013. `http://prime.inf.tu-dresden.de/idemix/`.

[28] JOHNSON, R., MOLNAR, D., SONG, D., AND WAGNER, D. Homomorphic signature schemes. In *Topics in Cryptology–CT-RSA 2002*. Springer, 2002, pp. 244–262.

[29] MICALI, S., AND RIVEST, R. L. Transitive signature schemes. In *Topics in Cryptology-CT–RSA 2002*. Springer, 2002, pp. 236–243.

[30] PARNO, B., HOWELL, J., GENTRY, C., AND RAYKOVA, M. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy (SP)* (2013), IEEE, pp. 238–252.

[31] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91* (1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 129–140.

[32] PENG, K., BOYD, C., AND DAWSON, E. Batch zero-knowledge proof and verification and its applications. *ACM Transactions on Information and System Security (TISSEC) 10*, 2 (2007), 6.

[33] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (Feb. 1978), 120–126.

[34] SCHIFFMAN, J., SUN, Y., VIJAYAKUMAR, H., AND JAEGER, T. Cloud verifier: Verifiable auditing service for iaas clouds. In *Services (SERVICES), 2013 IEEE Ninth World Congress on* (June 2013), pp. 239–246.

[35] SCHNORR, C. P. Efficient signature generation for smart cards. *Journal of Cryptology 4*, 3 (1991), 239–252.

[36] SHOUP, V. *A Computational Introduction to Number Theory and Algebra (Second Edition)*. Cambridge University Press, 2008. Online `http://www.shoup.net/ntb/`.

[37] XIAO, H., FORD, B., AND FEIGENBAUM, J. Structural cloud audits that protect private information. In *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop* (New York, NY, USA, 2013), CCSW '13, ACM, pp. 101–112.

[38] ZHAI, E., WOLINSKY, D. I., XIAO, H., LIU, H., SU, X., AND FORD, B. Auditing the structural reliability of the clouds. Tech. rep., YALEU/DCS/TR-1479, Department of Computer Science, Yale University, 2013., 2013.

[39] ZHANG, Y., PAPAMANTHOU, C., AND KATZ, J. ALITHEIA: Towards practical verifiable graph processing. In *21st ACM Conference on Computer and Communications Security (ACM CCS'14)* (Nov. 2014), ACM Press.

Geo-Location Separation of Virtualized Systems

Thomas Groß
School of Computing Science, Newcastle University, UK

Abstract

Background. Providers of virtualized infrastructures seek to offer added dependability by geo-spatially distributed deployments. For instance, a provider could offer to host the resources of a tenant in k-out-of-n European countries.

Aim. To prove in zero-knowledge that a deployment represented as a graph is distributed over a k-out-of-n subset of a set of candidate location labels.

Expected Results. We derive a predicate for a graph signature scheme that can efficiently prove separation over geo-locations.

Expected Impact. Cloud providers can thereby convince tenants that the geo-location SLAs they have claimed are fulfilled, without disclosure of the actual deployment plan.

1 Preliminaries

The preliminaries are taken verbatim from Groß introduction of graph signatures [13].

1.1 Assumptions

Special RSA Modulus A *special RSA modulus* has the form $N = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, the corresponding group is called *special RSA group*. *Strong RSA Assumption [1, 12, 15]:* Given an RSA modulus N and a random element $g \in \mathbb{Z}_N^*$, it is hard to compute $h \in \mathbb{Z}_N^*$ and integer $e > 1$ such that $h^e \equiv g \bmod N$. The modulus N is of a special form pq, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. *Quadratic Residues* The set QR_N is the set of Quadratic Residues of a special RSA group with modulus N.

1.2 Integer Commitments

Damgård and Fujisaki [10] showed for the Pedersen commitment scheme [14] that if it operates in a special RSA group and the committer is not privy to the factorization of the modulus, then the commitment scheme can be used to commit to *integers* of arbitrary size. The commitment scheme is information-theoretically hiding and computationally binding. The security parameter is ℓ. The public parameters are a group G with special RSA modulus N, and generators $(g_0, \ldots, g_m)$ of the cyclic subgroup QR_N. In order to commit to the values $(V_1, \ldots, V_l) \in (\mathbb{Z}_n^*)^l$, pick a random $R \in \{0,1\}^\ell$ and set $C = g_0^R \prod_{i=1}^{l} g_i^{v_i}$.

