Radboud Universiteit Nijmegen

Master Thesis

Protecting Personal Data in the Cloud

Using cryptography to enforce confidentiality guarantees under usability constraints

Author:
Vincent Slieker

Supervisors:
Dr. Erik Poll
Prof. Dr. Eric Verheul

August 24, 2015

ABSTRACT

In this thesis, we investigated to what extent current encryption methods can be applied, in practice, to enforce data confidentiality of Dutch personal data processed in the cloud. First, by setting out current the legal implications and the cryptographic possibilities to enforce confidently. In the second part, we design a solution for a specific scenario based on the Dutch license plate authority (RDW). For this scenario, we defined a set of requirements specifying the functionality to perform search, update, and summation queries and provide confidentiality guarantees required under the threat assumption of a compromised cloud provider.

We found that, from a legal point of view, only cloud providers with a "Safe Harbour" agreement or the applicable "EU Model Clause" included, can be contracted to process personal data, according to the Dutch data protection act (WDP) [Per]. However, the WBP states that additional guarantees to ensure data privacy are required. Although these other requirements are not specified they can be assumed to be in line with the EU commission's Article 29 parties vision on cloud computing [soFRotEC12] and include cryptographic solutions. We set out several cryptographic models with different characteristics (CryptDB, C-SDA, GhostDB, and FHE) that ensure confidentiality. We found that CryptDB provided the most query functionality without requiring additional hardware, having an overhead in the order of 25 percent [PRZB11a]. However, none of the evaluated cryptographic models satisfied confidentiality guarantees based on database correlations or query patterns. We, therefore, introduce a new encryption model based on CryptDB providing stronger confidentiality guarantees by limiting pattern creation over time and obfuscating correlations between multiple databases. These guarantees can be obtained at equal theoretical, computational performance as provided by CryptDB, but do require periodical re-encryption causing a temporary limitation to database updates, e.g. around 1 hour for 10GB in our setup. Our model shows that current encryption methods can suffice in enforcing the confidentiality of personal data in a cloud computing scenario while allowing for search and update queries, including the ability to calculate summations and an average.

Keywords: CryptDB, Cloud Computing, Homomorphic Encryption, Data Confidentiality, License Plate Information, Personal Data, Safe Harbour.

CONTENTS

ACRONYMS

General

BSN: Burger Service Number.

FHE: Full(y) homomorphic encryption.

GDPR (European) General Data Protection Regulation.

HElib: (IBM's) Homomorphic Encryption Library.

NIST: (US) National Institute of Standards and Technology.

RDW: RijksDienst Wegverkeer.

WBP: Wet Bescherming Persoonsgegevens.

Databases

DLPR: Database on License Plate Registration

PS-DLPR: Public Subset of the DLPR.

TLPRD: Toy License Plate Registration Database (Table 15)

Encryption Types

DET: Deterministic cryptographic model.

DET* A hybrid DET model based on RND (Section).

HOM: Homomorphic cryptographic model.

RND: Non-deterministic cryptographic model based on pseudo random input.

Requirements & Assumptions

CR-X: Confidentiality Requirement *X*.

FA-X: Functional Assumption *X*.

FR-X: Functional Requirement *X*.

TA-X: Threat Assumption *X*.

Part I

INTRODUCTION

1

BACKGROUND

1.1 PROBLEM DESCRIPTION

Cloud computing is a technology that allows software and hardware for computation and storage to be shared on the internet. In recent years, there has been an increase in the usage of cloud computing by governments and companies [Res13; Boo13]. According to the research and advisory company Gartner, there is a worldwide increase of cloud Infrastructure-as-a-Service of 32.8 percent in 2015 compared to the year before, resulting in a US$16.5 billion market [Moo15]. This increase in the use of cloud services can be explained by several benefits it provides, namely high mobility and flexible scalability, which can lead to better cost control [AFG+10]. However, the increasing shift to cloud-based solutions also raises concerns over the deliberate or accidental disclosure of private data by cloud service providers [Rya11]. These concerns are addressed by policies and legislations, but alone these seem insufficient. The laws in jurisdictions where private data gets collected may not continue to apply to that data post-transfer [Rya11]. Major U.S. Cloud providers Microsoft and Google have admitted they handed over private data of Europeans to U.S. authorities as they were forced by U.S. laws overruling previously made agreements in the EU, and could be forced to do so again [Whic; Whi11].

In recent years, new methods have been developed to complement trust in contractual agreements by encryption models enforcing data confidentially. One of these methods is homomorphic encryption which allows for calculations on encrypted data without the need of intermediate decryption. However, calculations on homomorphically encrypted data can be significantly slower than those on unencrypted data, leading to limited practical applicability [TEHEG12]. It is still unclear to what extend data protection models can be used in the cloud to ensure confidentiality on a practical level.

1.2 PURPOSE OF RESEARCH

The purpose of this thesis is to find the benefits and drawbacks of moving personal data to the cloud, and in what extend these drawbacks can be mitigated by the use of encryption techniques. We will set out a realistic scenario for the Dutch motor vehicle authority "RijksDienst Wegverkeer" (RDW) [Aut14] to investigate a set of problems and limitations that occur when moving personal data to the cloud. The RDW works with data of Dutch citizens and is required to keep personal information within this data private from unauthorized sources. For example, when the RDW transfer their data on license plates to a cloud

provider to use cloud benefits like increased scalability and they outsourcing of maintenance. The RDW can in such a scenario only provide this private information in encrypted format to a cloud provider in a way that confidentiality is assured. Both RDW's private and public license plate data are in the form of a database that the RDW maintains daily. The database has to be accessible in a limited form to other authorized organisations including the Dutch police. It is, therefore, essential that private data stored in the cloud is encrypted but still be accessible to other entities. A solution to this could be the use of an encryption model that allows for queries execution by the cloud provider, without the need for server-side decryption. The problem with this is that most encryption models bring a limitation to the functionality of a database in terms of either cause computational overhead or limiting the supported operations [Mat05]. Whether and in what degree these limitations are of importance to an organisation depends on the desired functionality and performance. This thesis will, therefore, serve as an overview of current data encryption solutions and their ability to cover the needs of the RDW, proving a possible baseline for other organisations.

Part II

RESEARCH METHOD

2

APPROACH

2.1 OVERVIEW

To investigate the extent to which current encryption methods or tools can be applied to enforce data privacy of personal data stored in the cloud, we will use the following approach.

1. **Literature study towards the background of cloud computing and processing of personal data.**
   In the first part of our literature study, we set out the current possibilities of cloud computing and that type of requirements have to be satisfied when choosing to move personal data to the cloud from a legal perspective.

2. **Literature study of methods and models to enforce data security in the cloud.**
   In the second part of our literature study, we set out several encryption methods and models that can provide several data confidentiality guarantees.

3. **Defining functional and confidentiality requirements for a specific cloud computing scenario.**
   In this part of our thesis, we set out the functional and confidentiality requirements of a specific scenario based on the processing of license plate data by the RDW. In our analysis, these requirements will serve as a baseline to consider an encryption model suited to process personal data.

4. **Analyses of the extent to which an encryption model can be applied in the cloud to enforce the confidentiality of personal data.**
   In this analysis, we set out how well encryption models can be deployed to satisfy the previously stated legal, functional and confidentiality requirements. The aim of this analysis is to provide an answer to our main research question.

2.2 RESEARCH QUESTION

Cloud computing has become a hot topic amongst all sort of enterprises as it offers highly scalable computational capabilities and pricing [NTTM15]. The processing of personal data in the cloud leaves enterprises with a stronger trust in cloud providers. This trust stretches from cases of continuous data availability to those of physical storage security. A solution for decreasing some dependencies can be found in the use of encrypted data to enforce confidentiality. Fully homomorphic encryption would be ideal for this as it allows encrypted

data to be manipulated as unencrypted data [TEHEG12]. However, the drawback of full homomorphic encryption is that it is too slow to be a practical business solution [TEHEG12]. In this research, we focus on (partial homomorphic) encryption schemes and to which degree and at what cost they can enforce the confidentiality of data. This research sets out several methods and tools that can enforce the confidentiality of sensitive data by using encryption models that do not require a cloud provider to have access to their private key. We will explain how these methods and tools can be applied in practice, and to what extent they can cover the functional requirements of businesses working with Dutch personal data. This study will be performed to answer the following research question:

To what extent can current encryption methods be applied, in practice, to enforce data confidentiality of Dutch personal data processed in the Cloud?

We will answer our research question at the hand of the following two sub-questions.

- Can encryption methods be used to allow the processing Dutch personal data in the cloud from a legal perspective?
- Is it feasible for the RDW to use encryption to process Dutch personal data in the cloud?

2.3 RESEARCH DELIMITATION

DATA LIMITATIONS There are different forms of data that can be stored or processed in the cloud. The difference between integers and strings, symbols and texts or data containing different levels of entropy can affect both the security guarantees and query types that are required. The focus of this thesis is on the example data and the problem of the RDW described in chapter 1.2 of which detailed use cases are given in Part iv.

LEGAL LIMITATIONS The juridical boundaries regarding the processing of personal (private) data are country dependent. In this thesis, we only examine the juridical boundaries and legal risks for a Dutch organisation (RDW). These boundaries will include Dutch and European laws and regulations regarding both personal data and secure cloud principles. These boundaries exclude other foreign laws and regulations except for the U.S. "Patriot Act", as an illustration to how Dutch/European data privacy laws might be surpassed by foreign entities.

CRYPTOGRAPHIC LIMITATIONS There are different cryptographic schemes that provide a degree of security. In this thesis, we will mainly focus on the confidentiality aspects of cryptographic schemes leaving other aspects as data availability and integrity outside of our scope. We can justify this by the fact that trust in availability always depends on the cloud provider as it can physically remove the database. The integrity of data is assumed to be secured by externally located logging systems and is not included as a requirement for our model.

Part III

METHODOLOGY

3

CLOUD COMPUTING

3.1 DEFINITION

The term 'cloud' has traditionally been used as a metaphor for networks and helps abstract their inherent complexity. Cloud computing is an application of computer networking in which computer services are outsourced. Hardware and software can be made available through the Internet to accommodate a consumer needs. Computing services are not required to be provided locally and can be provided remotely and in mass by services providers (e.g. cloud providers). The consumer of remote services leaves a measure of control to the provider, gaining the flexibility of resources in return [TOM08]. This flexibility of resources can translate to services offering storage and processing power, which can be provided on demand for a cost correlated to the amount and the time a functionality is required [TOM08]. A formal definition of cloud computing has been published by the US National Institute of Standards and Technology (NIST) in 2011 [MG11]. The definition of cloud computing according to NIST is the following:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

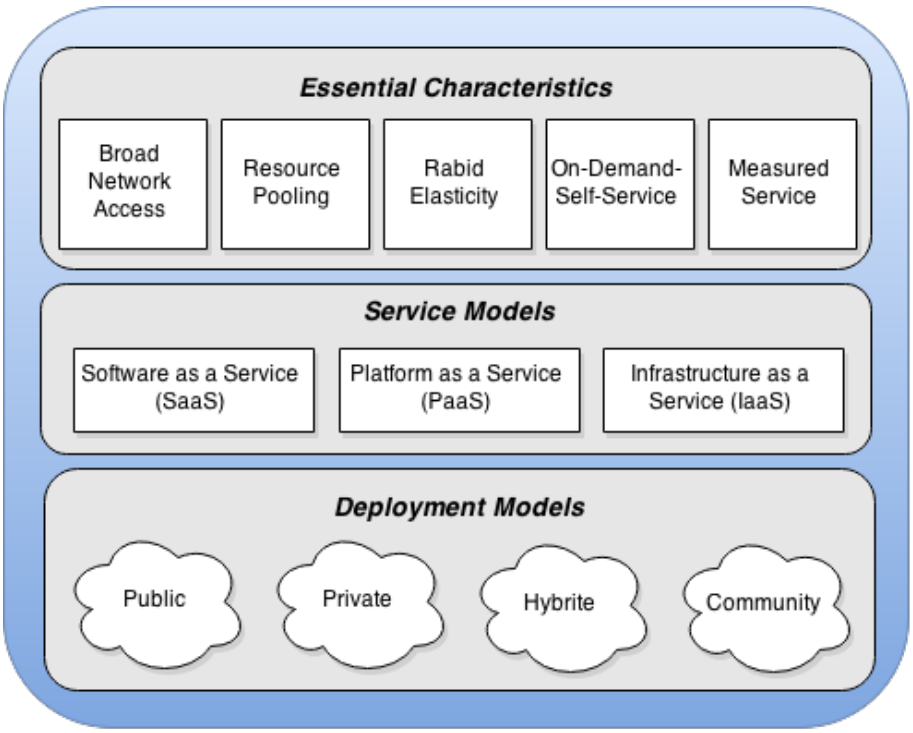

Figure 1: NIST visual model of cloud computing [MG09]

The NIST defines the cloud computational architecture at the hand of five essential characteristics, three service models, and four deployment models, as illustrate in figure 1. Explanation on the defining elements of cloud computing of figure 1 are provided in subsections : 3.1.1 (Characteristics), 3.1.2 (Service Models) and 3.1.3 (Deployment Models).

3.1.1 *Characteristics*

The five essential characteristics defined by NIST are typical of current cloud computing solutions when compared to traditional computational models and can be elaborated as followed:

ON-DEMAND-SELF-SERVICE : Consumer of cloud computing services can upgrade or downgrade services and acquire new services utilizing an automated process initiated by the consumer without human interaction from the cloud provider. Examples of this include a consumer acquiring additional storage space, processing power or virtual machines when needed, by simple means provided in the services web interface.

BROAD NETWORK ACCESS : Services are accessible over a then network and can be accessed through a standard mechanism provided by client platforms like laptops, smartphones, and tablets.

RESOURCE POOLING : Cloud services are efficiently using resources in the sense that computing resources of the cloud are pooled to serve multiple consumers in a multi-tenant

model. This leads to a scenario in which customers have no knowledge of the physical location of the provided resources (e.g. data-center or countries) as this may vary and is perceived from a higher level of abstraction.

RAPID ELASTICITY : The capacity of services can be scale quickly to the need of consumer's and appears to be unlimited from their perspective.

MEASURED SERVICE : Cloud systems automatically control and optimize resource as their demand is highly dynamic. Cloud systems monitor these resources and can provide insight into the use of a by the consumer utilized service, as this often affects the price of the service [KV10].

3.1.2 *Service Models*

Cloud service models can be divided into three basic models based on the their level of abstraction. This division is referred to as the "SPI model" named after the first letter of each model and can be explained by the following description:

SOFTWARE AS A SERVICE : The provided service by the cloud provider is software focused and in which the consumer is the end user. The software runs on the clouds infrastructure and is provided to the consumer as an application. Examples include email clients like Google's Gmail, Cisco's WebEx or Salesforce's CRM-systems [Wid].

PLATFORM AS SERVICE : The cloud provider enables a consumer to use the clouds infrastructure capabilities like storage, processing or network by providing means to the consumer to run his or her own software. This software can be developed on a by the cloud provided architecture. An example of these services includes Microsoft's Azure, Google's App Engine and Amazon's EC2 [NTTM15].

INFRASTRUCTURE AS A SERVICE : The cloud provides the consumer with controlled access to fundamental computing resources and enables the consumer to deploy arbitrary software, which can include operating systems. IAAS can be used to (temporally) extend an existing data center to cope with high demand. Examples of these services include Amazon's Elastic Compute Cloud (AWS), Microsoft's Azure and Google's Compute Engine (GCE) [NTTM15].

3.1.3 *Deployment Models*

The four different basic deployment models of cloud solutions are divided on the basis of consumer access.

PUBLIC CLOUD : The cloud infrastructure is controlled by a cloud provider that offers its service to the general public.

PRIVATE CLOUD : The cloud infrastructure is controlled by a cloud provider that offers its service to a single organization. That organization might also be the owner of

the cloud provider, resulting in less trust depended security measures then common in public cloud construction. The consumers of the cloud services in this scenario are typically comprised of independent departments within that organizations like different business units.

COMMUNITY CLOUD : The cloud infrastructure is shared by a specific community of consumers that have collaborated on the basis of shared concerns.

HYBRID CLOUD : Is a cloud infrastructure that is composed of multiple cloud infrastructures that remain indented entities, but that are bound because they depend on shared technology that enables portability.

3.2 SECURITY

3.2.1 *Threats*

An organization moving to the use of cloud computing might be exposed to risks in a variety of areas like data privacy, availability, service provisioning, malicious attacks, and regulatory compliance [KV10]. The Cloud Security Alliance (CSA) is trying to raise awareness for these threats and publicised the "The Notorious Nine: Cloud Computing Top Threats in 2013" report in order to reflects the main concerns regarding cloud security [G+13].

The main threats identified by CSA in 2013 where the following nine:

1. **Data Breaches**: The risk of organization's (sensitive) data falling into malicious hands. An example of this includes the notorious Apple iCloud hack of 2014 in which personal nude pictures of celebrity leaked on the internet [Wor].

2. **Data Loss**: The loss of an organisation's data is many scenario's undesirable and can, for example, occur due to attacks on the cloud by hackers or mistakes a clouds provider.

3. **Account Hijacking**: Hijacking of accounts is a known threat to organizations as phishing attacks and exploitation of software vulnerabilities are still common as set out in the 2015 report "Why phishing still works: user strategies for combating phishing attacks" [AAC15].

4. **Insecure APIs**: Cloud providers offer service that allow consumers to interact with certain basic APIs. It is, therefore, clear that the security of these APIs should not contain any exploitable weaknesses, as this would compromise a cloud's security.

5. **Denial of Service**: Attacks based on the Denial Of Services (DOS) have proven to be viable against many types of infrastructure including those of cloud providers and their services [NZMK15]. It is however worth mentioning that though cloud services can be vulnerable to (distributed) denial of services attacks, they also provide a degree of protection against DOS in their elastic nature that can be exploited by cloud providers to provide better consumer protecting [NZMK15; JVJ12].

6. **Malicious Insiders** Can be described by the hand of several definitions. CSA uses the CERN definition of an malicious insider which is defined as : *"A malicious insider threat to an organization is a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems."* An example of malicious insider is the American whistle blower Edward Snowden who released sensitive documents regarding the NSA's surveillance practices [GMP13]

7. **Abuse and Nefarious Use** Cloud providers make it possible for consumers to rent a large amount of computing power for a certain time without the need for hardware investments [KV10]. Cloud services are therefore also interesting for malicious consumers who might want to misuse them for illegal activities. Detection of abuse of services if therefore of importance from a providers point of view.

8. **Insufficient Due Diligence** An organization might introduce new security risks when moving to the cloud due to insufficient understanding of the deployed environment and services.

9. **Shared Technology Issues** Cloud providers deliver scalable services and in terms of infrastructure, platform and application. It is important form a security perspective that this scalability does not come at the cost of security as strong isolation properties need to be assured.

From a consumer point of view we can divided these threats in three categories of threats. First, threats caused by insufficiency knowledge or preparations like, "Insufficient Due Diligence", "Account Hijacking" and "Data Loss" which can be mitigate by adequate preparation. Second, threats that are primarily outside the control or scope of the consumer like "Insecure APIs", "Denial of Service", Shared Technology Issues" and "Abuse and Nefarious Use". Security fields outside the direct control of a consumer can however still be mitigated by means like contractual agreements and certifications that we will discuss in section 3.4 Third, threats that a consumer can address like "Data Breaches" and "Malicious Insiders" by enforcing consumer initiated encryption securing a degree of data confidentially at the cloud as further discussed in section 5.

3.3 PRIVACY REGULATIONS

3.3.1 *Dutch Data Protection Act*

The privacy of personal data is protected in the Netherlands under the Dutch data protection act "Wet Bescherming Persoonsgegevens" (WBP) [Per00]. The WBP is in effect since 2001 and gets uphold by a specialized government agency "College Bescherming Persoonsgegevens". The WBP is based on the European Data Protection Directive 95/46/EC [Dir95]. The WBP defines what is considered personal data by Dutch law and is used in our research method to classify sensitive data in section 6.3.1. The WBP also states the following main point in regard to the privacy of personal data:

- A Dutch citizen has the right to exercise a degree of control over his or her personal data (Chapter 6).
  - A citizen is at any time allowed (at a charge, Art 17) to request insight in his or her personal data stored or processes by an organization.
  - A citizen is allowed to request a correction of his or her personal data if this is not correct (Art 35)
  - A citizen is allowed to object formally to the processing of his or her personal data. An example of this includes the processing of personal data for marking processes (Art 8).
- Organizations processing personal data has obligations (Chapter 2) which lead to legal fines when not applied correctly (Art 75)
  - An organization may only processes or store personal data proportionate too and in compliance with well-defined legitimate purposes (Art 7, 9, 11). These purposes have to be registered at the appropriate authority for approval.
  - An organization may only process or store personal data of citizens that have given explicit consent to do so. (Art 8)
  - An organization must always notify a citizen about the purpose for which his or her personal data is collected and will be used. Unless an exception for this is made for this purpose, legitimized for example by the need to protect a legal investigation or due to the trivial nature of the intended purpose (Art 9).
  - An organization has to make sure that personal data is sufficiently protected according to security guarantees set out in the WBP that are attribute depended. Attributes that require stronger security guarantees opposed to standard information like name and age are special attributes like race, health, and religion (Art 17 - 22).
  - An organization may be charged for violation of the WBP (Art 75). Since the 2012 and 2014 [tH] amended of the WBP, violators of the WBP can receive fines from the third categories (maximum 20,250 Euro [Nag]) or fourth (maximum 810,000 Euro and 6 months jail [Nag]) depending on whether a certain violation occurred on intent.

3.3.2 *EU/U.S. Legislation*

The United States and the EU both have their take on the protection of personal and private data of their and foreign citizens [Exp]. In Europe privacy is considered a fundamental right that can only be limited in the case of absolute necessity [Par10], where in the U.S. privacy guarantees are left to neoliberal norms of a free market, as long as this forms no considerable national risk [Sha00]. This fundamental difference in view between the EU and the U.S. translates to different privacy laws and regulations. The European Union plans to have one formal framework (GDPR) for data privacy legislation that will apply to all EU countries, sectors, and people proving stronger confidentiality guarantees that general U.S. legislation. The U.S doesn't have a general privacy law or framework applicable

to all industries and formalizes main privacy laws specific to certain industries and sectors. Examples of industries specific privacy laws in the U.S. include the Electronic Communications Privacy Act (ECPA), Health Insurance Portability and Accountability Act (HIPAA) and the Children's Online Privacy Protection Act (COPPA) [ERB03]. In the US companies are allowed by default to collect and store personal data in (intrusive) ways that are forbidden in EU. An example of this is that U.S. companies can demand personal information from its customers or employees without justification, and process it without registration of purpose [Sul]. Under EU law, personal data can only be collected under strict conditions aiming for limited infringement of personal data. Limitations on the collection of personal data include that it may only be stored for a specific amount of time and in service of a well-defined and legitimate purpose. Guidelines on security guarantees are set out in EU Directive 95/46/EC [Dir95] on the protection of personal data, which is used in the Dutch WBP. Directive 95/46/EC also forms the basis for its future successor the General Data Protection Regulation (GDPR) [lg], which we will discuss in 3.3.3. The U.S. and the EU currently have an agreement to bridge their differences initiated by U.S. Department of Commerce and in consultation with the European Commission called the "Safe Harbor" principles [GLM01]. The Safe Harbor Principles are a framework which U.S companies can use for a certification that meets EU privacy requirements, allowing for better and more efficient cooperation between the EU and U.S. as set out further in 3.4.3.

3.3.3 *General Data Protection Regulation*

The European General Data Protection Regulation (GDPR) will be the successor of Directive 95/46/EC of 1995 and is currently being developed by European Commission [Com12]. The GDPR aims to unify data protection within the European Union in one law, incorporating directive 95/46/EC [Dir95] and covering previously unforeseen complications in the erupted fields of social networks and cloud computing [Tri]. Currently, only 1 in 100 cloud providers meet the security requirements proposed GDPR regulations as they provide insufficient privacy guarantees [Col]. One of the reasons for this is that currently 98.8% of cloud providers do not provide data encryption with client managed key's, leaving clients without adequate data protection control (Art. 29) [Col]. Solution for client controlled encryption are discussed in section 5. Based on the GDPR draft of 2012 [Alb12] and 2014 adjustments [Par] the GDPR will contain the following main changes from Directive 95/46/EC relevant to the use cases of section 6.4:

TERRITORIAL SCOPE ART. 3 :

- The Territorial scope get increased as the GDPR regulation applies to organizations based outside the EU if they process information relating to EU residents. This includes for example that the GDPR regulations applies direly in the cases of cloud computing when an EU-based company is transferring personal data (e.g. names, email addresses, IP-address) of EU residents to a cloud service outside the EU (e.g. U.S.). This results in that the GDPR effects the Safe Harbor agreement as compliance with EU regulation is no longer an opt-in process as described at 3.4.3.

- A One-Stop-Shop mechanism is also to be created to assure uniformity of supervisory authorities across the EU Member States. This centralization will be done by having the GDPR rules consistent across the EU and having only a single Data Protection Authority (DPA) responsible for each company [Fra].

CONSENT ART. 7 :
Art. 7 states that more (explicit) control is to be placed at the information providing party. Main points include that valid consent requires to be given explicitly for both the collection of data as its uses. A data controller has to be able to prove that (opt-in) consent was given by the data provider and is required to allow for an efficient and user-friendly withdrawn of consent by that provider.