1.3 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof

of knowledge of a discrete logarithm modulo a prime [16] or a composite [10, 12], (2) proof of knowledge of equality of representation modulo two (possibly different) composite [7] moduli, (3) proof that a commitment opens to the product of two other committed values [3, 7], (4) proof that a committed value lies in a given integer interval [2, 7], and also (5) proof of the disjunction or conjunction of any two of the previous [9]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption.

Proofs as described above can be expressed in the notation introduced by Camenisch and Stadler [8]. For instance,

$$PK\{(\alpha,\beta,\delta) : y = g^{\alpha}h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta} \wedge (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers* α*,* β*, and* δ *such that* $y = g^{\alpha}h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta}$ *holds, where* $u \leq \alpha \leq v$*,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities of which knowledge is being proven, while all other values are known to the verifier. We apply the Fiat-Shamir heuristic [11] to turn such proofs of knowledge into signatures on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^{\alpha}\}(m)$. Given a protocol in this notation, it is straightforward to derive an actual protocol implementing the proof.

1.4 Camenisch-Lysyanskaya Signatures

Let us introduce Camenisch-Lysyanskaya (CL) signatures in a Strong RSA setting [6].

Let $\ell_{\mathcal{M}}$, ℓ_e, ℓ_N, ℓ_r and L be system parameters; ℓ_r is a security parameter, $\ell_{\mathcal{M}}$ the message length, ℓ_e the length of the Strong RSA problem instance prime exponent, ℓ_N the size of the special RSA modulus. The scheme operates with a ℓ_N-bit special RSA modulus. Choose, uniformly at random, $R_0, \ldots, R_{L-1}, S, Z \in \mathrm{QR}_N$. The public key $\mathsf{pk}(\mathsf{I})$ is $(N, R_0, \ldots, R_{L-1}, S, Z)$, the private key $\mathsf{sk}(\mathsf{I})$ the factorization of the special RSA modulus. The *message space* is the set $\{(m_0, \ldots, m_{L-1}) : m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}\}$.

Signing hidden messages. On input $m_0, \ldots, m_{L-1}$, choose a random prime number e of length $\ell_e > \ell_{\mathcal{M}} + 2$, and a random number v of length $\ell_v = \ell_N + \ell_{\mathcal{M}} + \ell_r$. To sign hidden messages, user U commits to values V in an integer commitment C and proves knowledge of the representation of the commitment. The issuer I verifies the structure of C and signs the commitment:

$$A = \left(\frac{Z}{C R_l^{m_l} \ldots R_{L-1}^{m_{L-1}} S^{v'}} \right)^{1/e} \bmod N.$$

The user completes the signature as follows: $\sigma = (e, A, v) = (e, A, (v' + R))$.

To verify that the tuple (e, A, v) is a signature on message $(m_0, \ldots, m_{L-1})$, check that the following statements hold: $Z \equiv A^e R_0^{m_0} \ldots R_{L-1}^{m_{L-1}} S^v \pmod N$, $m_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}}$, and $2^{\ell_e} > e > 2^{\ell_e - 1}$ holds.

Theorem 1. *[6] The signature scheme is secure against adaptive chosen message attacks under the strong RSA assumption.*

Proving Knowledge of a Signature. The prover randomizes A: Given a signature (A, e, v), the tuple $(A' := AS^{-r} \bmod N, e, v' := v + er)$ is also a valid signature as well. Now, provided that $A \in \langle S \rangle$ and that r is chosen uniformly at random from $\{0,1\}^{\ell_N + \ell_\varnothing}$, the value A' is distributed statistically close to uniform over $\mathbb{Z}_N^*$. Thus, the user could compute a fresh A' each time, reveal it, and then run the protocol

$$\begin{aligned} PK\{(\varepsilon, \nu', \mu_0, \ldots, \mu_{L-1}) : \\ Z \equiv \pm R_0^{\mu_0} \cdots R_{L-1}^{\mu_{L-1}} A'^{\varepsilon} S^{\nu'} \pmod N \ \wedge \\ \mu_i \in \pm\{0,1\}^{\ell_{\mathcal{M}}} \ \wedge \ \varepsilon \in [2^{\ell_e - 1} + 1, 2^{\ell_e} - 1]\} \end{aligned}$$