RIGHT TO ERASURE ART. 12 :
The "Right to be forgotten" ruling c-131/12 [Coma] is replaced by the more limited right to erasure. A data controller now has to take all reasonable steps to have an individual's data erased, including those maintained by third parties, upon a legitimate request be an individual. The right to erasure focuses on these "reasonable steps" as the right to be forgotten deemed too broad to be effectively upheld in complex multiparty scenario's as those regarding cloud computing [Ram].

DATA PORTABILITY ART. 15 :
Consumers are required to have the ability to request a copy of their personal data that is being processed by their service provider, in a for them usable and electronic format.

NOTIFICATION ART. 31 :
Notification on security breaches gets stricter for cloud providers. Service providers of cloud solutions to EU residents are required to notify the relevant supervisory authority in the EU within 24 hours in case of a data breach.

DATA PROTECTION OFFICER ART. 35 :
Large companies (250 employees or more for cloud computing solutions [Ram]) are to appoint independent Data Protection Officers ("DPOs"). A DPO is responsible for registering all of the processing involving personal data and for ensuring compliance with the GDPR and other regulations across all 28 EU member states. DPOs have to have a broad understanding of not only legal legislation but also IT processes and data security aspects, making their requirements more specialized in data privacy than those of a Compliance Officers [DA].

SANCTIONS ART. 79 :
Sanctions get tougher. Non-compliance with the GDPR will have significant financial consequences. Fines can get up to a maximum of either 100 million euro or 5 percent of the worldwide turnover of an organisation, whichever is higher. The maximum amount of money demanded by these fines is a vast increase compared to the maximum penalties individual EU member states currently uphold. France, for example, allows for a fine of maximum 300.000 euros for imposing unilaterally new terms of service on users [Rad]. In comparison, a multinational as Google produces a revenue

of multiple times than in less than 10 minutes [Rel]. These higher fines are therefore aimed at drawing stronger attention and careful investigation of the GDPR by wealthy multinationals.

3.4 CERTIFICATIONS

In this section, we will describe the most common certification found at major cloud services and what type of guarantees they intend to provide. An analysis of these certifications will be used to form a recommendation for a cloud provider in our solution design and discussed in section 7.1. The three different type of Certifications used by major cloud providers regarding personal data that we set out in the section are given in a short overview of table 1.

Type of Certification	**Regional scope**	**Sufficient protection for Dutch personal data processing outside EU?**
ISO/IEC (Section3.4.1)	International Standards	No, on their own ISO/IEC certifications are not sufficient to comply with EU directive 95/46/EC. ISO/IEC are however able to assist in complying with the EU directive 95/46/EC [DPUotDGfJS].
Safe Harbor (Section 3.4.3)	U.S. initiated agreement with EU	Yes, for most US based companies commitment to the Safe Harbor agreement is sufficient to comply with EU directive 95/46/EC. The concerned personal data should however be covered by the Safe Harbor commitments and concerns a sector under the supervision of the Federal Trade Commission (FTC), or the Department of Transport. Agreement in Safe Harbor fall under the jurisdiction of the FTC [DPUotDGfJS].
EU Model Clauses (Section 3.4.2)	EU initiated privacy guidelines	Yes, use of the standard contract clauses provides the necessary safeguards to comply with EU directive 95/46/EC and can be seen as a EU initiated alternative for Safe Harbor. Two of the main differences are that, agreement to EU Model Clauses falls under the jurisdiction of EU member states and that they are not only available to U.S or EU companies [DPUotDGfJS].

Table 1: Different types of certifications regarding the privacy of personal data obtainable by cloud providers.

In table 1 we can see that either the Safe Harbor agreement or an EU Model Clause contract is required to comply with EU Directive 95/46/EC. Dutch Personal Data falls under the WBP, which is based on 95/46/EC and, therefore, upholds these same contractual requirements as section explained in section 3.3.1. Further details on these types of certifications are given in the following three subsections.

3.4.1 *ISO/IEC*

International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) work together to provide procedures and international standards. Standards released by ISO and IEC often start with ISO/IEC followed by six numbers and which can be obtained by organizations as certification, providing provable compliance with specific conducts required for good business practices. The following certification of ISO/IEC is of interest to cloud providers or IT organizations in general.

ISO/IEC 27001

ISO/IEC 27001 [NEN13] is a general information security standard published by ISO

and IEC. ISO/IEC 27001 is a standard describing procedural how information security standards from ISO/IEC 17799 are required to be implemented within an organization. In the Netherlands ISO/IEC 17799 is made into a national norm (NEderlandse Norm) NEN standard NEN-ISO/IEC 27001:2013 which has been made obligated for Dutch's governmental organizations by the College Standaardisatie [vV].

ISO/IEC 27002
: ISO/IEC 27002 [ISO13] is a general standard published by ISO and IEC providing practical implementation guidelines for information security management in an organization. Where ISO/IEC 27001 is more oriented on management ISO/IEC 27002 is more focused on establishing concrete controls needed for risk treatment. In the Netherlands ISO/IEC 27002 is made into the NEN standard NEN-ISO/IEC 27002:2013 and has been made obligated for Dutch's governmental organizations by the College Standaardisatie.

ISO/IEC 27017
: ISO/IEC 27017 [ISO] is a standard currently in development by ISO and IEC with the intended purpose of supplementing the general orientated ISO/IEC 27002 with cloud-specific information security controls guidance. ISO/IEC 27017 is also being developed with ISO/IEC 27018 in mind, covering a broader information security perspective than privacy alone.

ISO/IEC 27018
: ISO/IEC 27018 "Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors" [ISO14] is an information security standard published by the International Organization for Standardization (ISO) aimed at ensuring adequate privacy controls at cloud providers. One of the main points in ISO/IEC 27018 different from ISO/IEC 27001/27002 is that it addresses personally identifiable information. Personally identifiable information is an important field of Dutch cloud information processing, because of the Dutch data protection act as described in section 3.3.1.

3.4.2 *EU Model Clauses*

In section 3.3 we set out the difference in regulation between the U.S. and EU. EU model clauses are the result of these differences aiming to provided foreign organization with the guidelines to certify themselves according EU privacy regulations. The EU model clauses are issued by the Article 29 Working Party (Art. 29 WP) commission in agreement with the European Parliament and are based on data protection directive 95/46/EC.

The aim of these clauses is: *" To provide adequate safeguards with respect to the protection of the privacy and fundamental rights and freedoms of individuals and as regards the exercise of the corresponding rights."* [Comb]. and are in line with the Art. 29 WPs published an opinion on cloud computing [soFRotEC12]. Art. 29 WPs published option on cloud computing was a working party analysis of relevant issues for cloud computing service providers operating in the EU providing a data protection framework. Regarding data confidentiality, the following relevant statements are provided:

1. Encryption does not render personal data irreversibly anonymous (Directive 95/46/EC Recital 26).
2. Encryption of personal data should always be used when data is in transit and when available to data at rest (Article 8 of Directive 95/46/EC)
3. A cloud client may not rely on an encryption solution offered by the cloud provider. Encryption of personal data before sending it to the cloud is suggested.
4. "If a client plans to not only store, but also further process personal data in the cloud (e.g., searching databases for records), he must bear in mind that encryption cannot be maintained during processing of the data (except of very specific computations)." [soFRotEC12] . Note that the limitation described as "very" specific computations is debatable as complex schemes like CryptDB, GhostDB and C-SDA offer a variety of functionality under encryption as described in chapter 5

There are currently two sets of standard contractual clauses:

1. For transfers from data controllers to data controllers established outside the European Onion and European Economic Area.
2. For the transfer to processors established outside the European Onion and European Economic Area.

These clauses provided foreign cloud providers with sufficient privacy safeguards to qualify for the handling of personal data if all other use-case and nation-specific regulations are covered [dG]. Nation specific regulations might currently differ between EU nations, but will be uniform within next several years as discussed in section 3.3.3

3.4.3 *The Safe Harbor Agreement*

In order to bridge the difference in approach and in order to provide streamlined means for U.S. organizations to comply with Data Protection Directive 95/46/EC, the U.S. Department of Commerce developed the "Safe Harbor" framework in consultation with the European Commission. The Safe Harbor consists of seven principles with which U.S. companies must comply in order achieve certification of compliance with Directive 95/46/EC. A summation of rules based on these seven principles derived from the official decision in 2000/520/EC [oC] can be given as followed:

1. **Notice**: An organization has to inform each individual from which it collects data about both the fact it collects data of that individual and for which intend that collection takes place. This has to be done in a clear and for the individual understandable manner.
2. **Choice**: An organization has to offer each individual an opportunity to opt-out on personal data transfers to third parties or for purposes that are not in line with the original consent. This opportunity has to be given in advance and provided in a for the individual comprehensive way to exercise choice.

3. **Onward Transfer**: An organization can only disclose personal information to a third party that provides at least the same level of privacy protection and in combination with the Notice and Choice principle.

4. **Security**: An organization must take reasonable precautions to protect personal data from misuse like unauthorized access, disclosure, and destruction.

5. **Data Integrity**: An organization may only collected personal relevant and proportional to the purposes for which it is to be used.

6. **Access**: An individual must be able to achieve access to all personal data about him or her that an organization holds. An individual should also be able to have that information corrected, amended or deleted if that information is inaccurate. An organization can make an exception to this if an individual's request is disproportionate to the risks to that individual's privacy.

7. **Enforcement**: An organization is required to have an effective privacy protection mechanism in place that can be verified, allowing for the effective enforcement of the Safe Harbor rules.

3.5 INTRUSIVE FOREIGN LAWS

Last decade data storage and services moved from private servers to increasingly more international cloud services [AFG+10]. This move resulted in that data now moves through different servers, companies and countries to provide the best service at the lowest cost. In 2012 more 90% of Europe's cloud market was controlled by U.S. based companies and their local European divisions [Sav]. The involvement of foreign country's handling data considered private by European laws [Sch94b] lead to conflicts of interest when the involved country's laws do not legitimize or overrule privacy laws in the EU. In this section, we will focus in on the Patriot Act that provides the U.S. with legal means to overrule European laws.

3.5.1 *Patriot Act*

The USA PATRIOT Act (aka. Patriot Act) [Act01] is an acronym for *"Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism Act"* and is an U.S. law that was implemented by President George W. Bush in September 2001 following the 9/11 terrorist attacks, which had happened two days before. The 9/11 attacks consisted of several hijacks of passenger's airplanes by terrorist of the Islamic terrorist group of Al-Qaeda [Gun02]. The hijackers flew the captured planes including all passengers and personnel into several major landmarks in the U.S. that where located New York City and the Washington area. This attack lead to the destruction of the New York World Trade Center (WTC) and the death of nearly three thousand people including those of all the hijackers [ftAiGS14]. This terrorist attack moved the U.S. to adopt laws [Act01] extending the legal boundaries of law enforcement agencies to provide better security and prevent future attacks [oJ01]. The Patriot Act is known for the controversies it received by

U.S. and foreign media, including those of the Netherlands, criticizing the intrusive way the Patriot Act allows U.S. agencies to access personal data of both U.S. and Non-U.S. citizens [Coo05] [AG06] [VHAvEK12]. In section 215 of the Patriot Act titled : *"Access to records and other items under the foreign intelligence surveillance act"* stands the following insertion to the Foreign Intelligence Surveillance Act of 197:

> *"The Director of the Federal Bureau of Investigation or a designee of the Director (whose rank shall be no lower than Assistant Special Agent in Charge) may make an application for an order requiring the production of any tangible things (including books, records, papers, documents, and other items) for an investigation to protect against international terrorism or clandestine intelligence activities, provided that such investigation of a United States person is not conducted solely upon the basis of activities protected by the first amendment to the Constitution."* [Act01]

This insertion = allows for the request of "tangible things", which can be interpreted as anything that the Federal Bureau of Investigation (FBI) finds tangible. This vague definition room for collisions with other foreign laws based on the interpretation of the FBI. An area in which such collision can occur is cloud computing where European citizen have access to major cloud providers such Apple, Amazon, Google and Microsoft that are based in the U.S. and fall under the Patriot Act, but are also operational in EU law [Whic]. In such a scenario the Patriot Act can extend beyond the borders of the U.S. applying to EU companies which have a base in the U.S, an U.S. parent company, use the services of a U.S. subsidiary for data processing or use any third party to store or process data in the U.S. [Bod12]. The Patriot Act can, therefore, be applied on companies containing data of European citizens and force the disclosure of that data, even if disclosure would lead to the violations of EU regulations [Sav]. Cases of these types of disclosure have also been admitted by major corporations like Google and Microsoft that have confirmed that they handed over EU-stored data to U.S. law enforcement without European approval, as they were bound by request that were enforced upon them by the Patriot Act [Whib]. A legal agreement between the U.S. and Europe regarding data privacy under the "Safe Harbor" framework has not been able to prevent these data disclosures, illustrating the difficulties in protecting private data based on formal agreements [Sav].

4

ENCRYPTION

"If you would keep your secret from an enemy, tell it not to a friend" [Fra87].

Is a citation from the book "Poor Richard's Almanack" [Fra87] written by Benjamin Franklin, one of the founding fathers of the United States. In modern times, it still holds true that secrecy (confidentiality) can best be preserved by preventing the need to rely on other (trusted) parties. In section 3.5 we already set out the complexity of regional laws in respect to international data traffic and some of their collisions. An alternative to agreements and trust comes from mathematics where cryptographic can provide scientifically based guarantees regarding secrecy [Ma003]. It this chapter we describe encryption methods found in literature that can be applied to enforce secrecy while preserving homomorphic properties. Homomorphic properties allow ciphertext to be used in combination with certain operations like addition, multiplication or inequality checks as described in section 4.1. This theoretical baseline about homomorphic encryption will be used in the description of modern encryption schemes discussed in chapter 5.

4.1 HOMOMORPHIC ENCRYPTION

Homomorphic encryption is a form of encryption that allows for computations to be executed without the knowledge of the secret key. A specific type of computation can be performed on a cipher text and generate an encrypted result, which when decrypted, equals the result of that operation performed on the unencrypted data. An advantage of homomorphic encryption is that (intermediate) decryption is not required when performing specific calculations. Calculations on encrypted data can then be done by a party that is not trusted with the secret key used in the encryption scheme. A formal definition of homomorphic encryption is given by Sen as found in definition 4.1.

Definition 4.1. Homomorphic Cryptosystem as found in [Sen13]
Let the message space (M, o) be a finite (semi-)group, and let σ be the security parameter. A homomorphic public-key encryption scheme (or homomorphic cryptosystem) on M is a quadruple (K, E, D, A) of probabilistic, expected polynomial time algorithms, satisfying the following functionalities:

- **Key Generation:** On input 1^{σ} the algorithm K outputs an encryption/decryption key pair $(k_e, k_d) \equiv k \in K$ where K denotes the key space.

- **Encryption:** On inputs $1^\sigma, k_e$ and an element $m \in M$ the encryption algorithm E outputs a cipher text $c \in C$, where C denotes the ciphertext space.
- **Decryption:** The decryption algorithm D is deterministic. On inputs 1^σ, k, and an element $c \in C$ it outputs an element in the message space M so that for all $m \in M$ it holds that : if c = E($1^\sigma, k_e$, m) then Prob [D (1^σ ,k,c) $\neq$ m] is negligible, i.e., it holds that [D (1^σ ,k,c) $\neq$ m] $\leq 2^{-\sigma}$.
  - *Remark* : Adjusted from [Sen13] replacing "c = E($11^\sigma, k_e$, m)" with "c = E($1^\sigma, k_e$, m)".
- **Homomorphic Property:** A is an algorithm that on inputs $1^\sigma, k_e$ and elements $c_1, c_2 \in C$ outputs an element $c_3 \in C$ so that for all $m_1, m_2 \in M$ it holds that: if $m_3 \equiv m_2 \circ m_1$ and c_1 = E(1^σ, k_e, m_1) and c_2 = E(1^σ, k_e, m_2) then Prob [D (A (1^σ ,k_e, c_1, c_2))] $\neq m_3$] is negligible.

Homomorphic cryptosystems can generally be divided into the two categories partial homomorphic and fully homomorphic based on their cryptographic limitations. A partial homomorphic cryptosystems is a cryptosystems which has its homomorphic property as giving in definition 4.1 limited to either addition or multiplication. $m_3 \equiv m_2 \circ m_1$ holds in that case only for either $m_3 \equiv m_2 \times m_1$ or $m_3 \equiv m_2 + m_1$. A full homomorphic cryptosystems allows for both addition and multiplication for example allowing to have $m_3 \equiv m_2 \times m_1 + m_1$ still satisfying the homomorphic property. We can define these two types of homomorphic cryptosystems as an extension on definition 4.1.

Definition 4.2. Partial Homomorphic Cryptosystem Is a homomorphic cryptosystem for which the following rule replaces the given homomorphic property rule in definition 4.1. A is an algorithm that on inputs $1^\sigma, k_e$, and elements $c_1, c_2 \in C$, outputs an element $c_3 \in C$ so that for all $m_1, m_2 \in M$ it holds: if $m_3 \equiv m_2 \times m_1 \oplus m_2 + m_1$ and c_1 = E(1^σ, k_e, m_1) and c_2 = E(1^σ, k_e, m_2) then Prob [D (A (1^σ ,k_e, c_1, c_2))] $\neq m_3$] is negligible.

- **Additive** Partial homomorphic cryptosystem that allow for the $m_3 \equiv m_2 + m_1$ to hold true, but not $m_3 \equiv m_2 \times m_1$, are referred to as additive homomorphic cryptosystem.
- **Multiplicative** Partial homomorphic cryptosystem that allow for $m_3 \equiv m_2 \times m_1$ to hold true, but not $m_3 \equiv m_2 + m_1$, are referred to as multiplicative homomorphic cryptosystem.

Definition 4.3. Fully Homomorphic Cryptosystem Is a homomorphic cryptosystem for which the following rule replaces the given homomorphic property rule in definition 4.1. A is an algorithm that on inputs $1^\sigma, k_e$, and elements $c_1, c_2, \ldots, c_i \in C$, outputs an element $c_{i+1} \in C$ so that for all $m_1, m_2, \ldots, m_i \in M$ it holds: if $m_{i+1} \equiv m_1 \circ m_2 \circ \cdots \circ m_i$ and $\int_1^i c_i$ = E(1^σ, k_e, m_i) and A_{rec} and D_{rec} are the recursive versions of respectively A and D with input (1^σ ,k_e, m_x, m_y) and (1^σ ,k_e, c_x, c_y) taking all messages with and listed between x and y. Then the Prob [D_{rec} (A_{rec} ($1^\sigma, k_e, c_1, c_i$))] $\neq m_{i+1}$] is negligible.

- *Remark*: Some definitions of fully homomorphic cryptosystem do not require the homomorphic property of definition 4.1 to hold for multiple additions and multiplications as we will discuss in section 4.2. Homomorphic cryptosystem that allow for

multiple additions or multiplications with only 1 or limited other operations can be referred to as weak, making the distinction between strong and weak fully homomorphic encryption [Bar11]. Since in 2009 Gentry invented a strong fully homomorphic cryptosystem, allowing for unrestricted use of multiplication and addition causing fully homomorphic has become synonym with strong fully homomorphic and is referred to as full homomorphic [Gen09]. We choose to exclude the use of strongly and weakly as addition for the term fully homomorphic in the rest of this thesis as fully homomorphic always refers to strongly fully homomorphic (full homomorphic) if not denoted otherwise.

4.1.1 *Additive Homomorphic*

Partial homomorphic cryptosystem schemes that only allow for additions on their ciphertext include the probabilistic asymmetric algorithms Pailler [Pai99] and Goldwasser-Micalli [GM84]. The use of additive homomorphic schemes dates back to the first century where its properties were used by the (insecure) substitution scheme Caesar Cipher as mentioned by the Roman historian Gaius Suetonius Tranquillus [Dav66]. Additive property can be useful in different types of scenario's, so can Pailer and Goldwasser-Micalli be used for voting schemes that allow for the summation of votes without decrypting them, preserving the anonymity of the vote. The Caesar Cipher, though not very secure by current security standards allows for the concatenation of words in encrypted messages without the need of decrypting them. This properly also holds for later substitution schemes as the VigenÃÍre cipher if the key's length is known. Two ciphertexts can be concatenated if the first ciphertexts length equals that or a multiple of the key's length. More detailed examples of substitution properties can be found in "Communication theory of secrecy systems" [Sha49].

4.1.2 *Multiplicative*

Encryption schemes that only allow for multiplication on their cypher texts include (unpadded) RSA [RSA78] and ElGamal [ElG85]. Many multiplicative homomorphic schemes rely on the use of modulus calculations that make them efficient for practical security purposes as key lengths can remain relatively small compared to additive homomorphic schemes. Comparable to the additive E-voting schemes their exist multiplicative E-voting models that are built on the principle of factorization instead of summation and including, for example, ElGamal-based encryption schemes [PAB+05].

To illustrated the difference between additive and multiplicative homomorphic property we set out the simple example of addition and multiplication with unpadded RSA. This will show the partial homomorphic property of RSA, as it allows for multiplication but not addition to hold true for equality equations derived from definition 4.2. From definition 4.2 we can derive that an encryption scheme is additive partial homomorphic if it holds for message "x" and encryption operation "Enc(x)" that the following equations are true: If

$x_3 \equiv x_2 \times x_1$ than $c_3 \equiv \text{Enc}(x_2 + x_1) \equiv \text{Enc}(x_2)$ + $\text{Enc}(x_1) \equiv \text{Enc}(\ x_3)$ and $\text{Enc}(x_2 + x_1) \neq \text{Enc}(x_3)$.

$$Enc(x) \equiv x^e \mod m \equiv c \tag{1}$$

In formula 1 we have the encryption scheme for a basic RSA with plain message "x", modulus "m", public key exponent "e", and encryption function Enc(x) producing a cipher text "c".

$$Enc(x_1) * Enc(x_2) \equiv {x_1}^e * {x_2}^e \mod m \equiv (x_1 * x_2)^e \mod m \equiv Enc(x_1 * x_2) \tag{2}$$

In equation 2 we can see that if $x_3 \equiv x_2 \times x_1$ is true it holds that $\text{Enc}(x_2 + x_1) \equiv \text{Enc}(\ x_3)$. This means that the homomorphic property of RSA holds under multiplication making it (partial) homomorphic.

$$Enc(x_1) + Enc(x_2) \equiv {x_1}^e + {x_2}^e \mod m \neq (x_1 * x_2)^e \mod m \tag{3}$$

The homomorphic property of RSA does not hold under addition since both power and modulo are a multiplication based operation that do not preserve addition as is given by the proof in equation 3. This proves that $\text{Enc}(x_2 * x_1) \neq \text{Enc}(\ x_3)$, showing that his RSA encryption is multiplicative partial homomorphic and not additive or full homomorphic. Note that some implementations of RSA use padding breaking the multiplicative property making them non-homomorphic [CNS99].

4.2 FULL HOMOMORPHIC ENCRYPTION

The idea of Full Homomorphic Encryption (FHE) and its benefits were already recognized during the development of RSA and published in 1978 in a paper on the importance of homomorphismâĂŹs in which FHE was referred to as privacy homomorphism. [RAD78] [RSA78]. For three decades, it was unknown whether a full homomorphic encryption scheme was either theoretical or practically possible. There was no encryption scheme known that allowed for unlimited addition and multiplication operation to be used on encrypted data while preserving the homomorphic property. Only non-homomorphic encryption schemes and partial homomorphic schemes like unpadded RSA and where in use, allowing either multiplication or addition operations on their ciphertexts. One of the few exceptions to this is the homomorphic encryption scheme of Boneh et al. [BGN05] that allowed for multiple additions and only one multiplication using pairings. Though this scheme was a step closer to FHE, it offered no complete solution. For FHE it is crucial that both addition and multiplication operation can be used in unrestrained order and amount on encrypted data, making it possible to perform arbitrary operations on encrypted data without the possession of a private key.

4.2.1 *Gentry Scheme*

In 2009, Craig Gentry was the first to publish a full homomorphic encryption scheme, supporting both addition and multiplication in unrestricted amounts on encrypted data,

referred to in this thesis as Gentry scheme [Gen09]. Gentry's scheme is based upon lattice-based cryptography and the sparse subset sum problem. It uses an innovative bootstrapping approach to building a full homomorphic encryption scheme from a limited homomorphic encryption scheme through inductive reasoning. It starts by taking a weakly homomorphic Goldreich-Goldwasser-Halevi (GGH) based encryption scheme, which allows for addition and multiplication for a limited amount of operations in low-degree polynomials [Mic01][GGH97]. These functions are limited as the ciphertext noise grows by the use of each operation making the ciphertext eventually indecipherable. Gentry "solved" this problem of increasing noise by using a bootstrapping approach that can reduce noise allowing for more operations to be performed on the ciphertext. He inductively proves his bootstrapping approach by showing that it could always reduce the ciphertext noise to allow for at least one more operation by re-encryption (refreshing), changing a certain weakly fully homomorphic scheme into a strongly fully homomorphic scheme. The scheme is made bootstrap compatible by reducing the degree of the decryption polynomial by adding additional information which might cause some additional info to leak. However Gentry proved the security of the key space of his scheme can be reduced to some of the worst-case hardness problems ideal lattices when a correct key-generation process is implemented correctly [Gen10]. The first successful implementation of Gentry's scheme with [Gen09] and without [GH11] bootstrapping were released in 2010. Later it was also shown that implementations of this bootstrapping approach worked on several other known lattice-based weakly homomorphic scheme like NTRUEncrypt [HPS98] and other variations on the GGH encryption scheme [GGH97][MCG08], making them modifiable to fully homomorphic schemes [GH11].