1.5 Set Membership from CL-Signatures

Set membership proofs can be constructed from CL-Signatures following a method proposed by Camenisch, Chaabouni and shelat [4]. For a set $\mathcal{S} =$

$\{m_0,\ldots,m_i,\ldots,m_l\}$, the issuer signs all set members m_i in CL-Signatures $\sigma_i = (A,e,v)$ and publishes the set of message-signature pairs $\{(m_i,\sigma_i)\}$ integerly. To prove set membership of a value committed in C, the prover shows knowledge of the blinded signature σ_i' corresponding to the message m_i and equality of exponents with C. We explain this technique in detail in the extended version of this paper and denote a set membership proof $\mu[C] \in \mathscr{S}$, which reads μ encoded in commitment C is member of set $\mathscr{S}$.

1.6 Camenisch-Groß Encoding

The Camenisch-Groß (CG) Encoding [5] establishes structure on the CL message space by encoding multiple binary and finite-set values into a single message, and we will use a similar paradigm to encode graphs efficiently. We explain the key principles briefly and give more details in the extended version of this paper.

The core principle of the CG-Encoding is to represent binary and finite-set attribute values as prime numbers. It uses divisibility and coprimality to show whether an attribute value is present in or absent from a credential. The attribute values certified in a credential, say e_i, e_j, and e_l, are represented in a single message of the CL-Signature, by signing the product of their prime representative $E = e_i \cdot e_j \cdot e_l$ in an Integer attribute. The association between the value and the prime number of the encoding is certified by the credential issuer.

Divisibility/AND-Proof. To prove that a disclosed prime representative e_i is present in E, we prove that e_i divides the committed product E, we show that we know a secret μ' that completes the product:

$$PK\{(\mu',\rho): \quad D \equiv \pm(g^{e_i})^{\mu'}h^{\rho} \pmod N\}.$$

Coprimality/NOT-Proof. We show that one or multiple prime representatives are not present in a credential, we show coprimality. To prove that two values E and F are coprime, i.e., $\gcd(E,F) = 1$, we prove there exist integers a and b such that Bézout's Identity equals 1, where a and b for this equation do not exist, if $\gcd(E,F) > 1$.

$$PK\{(\mu,\rho,\alpha,\beta,\rho'): \quad D \equiv \pm g^{\mu}h^{\rho} \pmod N \\ \wedge\ g \equiv \pm D^{\alpha}(g^{F})^{\beta}h^{\rho'} \pmod N\}.$$

OR-Proof To show that a credential contains an attribute e that is contained in an OR-list, we show there exists an integer a such that $ae = \prod_i^{\ell} e_i$; if e is not in the list, then there is no such integer a as e does not divide the product. We use the notation $\alpha \subseteq \Xi$ for an OR-proof that α contains one or more values of Ξ.

2 Setup

We have three parties:

1. Auditor/Issuer I,
2. Provider/Prover P, and
3. Tenant/Verifier V.

In addition to the setup defined for graph signatures [13], Issuer I certifies a list of vertex labels for locations $\mathscr{L}_{\mathscr{V},L}$ together with a set of prime representatives $\Xi_{\mathscr{L},L}$, disjoint from the set of vertex identifiers.

Without loss of generality, we assume that $\mathscr{L}_{\mathscr{V},L}$ and $\Xi_{\mathscr{L},L}$ contain 193 labels and their representatives, one for each country of the United Nations. For instance, the issuer could allocate the first 193 prime numbers between 2 and 1171 for that purpose. Hence, the locations can be encoded with a bit-length of 11 bits.

3 Certification

In the certification phase of the virtualized infrastructure, Issuer I not only determines the topology of the infrastructure. The issuer also securely determines the geo-location of the physical hosts and (network and storage) devices of the system and, thereby, establishes a label from the set $\mathscr{L}_{\mathscr{V},L}$ and a corresponding prime representative from $\Xi_{\mathscr{L},L}$ for each vertex representing a physical host or device. It is out of the score of this report to define how the geo-location of the devices is determined securely.

We define the label allocation function $f_{\mathcal{V}}$ as follows: For each vertex in the topology graph representing a physical resource, the issuer certifies the product of its vertex identifier and its geo-location label from $\mathscr{L}_{\mathcal{V},L}$. For each vertex in the topology graph representing a virtual resource, the issuer certifies the the product of its vertex identifier and the geo-location of the physical entity hosting it.

As a consequence, each vertex in graph will be labeled with a geo-location from $\mathscr{L}_{\mathcal{V},L}$, either determined for the physical device or inherited from it.

4 Proof of Possession

4.1 Proof of Representation

The proof of separation created by the Prover P starts from a regular proof of possession, where it is not necessary for the core proof to establish an edge decomposition.

$$\begin{aligned} PK\{&(\mu_i, \mu_{(i,j)}, \varepsilon, \nu', \rho, \rho_i) : \\ &Z \equiv \pm R_{\pi(i)}^{\mu_i} \cdots R_{\pi(i,j)}^{\mu_{(i,j)}} \cdots A'^{\varepsilon} S^{\nu'} \pmod N \ \wedge \\ &C_i \equiv \pm R^{\mu_i} S^{\rho_i} \pmod N \end{aligned}$$

4.2 Proof of Vertex Composition

Second, the prover computes a proof of correct vertex composition, which shows that that each message m_i is composed of exactly one vertex identifier $e_i \in \Xi_{\mathcal{V}}$ and exaxtly one label representative $e_k \in \Xi_{\mathscr{L},L}$.

$$\begin{aligned} PK\{&(\varepsilon_i, \check{\rho}_i, \gamma_i, \rho_i') : \\ &\check{C}_i \equiv \pm R^{\varepsilon_i} S^{\check{\rho}_i} \pmod N \ \wedge \\ &C_i \equiv \pm \check{C}^{\gamma_i} S^{\rho_i'} \pmod N. \end{aligned}$$

Note that when the issuer I is trusted to only certify correct graphs then there is no need to prove further properties of the decomposition. It is possible, however, to complement this proof to show that the label and vertex representatives are coming from particular subsets. This can be achieved with an extended proof of knowledge:

$$\begin{aligned} PK\{&(\varepsilon_i, \check{\rho}_i, \gamma_i, \rho_i') : \\ &\check{C}_i \equiv \pm R^{\varepsilon_i} S^{\check{\rho}_i} \pmod N \ \wedge \\ &C_i \equiv \pm \check{C}^{\gamma_i} S^{\rho_i'} \pmod N \ \wedge \\ &\gamma_i[C_i] \subseteq \Xi_{\mathscr{L},L} \ \wedge \ \varepsilon_i[\check{C}_i] \in \Xi_{\mathcal{V}}\}. \end{aligned}$$

4.3 Tools for the Proofs of Separation

The proof of separation if based on a proof of co-primality. Essentially, by proving that vertex representations are coprime we show that they do not share a common location label. As a result, two vertex representations shown to be coprime will be considered separate with respect to their geo-location.

As a principle, all proofs of co-primality and by extension geo-location separation are based on proving that Bézout's Identity equals 1 over the vertex commitments. In a simplified case we can assume without loss of generality that the geo-location labels are the only labels present. Then it is sufficient to reason over the messages m_i and m_j. Recall, that m_i and m_j are coprime, if and only if there exist integers a and b such that

$$1 = am_i + bm_j.$$

As a proof over commitments C_i and C_j holding representations of m_i and m_j, we have:

$$\begin{aligned} PK\{&(\forall i, j : \alpha_{i,j}, \beta_{i,j}, \rho_{i,j}) : \\ &R \equiv \pm C_i^{\alpha_{i,j}} C_j^{\beta_{i,j}} S^{\rho i,j} \pmod N \end{aligned}$$

This proof offers the first method for a proof of geo-location separation based on pair-wise difference. It comes at a cost of three modular exponentiations per vertex pair in question.

5 Proof of k-out-of-n Separation

We base an advanced construction of k-out-of-n separation on existing proof predicates for graph signatures, based on Groß' construction for topology proofs [**?**]. In particular, we use a construction

to represent a set of vertices as a cumulative product, represented by the proof predicate $\mathsf{set}(\mathscr{V}, V)$. We also use a construction to establish a partition over k vertex subsets, represented by the proof predicate $\mathsf{partition}(\mathscr{V}, V_1, \ldots, V_k)$.

5.1 Without Constraints on the Label Set

Here we can assume that n is the size of the entire location label set, $\mathscr{L}_{\mathscr{V},L}$.

To show that the resources are separated over k different geo-locations, the prover proceeds as follows.