4.2.2 *Current Developments*

There currently have been several improvements over, and optimizations of Gentry scheme. Gentry original scheme required the use of ideal lattice, but a fully homomorphic encryption scheme without the use of ideal lattice based on elementary modular arithmetic also proved possible after research of Dijk et al. [VDGHV10]. These two models and their optimizations are often referred to as first generation fully homomorphic encryption schemes, as they differ from more recent (second generation) model developments in terms of stronger computational and storage limitations. First generation FHE have a computational complexity that takes in the order of minutes per bitwise operation and require public keys of size 10^9 Megabyte in order to comply with reasonable current day security standards [Mor13] [SV10]. Second generation fully homomorphic encryption scheme are based on the hardness of the solving the Learning With Errors (LWE) machine learning problem introduced by Regev in 2005 [Reg05] or on a variant of the lattice-based NTRU problem [LATV13]. This allows second generation FHE schemes a much lower increase of noise in their ciphers during homomorphic computations, decreasing the frequency in which ciphers have to be re-encrypted.

The following list will include the four [Yun13] best-known types full homomorphic encryption schemes and their main improvements, giving an impression of the current

state of the art of FHE schemes and their developments. Several of these schemes are also included in the by IBM in 2013 released homomorphic encryption library HElib [HS14a], which is publicly available (GPL license) and will be referred to section 5 to serve as the baseline for current homomorphic encryption capabilities.

First Generation

1. FHE based on ideal lattices [Gen09]. *Note: Included in the HElib library [HS14a]*
   - Optimization of key-generation proves that worst-case hardness is equal to that of some ideal lattice problems [Gen10].
   - Optimization that reduces the amount of vectors needed for calculations regarding the "sparse subset-sum" problem [SS10].
   - Optimization that enables the trade-off between lower degrees of decryption polynomial at the cost of small increases in the probability of decryption errors [SS10].
   - Optimization that eliminates the need for the determinant of the lattice to be prime. Reported performance of 30 minutes after each multiplication to renew the ciphertext and the use of 2.3×10^3 Megabyte public key [GH11].
2. FHE over the Integers [VDGHV10]
   - Optimization that proves that public key elements can be in quadratic form while remaining semantically secure, allowing for improvements in key space requirements [CMNT11].
   - Introduction of a new modulus switching technique allowing for replacement of the original bootstrapping scheme. Reported performance of 6 minutes after each multiplication to renew the ciphertext and the use of 10.1 Megabyte public key [CNT12].
   - Introduction of batch-capability and performance optimizations achieving homomorphic evaluations of approximately 12 minutes per AES ciphertext on a desktop computer, achieving comparable result to the implementations of LWE at Crypto 2012 [CCK$^+$13].
   - Optimization obtained through use of the scale-invariant property allows for the implementation of a more efficient evaluation of the AES. Encryption circuits achieve a performance of 23 seconds per AES block evaluation at a 72-bit security level and 3 minutes per AES block at a 80-bit security level. [CLT14]

Second Generation

1. Scheme based on (Ring) Learning With Errors (R)LWE [BV11a] [BV11b] [BGV12].
   - Blueprint idea of the Learning With Errors based Fully homomorphic encryption schemes [BV11a]
   - Introduction of modulus switching to reduce noise which forms the core of the (R)LWE FHE scheme [BV11b] [BGV12]. *Note: Included in the HElib library [HS14a]*

- Optimization of the public-key model achieve a complexity of $k \cdot \text{polylog}(k) + \log |\texttt{DB}|$ bits per single-bit query, in order to achieve security against 2^k-time adversaries [BV14]. *Note: Included in the HElib library [HS14a]*
- Introduction of unidirectional proxy re-encryption later adapted in LWE FHE improves efficiency of the Weakly/Somewhat Homomorphic Encryption (SHE) [LV11].
- Introduction Homomorphic evaluation of an AES circuit [GHS12c]

2. NTRU-Based FHE [LATV13]
   - Blueprint idea for a new type of second generation FHE that is not based on the Learning With Errors problem, but uses the open source public-key cryptosystem NTRU [LATV13]. *Note: Included in the HElib library [HS14a]*
   - Proof on improving NTRU-Based FHE to allow for multiple key's [Che14].
   - Optimization of LWE by [BV14] is shown not to be compatible with the NTRU based FHE due to its ciphertext-packing techniques [GHS11].
   - Optimization by means of a new bootstrapping algorithm increasing efficiently of the NTRU approach to quasi-linear $O(\lambda)$ number of homomorphic operations on GSW ciphertexts under the 2^λ security assumption [ASP14].

In recent years, there have been many improvements regarding the speed, security and compatibility of FHE schemes. The first ideal lattice-based FHE scheme introduced by Gentry was impractical for any real world applications showing a gap between theoretical and practical availability. Second generations FHE though still largely impracticable have limited that gap allowing for somewhat usable implementations that have been made easily accessible by IBM's open source HElib library. The best performance of the an FHE in the HElib library in late 2014 was reported to be able to evaluate one AES-encryption input in about 2 seconds [HS14b]. From these benchmarks we conclude the FHE provides no practical, implementable solution to enforce data privacy of personal data stored at third parties. For a more practical implications, we investigated several database schemes trying to achieve data privacy by leveraging the limitations in FHE while still providing a broad range of functionality.

5

DATABASE ENCRYPTION MODELS

To evaluate practical solutions that enforce the confidentiality of personal data processed in the Cloud, we set out several recently released open-source encryption tools/schemes. All the tools we selected and investigated are compliant with our RDW use-cases, in the sense that they can handle SQL databases in existing major cloud solutions (e.g. Microsoft Azure, Amazon EC2 or Google Compute Engine [ZCB10]) without entrusting these providers with a private key as explained by the legal limitations set out in chapter 3.4.2.

5.1 TYPE OF MODELS

Our main focus was on the MIT encryption scheme of **CryptDB** [PZB11] as it a claims to support most common queries on encrypted data while preserving security guaranties regarding confidentiality without the need of dedicated hardware, as is further discussed in section 5.5. CryptDB also received positive reviews in the media having Forbes describing it as a breakthrough [Gre] and Google crediting CryptDB's design in its new encrypted big query client", according to the CryptDB homepage [oT11]. Four other tools we considered where: **C-SDA** [BP02] , **GhostDB** [ABB+07], the **Helib** library [HS14b] and **Naive Approach** which is a simple ad hoc solution. We considered these tools for their different approaches in obtaining data confidentiality in SQL environments compared to CryptDB. Interesting is that these other approaches lead to different priorities and constraints in regard to performance, functionality and requirements as visualized by a Venn diagram in figure 2.

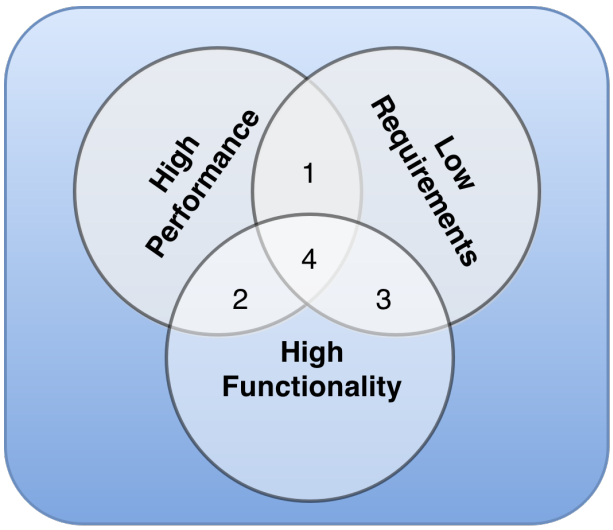

Figure 2: Venn diagram on the position of SQL encryption schemes in regard to their performance in terms of query processing speed, functionality in terms of supported types and requirements in terms of needed dedicated hardware or additional external services. 1: CryptDB & Naive approach 2: C-SDA & GhostDB. 3: Helib's FHE.

The tools we included in our thesis are limited in either functionality (area 1), performance (area 3), or by requiring additional hardware (area 2). An ideal scheme would have none of these limitations and be in area 4 of figure 2. Area 4 can be considered the field of efficient fully homomorphic encryption schemes without functionality or platform limitations. Fully homomorphic encryption schemes are currently not yet efficient as second generation FHE perform N operations on encrypted data with a best $N \cdot polylog(s)$, where s is the security parameter [GHS12b] [GHS12a] [GHS12d], as explained further in chapter 4.1. This leads us to consider tools in area 1,2 and 3 and the impact of their limitations on our use cases set out in chapter 6.4. A short description about these tools, including the motivation to why to include them in this thesis, can be found below in the form of a compact overview.

1. **Naive Approach :** Is an ad hoc solution to our case description of chapter 6 based on deterministic encryption. This solution will serve as a baseline for the amount of functionally achieved by simple data encryption solutions in line with Occam's razor. Characteristic to this approach is that it does not rely on dedicated hardware and works under the same constraints as CryptDB. We refer for more details on our concept of a naive approach to section 5.2

2. **C-SDA :** Stands for Chip Secured Data Access and is a model based on the idea to insulate data encryption, query evaluation and access right from the server to the client by the use of secure hardware. [BP02]. Characteristic to this approach is the use of dedicated hardware at the client to overcome query limitations on encrypted data. A more detail summary of C-SDA can be found in section 5.3.

3. **GhostDB :** GhostDB uses a model where private data is stored on smart USB key that can be used on a client's PC to access and query both a remote public database and the private data stored on the USB. It forms a secure solution for transporting and working with private data in combination with public data at an untrusted server or untrusted workstations. Characteristic of this approach is the dedicated hardware at the client instead of the server. A more detail summary of GhostDB can be found in section 5.4.

4. **HElib :** Various homomorphic encryption models are described in Chapter 4.2 of which the TRU-Based FHE [LATV13] and LWE [BV11a] [BV11b] [BGV12] are included in IBM's open source HElib library [HS14b]. We have taken the earlier the performance complexity of $N \cdot polylog(s)$ from HElib as the baseline for FHE and set out several of its primary FHE schemes in section 4.2. We will not evaluate Helib further for our use due to performance limitation set out in section 4.2.2, showing how evaluation times of several minutes for basic operations. We, however, do point out Helib as an interesting tool for further FHE research.

5.2 NAIVE APPROACH

A Naive Approach (NA) to secure cloud storage is a simple model implementing encryption of attributes at the server without taking versatility as a goal. We will set out such a simple solution to serve as a baseline in terms of functionality and performance for an encryption solution that does not depend on dedicated encryption hardware.

For this approach, we make two assumptions:

- Queries are unmodified and map one on 1 to our encryption model without optimizations .
- Queries not supported by the server are done client side by requesting all needed data.

A minimal amount of functionality that is required by SQL stalemates is the ability to make equality checks to search for individual data elements.

5.2.1 *Applied Methods*

In NA, we encrypt all private columns at the server including column names with AES in ECB mode (figure 10) and leave all public data encrypted as can be seen in table 2. In this model, clients have both encryption and decryption key and provide no key to the server.

Name	BSN	Legal
Caterena Kolk	316485334	OK
Teske van der Wijk	751753761	THEFT
Teske van der Wijk	751753761	OK

$\longrightarrow$

Name	0xB35	0x313
Caterena Kolk	0x49C	0x2A4
Teske van der Wijk	0xAD3	0xB84
Teske van der Wijk	0xAD3	0x2A4

Table 2: Example based on the TLPRD (table 15) with encryption of private column BSN and public column Legal using 3DES. Left is a table as seen by a client. Right is the table as stored at the server

Queries get modified by the client by encrypting all private attributes in the query before sending it to the server. The server sends back a result in which private attributed are encrypted and require decryption at the client. A simple plaintext query involving equality that can be used is P.1 This query gets encrypted by a client to query E.1 and gets send to the server. The server is able to identify rows that contain the encrypted value "0xAD" in their column and send these back to the client.

$$SELECT \ * \ FROM \ table \ WHERE \ BSN = 751753761; \tag{P.1}$$

$$SELECT \ * \ FROM \ table \ WHERE \ 0xB35 = 0xAD3; \tag{E.1}$$

In this encryption model, it is possible for the server to perform searches including both encrypted and unencrypted columns simultaneously as they do not differentiate. Note that all computational overhead is client side.

5.2.2 *Limitations*

In this model, it is not possible for a client to perform inequality checks like $\leq$, calculation on encrypted columns. Unencrypted columns still maintain all their functionality in terms of query operations as long as no encrypted columns are involved.

Confidentiality guarantees are limited as the use of a deterministic encryption reveals equality in all private columns:
In our example of table 2 we can see that the server can search the encrypted *BSN* column on matching values. In the BSN table, this provides no significant confidentiality compromise as the column *Name* already gave hints which rows to group. However, the equality checks in column "Legal" compromise confidentiality as equality between values release new information about the plain content.

1. The values within "Legal" show a meaning full information in their repetition as the value "OK" is more common than "THEFT", which shown clearer in TLPRD of table 15. Because it is likely that most cars are not stolen it possible for a server to differentiate the groups "THEFT" and "OKE" within the column "legal" due to small set different values.

2. The equality checks in the "column legal" compromise confidentiality if only one entry is known. If the server knows the "Legal" value belonging to "Caterena Kolk" column

is "OK" it knows whether it will be "OK" for all other rows. Note that this problem is less severe for the BSN column as not many values repeat leaving only a very small set of max two entries compromised in the TLPRD.

Another important threat to confidentially that is should be considered in this ad hoc solutions comes from the placement of encrypted columns in the same table as unencrypted columns. Unencrypted columns may leak information about data in encrypted columns obtainable deductive reasoning. In the example provided earlier in table 2 we can see that "Teske van der Wijk" has two entries. Let's assume an attacker knows that there are three columns, Name, BSN and Legal and that there are only 2 type of values for the column Legal of which one is stolen. It is immediately clear for an attacker that the third column "0x313" decrypts to "Legal" because "0xB35" translates to "BSN" due to the fact that it impossible for Teske to have different BSN numbers. The attacker can then also derive that one of Teske's cars is stolen from the fact that she has two different encrypted values in the column labelled "0xB35". Because of these properties it is important for a confidentiality preserving scheme to take the meaning of data into consideration when using a deterministic solution, especially when unencrypted, and encrypted data is stored in the same table.

5.3 C-SDA

Chip-Secured Data Access (C-SDA) is a model proposed by Bouganim et al.,[BP02] to enforce data confidentiality in untrusted database environments. We will set briefly explain the basic idea behind of C-SDA describing the involvement of a smart card. We will then set out the limitations of C-SDA illustrating the why this type of solutions is unsuited for our use case described in section 6.4.

An important aspect of C-SDA is the use of smart cards at the servers location to act as a trusted mediator between the client and the server, proving both control over access rights and secure encryption possibilities as illustrated in figure 3.

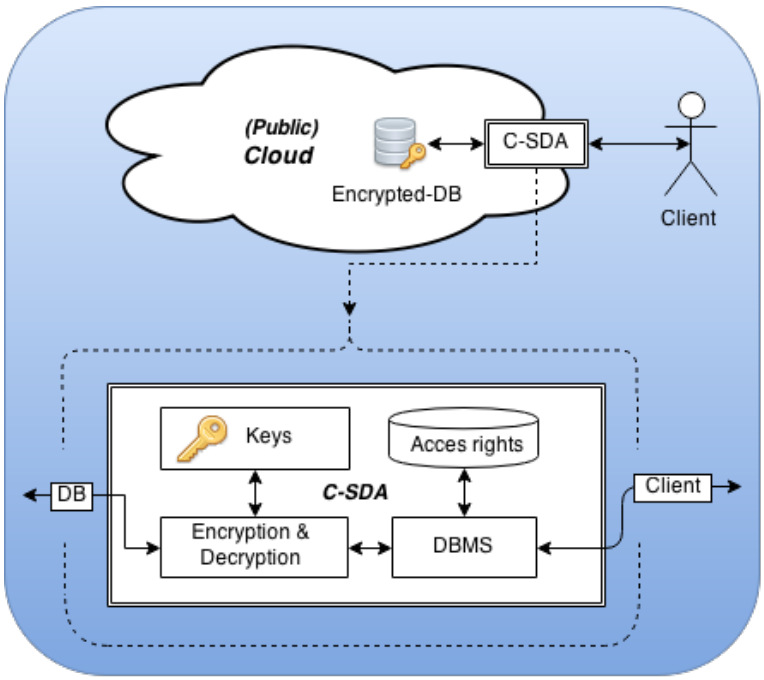

Figure 3: C-SDA's scheme for confidentiality adapted from [BP02].

In our threat model of section 6.5 we focus on data confidentiality. C-SDA enforces confidentiality by having a smart card that performs operations on sensitive data when they those cannot be trusted to the server. C-SDA implements this using a new query strategy that takes the flowing three entities into account: a client, the smart card, and the server. Each of these entities can perform computations and handles a subset of queries. The computational load is being balanced to the server by default if no confidentiality compromises are required. Computational resources of the client and smart card are exploited when specific queries are requested required conventionality guarantees that the server is unable to provide.

5.3.1 *Sub-Queries*

This is implemented by a model that splits queries (Q) into sub-queries in the form Q-server, Q-card, Q-client:

Q-SERVER Are queries that can be handled by the server without strong confidentiality compromises. Compared to CryptDB we could classify this as queries functional on a DET encryption layer allowing for queries based on the "Deterministic" primitive operator, as explained in 5.5.1. C-SDA assumes a block encryption algorithms like DES [oST93] or Blowfish [Sch94a] as they satisfied equality checks on encrypted data and were able to work efficiently with large amounts of data [Ebe93]. The confidentiality compromise made by having a visible correlation between the encryption of equal values is deemed acceptable and only weakly intrusive as also classified by CryptDB's standards as shown in Table 6. These block ciphers are however not order preserving,

preventing order based queries to be computed effectively. C-SDA does allow for the implementation of Order Preserving symmetric Encryption [BCLO09], but at the cost of confidentiality guarantees as described for OPE in 5.5.1.

Q-CARD The smart card evaluates all sub-queries based on the primitive operator not supported by the server. This allows C-SDA to perform homomorphic operations on its data as it enables aggregation functions and decryption and encryption operations if required. This requires the computational load on the smart card that is at least linear in time of the number elements taken for intimidated decryption. Complex queries are therefore not efficient in the use of RAM, and calculation should not exceed the complexity of basic operations like inequality checks, summations and determining an average. Constraints on the smart cards hardware are difficult to mediate as it precludes the generation of intermediate queries according to [BP02] because of the following limitations:

1. Random-Access Memory (RAM) cannot support intermediate queries.
2. RAM cannot transfer the results to EEPROM due to high write costs.
3. RAM cannot transfer the results to external terminals due to confidentiality risks.

Q-CLIENT The client is only required to evaluate sub-queries related to result in the presentation like distinct operator.

5.3.2 *Smart cards*

C-SDA implements several optimizations to optimize performance by providing minimal load at the smart card. Load on the smart card will however still be significant as it handles all request regarding queries with requirements going beyond equality or order checks depending on the chosen server encryption. Current day smart cards have limited performance in cryptographic as can be found in recent benchmark [HMMT14]. A common type of smart cards like JavaCards, .NET cards and MultOS cards perform poor on atomic operations as Hash functions, Random Number Generation, and Big-Integer Modular Arithmetic Operations failing to achieve reasonable execution times [HMMT14]. These operations are at the core of many cryptographic schemes indicating the difficulty of software optimizations [WY05]. Side notes in [HMMT14] research were made stating that though the performance of their model was limited in their benchmark using "normal" smart cards, the processing time of cryptographic operations might significantly be increased using hardware-accelerated cards. This statement is also supported by recent research [HBB13], where hash-based signature generation on a hardware-accelerated smart card was achieved at a low run-time complexity, showing practical possibilities for a smart card based solution.

5.3.3 *Limitations (DH at the Server)*

Though C-SDA implements several method to improve query handling performance [BP02] bottle necks remain in the following areas :

COST AND FLEXIBILITY C-SDA requires the use of dedicated hardware adding cost. Prices of smart cards vary but can considered negligible as they can be bought for a few dollars [BP02]. The placing of smart cards at a service location will also take time and might require additional agreements or audits as hardware is being transferred to and integrated with the server. This also limits the flexibility in which a database model can increase capacity or be migrated to other servers that for example might offer better pricing at that time.

SECURITY Smart cards are known for their tempering resistant designs which provide a high degree of physical security [RE10] [Heno1]. However, attacks have been devised against smart cards including Simple Power Analysis (SPA) [KJJ99] and Differential power analysis (DPA) [KJJ99]. This would give an adversary an unrestricted amount of time to attack the physical security of the smart card adding a new attack vector to a security model.

CONFIDENTIALITY C-SDA provides security under the need for certain functional requirements. C-SDA implements this by using a mono-encryption layer encrypting everything at the server using the same encryption scheme, opposed to CryptDB's multi-encryption layer approach specified in section 5.5.2. This leads to weaker security guarantees as columns that do not require equality or order checks might still leak those type of data correlations given that confidently guarantees are equal over all columns.

PERFORMANCE C-SDA depends on a smart card which has limited processing power and memory and would, therefore, become the bottleneck when this solution is scaled.

5.4 GHOSTDB

GhostDB [ABB+07] is a database encryption scheme that relies on secure hardware to enforce data confidentiality in untrusted database environments, like models as C-SDA. The way GhostDB deploys secure hardware is fundamental different from C-SDA as the secure hardware is used client side and requires no addition cooperation from a server provider. The way GhostDB achieves this is by splitting a database into a private and public part, entrusting only the public data to a server and keeping the private data stored at a secure token.

5.4.1 *Separate Databases*

The basic architecture of GhostDB is shown in figure 4, illustrating the placement of public/private data and the way a client accesses data through GhostDB.

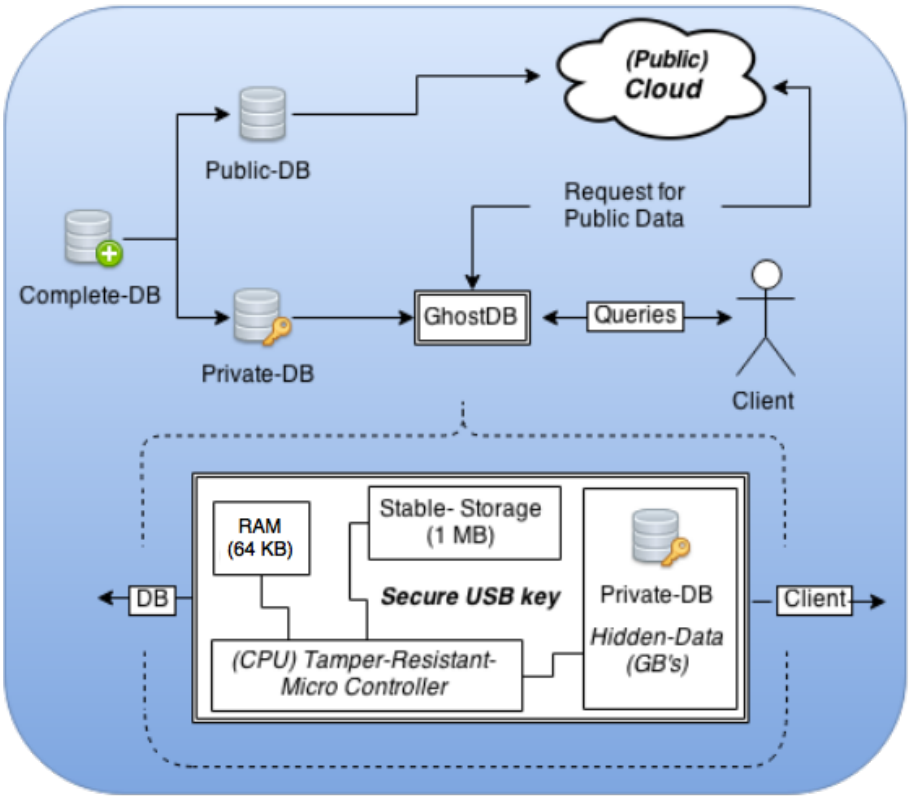

Figure 4: Illustration of the basic architecture of GhostDB. Complete-DB contains both private and public information and is split into an encrypted private part stored at secure GhostDB hardware and a public part placed at an (untrusted) server (i.e. Cloud). In GhostDB's initial paper [ABB+07] secure hardware was used in the form of a secure USB 2.0 key consisting of the following essential components and their selected capabilities : 64 RAM, 1 MB stable-storage, a temper resistant micro controller and several GBs of secure storage's.

Their client can use queries as normal, letting GhostDB relay them to a remote database if only public information is involved. For queries requiring private information GhostDB can access and process the private data stored on the secure token (i.e. Secure USB) and use it in combination with public data processes and provided by the cloud. Techniques like climbing indexes, subtree key tables and post-filtering by Bloom filters are used to improve performance allowing the use of GhostDB for complex queries and large databases [ABB+07]. This method has several advantages as it allows for the linkage of sensitive data with public data assuring that no private data gets leaked to an untrusted server. So is it possible to use GhostDB in insecure environments only risking the leakage of the query results, making it a possible solution for traveling scenarios in which a client has no access to a trusted computer.

5.4.2 *Limitations (DH at the Client)*

GhostDB has no query limitations in the sense that a secure token can perform all SQL operations a personal computer or server can. It is however obviously that the placement of private data on a secure token limits the benefits of a cloud solution that can be exploited, as all processing of private data is done entirely by the secure token. Two aspects of this not yet mentioned at the limitations for dedicated hardware (at the server) in subsection 5.3.3 and are specific for GhostDB's client-side secure hardware are:

MULTIPLE USERS Limitation arises when GhostDB is used by multiple clients performing query operations that are not limited to search, modifying private data. A secure server has to be setup and security risks from stolen, and lost tokens have to be considered as those store private data.