1. Designate a partition of k subsets, which separated by geo-location. (This entails that all resources with the same geo-location are assigned to the same subset)
2. Compute a commitment on a cumulative product for each subset, as its representation.
3. Compute a set composition proof for each subset.
4. Compute a proof of pair-wise disjointness for the partition's subsets.

5.2 With Constraints on the Label Set

If $n < |\mathscr{L}_{\mathscr{V},L}|$ and the proof is made with respect to a location label subset $L \subset \mathscr{L}_{\mathscr{V},L}$, then the prover needs to convince the verifier in addition to the partition described above that the labels are from the right subset. This is facilitated with subset proofs as outlined in the introduction of graph signatures [13].

6 Computational Complexity Considerations

While earlier work in this space [**?**,13] offered complexity considerations based on modular exponentiations as unit of computation, these analyses ignore that the exponentiations will largely only be with small exponents (less than 32 bits).

Consequently, we are bound to get a more precise evaluation of the computational complexity by considering the *number of multiplications*. The number of multiplications depends on the size of message exponents, which also determines the size of the randomness chosen for the Σ-proofs.

6.1 Bitlength of Message Exponents

The size of the message exponents for vertices depends on the bitlength of the largest prime representatives for labels and vertices:

$$|m|_2 = |\mathsf{max}(\Xi_{\mathscr{L},L})|_2 + |\mathsf{max}(\Xi_{\mathscr{V}})|_2$$

Hence, the size is dependent on the number of the dictionary of location labels and the maximal graph size. From the Prime Number Theorem [**?**], we know that the n-th prime is approximated by $n \ln n$. Thereby, holds

$$\mathsf{max}(\Xi_{\mathscr{L},L}) \sim |\mathscr{L}_{\mathscr{V},L}| \ \mathsf{ln}\ |\mathscr{L}_{\mathscr{V},L}| \text{and}$$

$$\mathsf{max}(\Xi_{\mathscr{V}}) \sim (|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|) \ \mathsf{ln}\ (|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|).$$

As a result, the maximal bitlength of a message exponent is given by

$$\begin{aligned} |m|_2 &= \mathsf{log}_2(|\mathscr{L}_{\mathscr{V},L}| \ \mathsf{ln}\ |\mathscr{L}_{\mathscr{V},L}|) \\ &\quad + \mathsf{log}_2((|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|) \ \mathsf{ln}\ (|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|)). \end{aligned}$$

Example 1 (Geo-Location Separation by Country)**.** *Let us assume that we have* $|\mathscr{L}_{\mathscr{V},L}| = 193$ *with the location labels encoding all countries of the United Nations (with the first 193 prime numbers). Then,* $|\mathsf{max}(\Xi_{\mathscr{L},L})|_2 \leq 11$.

Consider a graph signature setup prepared to handle graphs with 50.000 vertices.

$$\mathsf{max}(\Xi_{\mathscr{V}}) \sim 50.193 \ \mathsf{ln}\ 50.193 = 540.989.$$

As a result we have

$$|\mathsf{max}(\Xi_{\mathscr{V}})|_2 = \mathsf{log}_2(540.989) \leq 20.$$

Overall, a graph signature scheme for graphs with up to 50.000 *vertices and 193 countries can operate on message exponents with a bitlength of* 31 *bits.*

6.2 Complexity in Multi-Base Exponentiations

For the computational complexity analysis, we consider two parameters $n = |\mathscr{V}|$, number of vertices, and $m = |\mathscr{E}|$, the number of edges. For simplicity, we assume that the graph signature encodes a graph with representations for all vertices.

6.3 Complexity in Multiplications

In addition to the parameters $n = |\mathscr{V}|$ and $m = |\mathscr{E}|$, the number of multiplications depends on the number of certified location labels $\ell_L = |\mathscr{L}_{\mathscr{V},L}|$.

We note that the Camenisch-Lysyanskaya signature scheme specifies minimal sizes for the exponents of e and v and that the Integer Commitment Scheme will depend on a minimal size of randomness r, values that depend on the setup of the CL-signature scheme implied by the modulus bitlength. The following constraints hold by the Identity Mixer Specification 2.3.43:

- $\ell_n = |N|_2 = 2048$
- $\ell_m = 256$.
- $\ell_e = \ell_m + 2 = 597$
- $\ell_v = \ell_n + \ell_m + \ell_{\bar{r}} = 2724$, where $\ell_{\bar{r}}$ is a security parameter (default $\ell_{\bar{r}} = 80$).