PERFORMANCE The secure token is limited in terms of RAM and CPU power that will form a bottleneck for query processing performance. Note that resources of the secure token are not extensively used for public data processing as this can be done in the cloud.

5.5 CRYPTDB

CryptDB is a cryptographic DataBase Management System (DBMS) developed at MIT [PRZB11a]. It aims to provide data privacy guarantees in the face of a compromised server by enforcing data and query encryption managed from a trusted proxy environment. This trusted CryptDB proxy can run from a client's personal computer or local server and does not require specialized hardware. A basic CryptDB's deployments model is illustrated in figure 5.

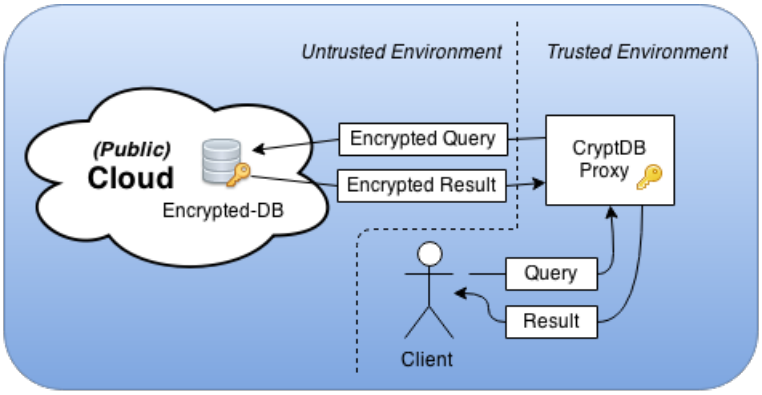

Figure 5: CryptDB deployment overview. CryptDB runs as proxy in a secure environment (e.g. Client's server) and handles all encryption and decryption processes. CryptDB stores a master key for all cryptographic processes and requires no client side input other that the to be executed query, which remain unmodified from a client's perspective.

A client sends an encrypted query request to the DB server that executes the encrypted query on the encrypted DB without being able to completely decrypted the addressed values. The DB server then returns the evaluated query result back to the user for final decryption. To achieve this CryptDB's model is constructed on the following 3 ideas that keep usability in mind [PZB11] :

1. Using an encryption strategy that is SQL specific by mapping SQL operations to encryption schemes based on the functionality required by their primitive operators as described in section 5.5.1.

2. Implementing a secure proxy that allows for efficient adjustments of each data items encryption level. Providing the model with query-based encryption levels providing a trade-off between security and required functionality at runtime as explained in section 5.5.2.

3. The use of onion encryption structure to facilitated the different encryption layers aiming to give a maximum level of security given the needed functionality as is explained together with the second idea in section 5.5.2.

5.5.1 *Mapping to Different Encryption Models*

In order to map SQL operations to an encryption scheme CryptDB uses the fact that SQL databases are relational databases that rely on relational algebra that define each SQL operation with a limited set of primitive operators (i.e. equality checks, sums and joins) [Cod82]. These primitive operators can be mapped to different encryption schemes as they require different mathematical properties. An example of this is that deterministic encryption is required for equality checks but not for addition. CryptDB leverages this by implementing different encryption schemes for different primitive operators that result in adaptive security guarantees that we will explain in section 5.5.2. By default, CryptDB uses existing and well proven cryptographic schemes for most of these operators as:

RND : Cipher Block Chaining (CBC) [EMST78] for pseudo-randomness, as described at section 5.5.1.1.

HOM : The Paillier crypto system [Pai99] for addition as described at section 5.5.1.2.

SEARCH : An adaption of Song et al.'s encrypted word search [SR01] [SWP00], as described at section 5.5.1.3

DET : AES [DR02] & Blowfish [Sch94a] for the allowance of equality checks, as described at section 5.5.1.4.

OPE : A CryptDB specific scheme based on Boldyreva at al. [BCLO09] for cross-column equality, as described at subsection 5.5.1.5.

JOIN : A CryptDB specific scheme based for order-preservation, as described at section 5.5.1.6.

The encryption scheme's for these primitive operations can be chosen different, as long as they still allow for sufficient functionality and provide strong security guarantees. In practice, this results in a trade-off between functionality and security guarantees, as broader functionality decreases the security guarantees that can be provided as illustrated in figure 6. An example of this is the earlier mentioned deterministic encryption for equality checks. Deterministic encryption allows for computationally efficient equality checks, but it also enables an attacker to observe data patterns that occur when values are used multiple times.

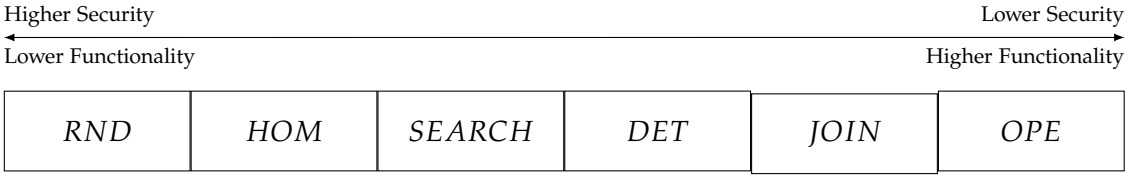

Figure 6: Primitive operators used by CryptDB and their respective negatively correlated ordering in terms of confidently guaranties and functionally.

In the following section, we will set out how CryptDB implements encryption schemes for the different operator found in figure 6 and how CryptDB obtains different functionality and confidentiality guarantees for these operators. In the following section of 5.5.2 we will set out how these different schemes are combined in onion based encryption structure to protect the weaker security guarantees based operators.

5.5.1.1 *RND : Randomizing Ciphers*

RND is CryptDB's equivalent for the primitive operator "random", which is designed to provide maximum security under the least functional requirement. It ensures that two equal values are mapped to different ciphertexts protecting the database from correlation based attacks. By default CryptDB uses secure Cipher Block Chaining (CBC) [EMST78] in combination with the block cipher Blowfish [Sch94a] for integer values and in combination with the Advanced Encryption Standard (AES) [DR02] block cipher for all other values, as further explained at the DET operator in chapter 5.5.1.4.

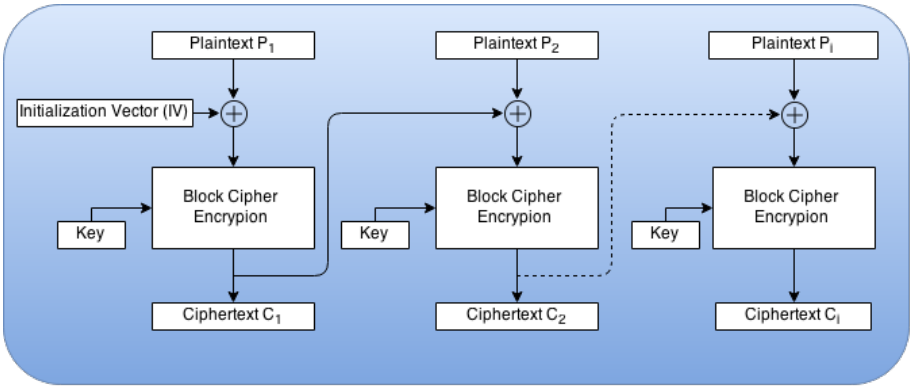

Figure 7: Cipher Block Chaining (CBC) : Each block of plaintext is XORed with the previous cipher block before being encrypted. If there is no previous cipher block an Initiation Vector IV is used instead

CBC is used in both cases to prevent correlations between similar values, as CBC encrypts values different based on the last encrypted cipher block or an initialization vector creating a form of pseudo-random [BKR94]. The generation of CBC pseudo-random cipher text as shown in figure 7, where $\oplus$ denotes the XOR operation used between the plaintext and a previously obtained value to serve is random input for a block cipher. This leads to the following encryption and decryption functions used in CBC :

ENCRYPTION : $C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV.$

DECRYPTION : $P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV.$

5.5.1.2 *HOM : Homomorphic Encryption*

HOM is an integer specific primitive operator used for homomorphic operations on encrypted data by CryptDB. It uses the Paillier crypto system, which is probabilistic asym-

metric algorithm for public key cryptography invented in 1999 by Pascal Paillier [Pai99]. The Paillier cryptosystem uses the following algorithm:

PAILLIER CRYPTOSYSTEM :

KEY GENERATION :

1. Select two large prime numbers p and q.
2. Determine $n = pq$.
3. Determine $\lambda = \text{lcm}(p-1, q-1)$
4. Select a random integer $g \in \mathbb{Z}^*_{n^2}$ (g in invertible modulo n^2) where n and $(L(g^\lambda \bmod n^2))$ are coprime.
   - Function L denotes $Z^*_{n^2} \to Z_n$ by $L(u) = \frac{u-1}{n}$
     - Where $\frac{a}{b}$ denotes the quotient of a divided by b, i.e. the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.
   - λ demotes the Carmichael function $\lambda(p * q) = lcm((p-1)(q-1))$.
5. Take as (public) encryption key (n, g).
6. Take as (private) decryption key (p, q).

FUNCTIONS :

ENCRYPTION $c = g^m \cdot r^n \bmod n^2$.

- Where $r \in \mathbb{Z}^*_n$ is a randomly selected integer.
  - Where m is the plain message in $\mathbb{Z}_n$.
  - Note that r is used in this probabilistic scheme so that a given plaintext can have multiple different ciphertexts. This means that m does not deterministically leads to a certain c which prevents information leakage causes by repetitive use of the same input value m as we explain in section 5.5.1.4 on deterministic models.
  - Where c is a ciphertext in $\mathbb{Z}^*_{n^2}$.

DECRYPTION $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)}$.

- Note that knowledge of r is not needed for the decryption of c.

ADDITIVE HOMOMORPHIC PROPERTY :

- Two ciphertexts c_1, c_2 will decrypt to the sum of their corresponding plaintexts m_1, m_2 when multiplied.
  - $c_1 = g^{m_1} * r_1^n \bmod n^2$.
  - $c_2 = g^{m_2} * r_2^n \bmod n^2$
  - $c_1 * c_2 = g^{m_1} * r_1^n * g^{m_2} * r_2^n \bmod n^2 = g^{m_1+m_2} * (r_1 * r_2)^n \bmod n^2$
    - Note that for a plaintext m_3 it holds that $m_3 = m_1 + m_2 = c_1 * c_2 = c_3$ with $r_3 = r_1 * r_2$. The addition of two values is thus computed using multiplication under encryption instead of addition.

Paillier is used for its additive homomorphic properly allowing the addition of encrypted values without the need for decryption. Two cipher texts c_1 and c_2 of two corresponding messages m_1 and m_2 can be used to compute a correct cipher for the message $m_1 + m_2$ with a very high probability according to Definition 4.2 of Partial Homomorphic Cryptosystems. The cipher of $m_1 + m_2$ can be obtained by the multiplication of c_1 times c_2, resulting in a cipher of $m_1 + m_2$ modulo some public key value. CryptDB uses this property for its HOM operator which can for example map the SQL primitive of SUM to the Paillier cryptosystem (e.g. changing addition for multiplication $x_1 + x_2 \rightarrow c_1 \times c_2$), allowing for summation without full decryption and with privacy guaranties [PRZB11a].

The Paillier cryptosystem is proven semantic secure against chosen-plaintext attacks (IND-CPA) based on the Decisional Composite Residuosity Assumption (DCRA) which is believed to be intractable as there is no probabilistic polynomial time distinguishes known that can solve this problem [RV05]. The Paillier cryptosystems is, however, less secure then CryptDB's RND implementation as its homomorphic property makes it malleable, meaning that it does not protect against adaptive chosen-ciphertext attacks (IND-CCA2) which allows an attacker to manipulated a ciphertext to another ciphertext that decrypts to a related and possibly meaningful plaintext [Sah99].

5.5.1.3 *SEARCH : Word Search*

Is a text specific encryption variant of the primitive operator "Word search" implemented by CryptDB. It uses a modified and extended implementation of Song et al. [SWP00] encrypted search scheme that provide provable security against an untrusted server performing query analysis in an attempt to reveal the plaintext [PRZB11a]. SEARCH provides the following three techniques that provided probable security.

1. Controlled Searching (CS), which prevents an untrusted server from searching for a word without the client (implicit) authorization. A server can only match tokens (search) in a sub-domain for which it has obtained a key. This sub-domain can consist of part of a column but also a sentence.

2. Hidden Queries (HQ), which allow a client to request a untrusted server to search for a word, without the need for the client to reveal that word to the server.

3. Query Isolation (QI), which means that the untrusted server learns nothing other than the search result for the query, resulting in a minimum amount of information leakage to the server.