In addition, the Identity Mixer library version 2.3.43 specifies the bitlength of the randomness of Damgård-Fujisaki commitments as

$$\ell_r = |r|_2 = \log_2 \lfloor n/4 \rfloor .$$

For modular exponentiations, the computational complexity of the binary square-and-multiply algorithm uses $O(k)$ n-bit multiplications, where k is the bitlength of the exponent.

Example 2 (50.000 Vertices and 193 Location Labels)**.** *In this example, we consider a fully connected graph, such that* $m = \frac{n(n-1)}{2}$. *We have* $\ell_V = 31$, $\ell_E = 40$, $\ell_r = \ell_n - 2 = 2046$. *All other parameters are chosen as in the Identity Mixer Specification version 2.3.43. The proof of possession costs* $2724 + 597 + 2046 \cdot 100.000 + 31 \cdot 150.000 + 40 \cdot 1.249.975.000 = 50.208.253.321$. *The vertex composition costs* $2046 \cdot 150.000 + 31 \cdot 150.000 = 311.550.000$.

References

[1] BARIĆ, N., AND PFITZMANN, B. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 480–494.

[2] BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology — EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 431–444.

[3] BRANDS, S. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology — EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 318–333.

[4] CAMENISCH, J., CHAABOUNI, R., AND SHELAT, A. Efficient protocols for set membership and range proofs. In *Advances in Cryptology-ASIACRYPT 2008* (2008), Springer, pp. 234–252.

[5] CAMENISCH, J., AND GROSS, T. Efficient attributes for anonymous credentials. *ACM Transactions on Information and System Security (TISSEC) 15*, 1 (2012), 4.

[6] CAMENISCH, J., AND LYSYANSKAYA, A. A signature scheme with efficient protocols. In *Security in Communication Networks SCN 2002* (2003), vol. 2576 of *LNCS*, Springer Verlag, pp. 268–289.

[7] CAMENISCH, J., AND MICHELS, M. Proving in zero-knowledge that a number *n* is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 107–122.

[8] CAMENISCH, J., AND STADLER, M. Efficient group signature schemes for large

Table 1: Efficiency of proofs of predicates in multi-base exponentiations (MultiExps) dependent on the number of vertices n and of edges m. k is the number of partitions considered for the separation. We consider it to hold that $k \ll \ell \leq n$ and $O(k\ell) = O(n)$.

Predicate	**Commitments**	**MultiExps**		**ModExps**	
	#	#	O	#	O
Possession	n	$2n+1$	$O(n)$	$3n+m+2$	$O(n+m)$
Vertex Composition	n	$3n$	$O(n)$	$6n$	$O(n)$
Geo-location separation	$k(\ell+1)$	$k^2+2k(\ell+1)$	$O(k^2) = O(n)$	$3k^2+4k(\ell+1)+2$	$O(k^2) = O(n)$

Table 2: Efficiency of proofs of predicates in multiplications dependent on the number of vertices n and of edges m. k is the number of partitions considered for the separation. We consider it to hold that $k \ll \ell \leq n$ and $O(k\ell) = O(n)$. We have parameters of $\ell_v > \ell_n > \ell_e > \ell_r > \ell_m > \ell_E > \ell_V$. $\ell_V = \mathsf{log}_2(|\mathscr{L}_{\mathscr{V},L}| \ \mathsf{ln} \ |\mathscr{L}_{\mathscr{V},L}|) + \mathsf{log}_2((|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|) \ \mathsf{ln} \ (|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|))$ and $\ell_E = 2\mathsf{log}_2((|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|) \ \mathsf{ln} \ (|\mathscr{L}_{\mathscr{V},L}| + \mathscr{V}|))$

Predicate	**Commitments**	**Multiplications**
	#	#
Possession	n	$\ell_v + \ell_e + 2n\ell_r + 3n\ell_V + m\ell_E$
Vertex Composition	n	$3n\ell_r + 3n\ell_V$
Geo-location separation	$k(\ell+1)$	$4k(\ell+1)(\ell_r+\ell_m) + 2k^2\ell_m + k^2\ell_r + 2$

groups. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1296 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 410–424.

[9] CRAMER, R., DAMGÅRD, I., AND SCHOENMAKERS, B. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94* (1994), Y. G. Desmedt, Ed., vol. 839 of *LNCS*, Springer Verlag, pp. 174–187.

[10] DAMGÅRD, I., AND FUJISAKI, E. An integer commitment scheme based on groups with hidden order. `http://eprint.iacr.org/2001`, 2001.

[11] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 186–194.

[12] FUJISAKI, E., AND OKAMOTO, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 16–30.

[13] GROSS, T. 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.

[14] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91* (1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 129–140.

[15] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commu-*

nications of the ACM 21, 2 (Feb. 1978), 120–126.

[16] SCHNORR, C. P. Efficient signature generation for smart cards. *Journal of Cryptology 4*, 3 (1991), 239–252.

List of Acronyms

BFT	Byzantine Fault Tolerance
CCTV	Closed-Circuit Television
CL-GS	Group Signature Scheme with Controllable Linkability
CSS	Computational Secret Sharing
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
EUF-CMA	Existential Unforgeability under adaptively Chosen Message Attacks
FPE	Format Preserving Encryption
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extensions
JCPF	Java Crypto Provider Framework
JSON	JavaScript Object Notation
LA	Linking Authority
OPE	Order Preserving Encryption
PSS	Proactive Secret Sharing
RSA-FDH	RSA Full Domain Hash
RSA-PSS	RSA Probabilistic Signature Scheme
RSS	Redactable Signature Scheme
t-SDH	t Strong Diffie Hellman
VDP	Verifiable Data Processing
XML	eXtensible Markup Language

List of Figures

List of Tables

References

[BG11] Sören Bleikertz and Thomas Groß. A Virtualization Assurance Language for Isolation and Deployment. In *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY'11)*, pages 33–40. IEEE, Jun 2011.

[BGSE11] Sören Bleikertz, Thomas Groß, Matthias Schunter, and Konrad Eriksson. Automated Information Flow Analysis of Virtualized Infrastructures. In *16th European Symposium on Research in Computer Security (ESORICS'11)*. Springer, Sep 2011.

[BVG14] Sören Bleikertz, Carsten Vogel, and Thomas Groß. Cloud Radar: near real-time detection of security failures in dynamic virtualized infrastructures. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC'14)*, pages 26–35. ACM, 2014.

[BVGM15] Sören Bleikertz, Carsten Vogel, Thomas Groß, and Sebastian Mödersheim. Proactive security analysis of changes in virtualized infrastructures. In *Proceedings of the 31th Annual Computer Security Applications Conference (ACSAC'15)*, 2015.

[CCGS10] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In *Advances in Cryptology–CRYPTO 2010*, pages 255–276. Springer, August 2010.

[CG08] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 15th ACM conference on Computer and communications security (CCS 2008)*, pages 345–356. ACM Press, 2008.

[CG12] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. *ACM Transactions on Information and System Security (TISSEC)*, 15(1):4:1–4:30, 2012.

[CL02] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN*, LNCS. Springer, 2002.

[CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology – CRYPTO 1997*, pages 410–424, 1997.

[Gro14] Thomas Groß. Efficient certification and zero-knowledge proofs of knowledge on infrastructure topology graphs. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security (CCSW 2014)*, pages 69–80. ACM, 2014.

[Gro15] Thomas Groß. Signatures and efficient proofs on committed graphs and NP-statements. In *19th International Conference on Financial Cryptography and Data Security (FC 2015)*, pages 293–314, 2015.

[Gro17] Thomas Groß. Geo-location separation of virtualized systems. Technical Report CS-TR, Newcastle University, 2017.

[IBM13] IBM. Specification of the Identity Mixer cryptographic library, v. 2.3.40. Specification, IBM Research, January 2013. http://prime.inf.tu-dresden.de/idemix/.

[ISO06] ISO. *International Standard ISO 3166-1, Codes for the representation of names of countries and their subdivisions–Part 1: Country codes, ISO 3166-1 apha-2*. International Organization on Standardization, Geneva, 2006.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[KR09] Maciej Koutny and Brian Randell. Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques. *Fundamenta Informaticae*, 97(1-2):41–91, 2009.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[MFF+08] Luc Moreau, Juliana Freire, Joe Futrelle, Robert E McGrath, Jim Myers, and Patrick Paulson. The open provenance model: An overview. In *International Provenance and Annotation Workshop*, pages 323–326. Springer, 2008.

[Nef01] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM, 2001.