SEARCH is implemented only for MySQL statements and supports operations like LIKE enabling query's as " SELECT * FROM messages WHERE msg LIKE word". It is however not implemented to search for less than full words like arbitrary regular expressions and will leak the number of expressions searched for. The number and complexity of word searches can be increased by using multiple LIKE, AND, and OR operations. This allows for a wide range of variety and might supports the implementation of some arbitrary regular expressions as their incompatibility is not necessarily based on a theoretical limitation by CryptDB [PRZB11a].

The basic schemes explaining the principles of CS, HQ and QI as derived from Song et al. [SWP00]:

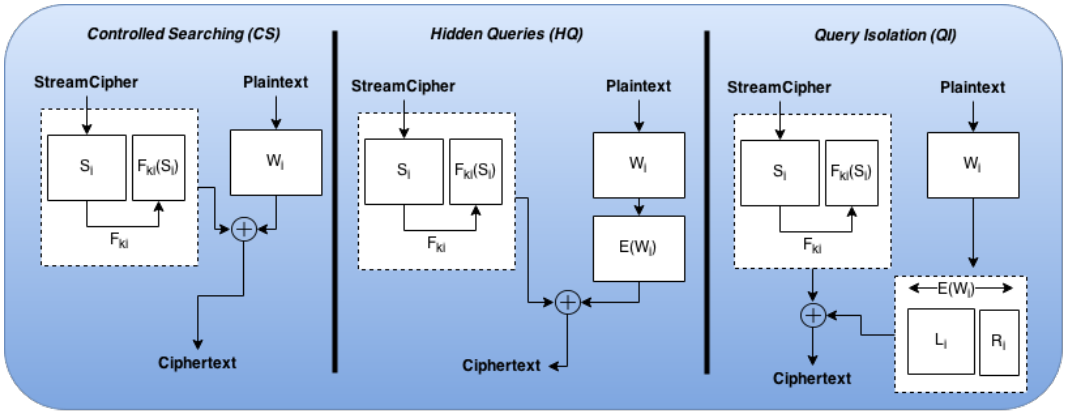

Figure 8: Basic scheme for Controlled Searching (Left), Hidden Queries (Middle) and Query Isolation (Right). A plaintext W is divided in some individual words $W_1, W_2, ..., W_n$ in which we refer to a single word as W_i. Stream-cipher S produces different cipher stream S_i, which gets concatenated with the result of pseudo-random function F. Function F produces a cipher based on input S_i and key K_i and can be seen as a hash function. Exclusive or is in the figure annotated by ($\oplus$) and used in every encryption as the final step to combine text-based encryption with a search key k_i based cipher. In QI $E(W_i)$ annotates an encryption function with input w_i and can be split in sub-ciphers L_i and R_i as further explained in 5.5.1.3.

In figure 8 we illustrated the basic schemes for CS, HQ, and QI. Note that each scheme is based on the previous one extending its secrecy guarantees. HQ will be explained in more detail in figure 9 as it also explains the concept behind CS and illustrates the limitation that is solved in the explanation of QI.

1. Controlled Searching (CS) works by encrypting each word W_i in a plaintext using an exclusive-or operation with a secure pseudorandom cipher combination. This ensures that every word is encrypted differently preventing unauthorized equality checks. Authorized checks will not be performed by reviling the used XOR input's as this would decrypt all the ciphertext including word that are not relevant to the search. A search is done by the server by using XOR on a ciphertext with the search word W_s producing the original cipher input of the form S_i concatenated $F_{ki}S_i$ where $F_{ki}S_i$ is secure pseudo-random permutation on S_i using a location-based key k_i. Note that is unknown to the server at this point what the form of (S_i, $F_{ki}S_i$) is, as k_i is a secret key needed to perform this check. A client can now control a servers search by revealing only the k_i values of locations in the document where a server is permitted to search. This can, for example, be set up by having the key values correspond to different rows, lines or word counters preventing equality checks beyond a certain range that can be determent on run time.

2. Hidden Queries (HQ) resembles Controlled Searching with the adaptation that the search words are encrypted. This prevents the apparent leakage of search criteria revealed in CS as explained in figure 9.

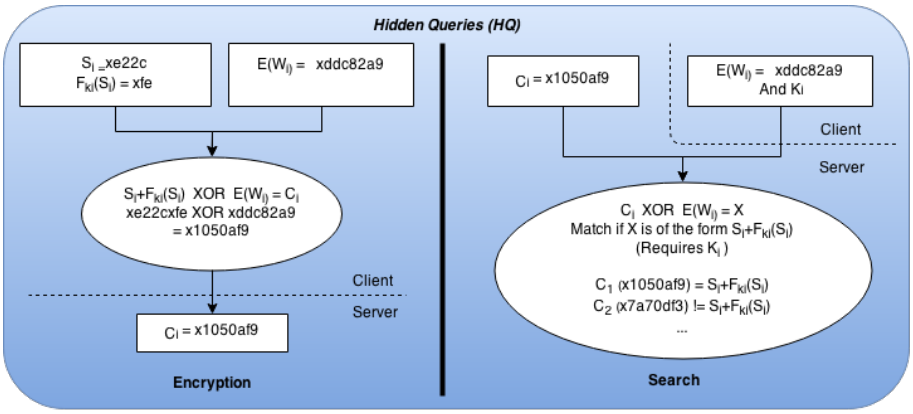

Figure 9: Explanation of the Encryption and Search construction of the Hidden Queries scheme. S_i is generated with a random stream cipher and used in function F which is publicly known and has the property that a key K is required to derive its result, which contains a distinguishable correlation to its input. $E(W_i)$ is the encryption of word W_i and is hidden from the server until a search request is performed by the client, in combination with one or multiple range keys K. A Server can search for an encrypted word by matching whether it matches a key based pattern of F when it is xor-ed with a stored ciphertext. Note that the used key(s) during token matching is not based on $E(W_i)$ by the position of C_i

In 9 we have the encryption of values performed by the client and the search routine of the server receiving an encrypted word W_i and key K_i from the client. Because a server never receives the plaintext value of W_i it is unable to know the search criteria and remains only able to match token X for cipher text of which it knows K_i. A client is likely to release multiple keys belonging to a defined search ranges that can, for example, be defined as the column number of the text. The check whether C_i is of the form $S_i + F_{ki}(S_1)$ can be performed efficiently by the server once the corresponding K_i as F is chosen as such and known to the server. Note that if k_i is selected as $F_k(E_k(W_i))$ a limitation arises as a plaintext no longer be derived from just the obtained ciphertext.

3. Query Isolation (QI), is an extension on HQ in which a untrusted server was able to search a word with the knowledge that word as it only obtains an encryption. A limitation in HQ and CS is that a client is no longer able to derive the plaintext from just the ciphertext. In HQ key k_i is needed in combination with a ciphertext and stream cipher S_i to derive E(W_i). QI solves this by splitting the encryption of word E(W_i) into two parts (L_i,R_i). Where L_i is the first n number of bits of $E(W_i)$ and where R_i is of the remaining m number of bits. L_i is taken of equal length to S_i so that it can be obtained from the ciphertext. Instead of generating $k_i \equiv F_k(E_k(W_i))$ to decrypt, a client can generate S_i as the seed of the stream cipher is known to the client. This

allows a client to recover L_i by using exclusive or on S_i against the fist n bits of the ciphertext. From L_i a client can compute ki and thus decrypt the full ciphertext, as is proven by Song et al. [SWP00]. There is a probability that two different plaintext result in the same L_i. Using the birthday paradox it is proven that the probability of at least one collision after encrypting x words equals $\frac{x^2-x}{2^{n-m+1}}$ [SWP00]. Pre-encryption of words is required to prevent this collision and create query isolation. Query isolation means that even when a single key k_i is known to the server no information is leaked other than ability to identify the position where a $E(W_i)$ occurs. Due to the pre-encryption of words we can assure that no collisions occur and have k_i depended on both the position as L_i. This means that a known k_i cannot be misused to search for other word correlations by brute-forcing different $E(W_x)$ combinations for a certain k_i. This is the case due to the fact that there are 2^m different $E(W_x)$ that provide a correct format of $(S_i, F_{ki}(S_i))$, without the server being able to tell which $E(W_x)$ corresponds to an actual word.

5.5.1.4 *DET : Deterministic Encryption*

Is an encryption variant of the "Deterministic" primitive operator. It's implementation should, therefore, be based on a semantically secure pseudo-random permutation (PRP) scheme [PP05]. CryptDB chooses to implement this similar RND with the use of Blowfish and AES, but without CBC. CryptDB chooses to use the deterministic encryption schemes of AES and Blowfish due to their recognition as strong and efficient cryptosystem [PRZB11a]. The removal of CBC means that DET provides slightly lower security guarantees as RND. RND's ciphertexts are generated deterministically showing a correlation between equal values as they are encrypted to the same ciphertext. The matching of values is, however, necessary to provide the additional functionality that is not present in the RND operator. It allows for equality predicates, equality joins, GROUP BY, COUNT, DISTINCT and similar MySQL operations [PRZB11a]. In figure 10 we set out the encryption scheme of AES as used for String encryption by CryptDB. After figure 10 we will describe CryptDB's motivation to encrypted integers with Blowfish instead of AES and show the Blowfish's encryption/decryption scheme as set out in figure 11. Note that both AES and Blowfish do not include random generators and obtain their cipher based on plaintext and key input giving them their deterministic property.

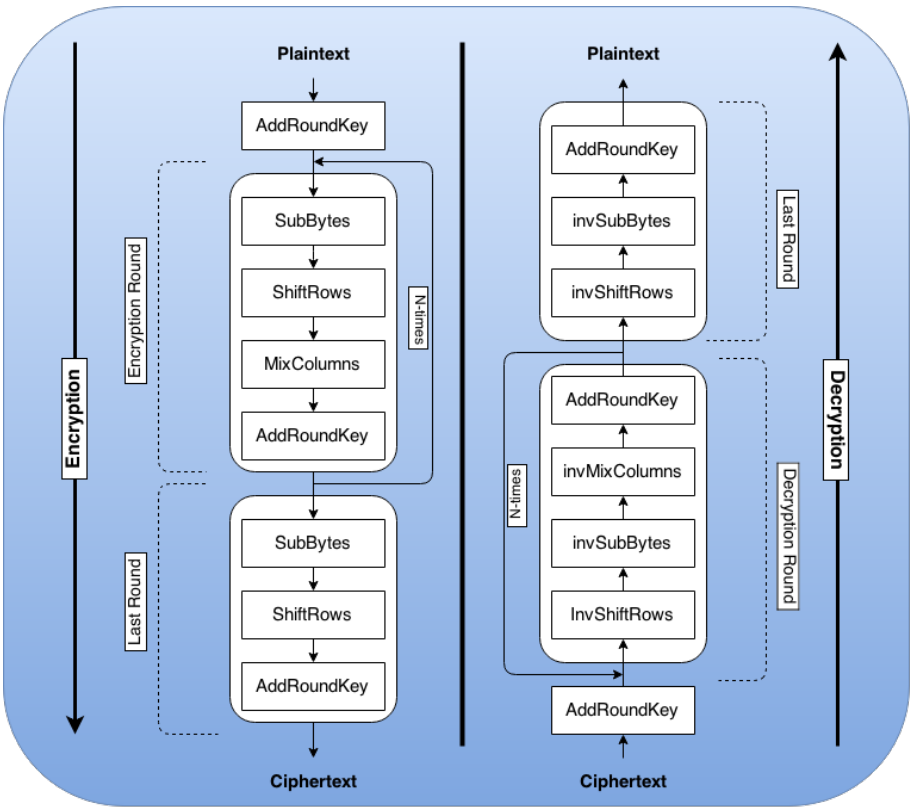

Figure 10: Advanced Encryption Standard (AES). The number of N (cycles of repetition) in this model is depended on the key length with 10, 12 or 14 cycles of repetition for respectively a 128, 292 or 256-bit key key-length. The steps: AddRoundkey, SubByte, ShiftRows, MixColumns and their inverse (inv) counterparts are implemented as matrix manipulation. In **AddRoundkey** each byte of the state is combined with a byte of the round subkey using the XOR operation. In **SubByte** each byte in the state is replaced with its entry in a fixed (8-bit) lookup table. In **ShiftRows** bytes in each row of the state are shifted cyclic to the left with a different shift distance per row of either zero, one, two or three columns. For **MixColumns** each column of the state is multiplied with a fixed polynomial.

The choose to encrypt integers with Blowfish was made due to its shorter 64-bit block size generating relatively short ciphertext compared to AES, trading between efficiency and protection strength [PRZB11a]. This scheme protects against protects against chosen-plaintext attacks (IND-CPA) but is not adaptive chosen-ciphertext attacks (IND-CCA2) secure. This design decision was made due to its threat assumptions that do not include active tempering of the server by default. This can, however, be adjusted by changing to a more secure block cipher like the semantically secure UFE cipher [Des00], as mentioned in one of CryptDB's release papers [PRZB11a] and documentation [oT11].

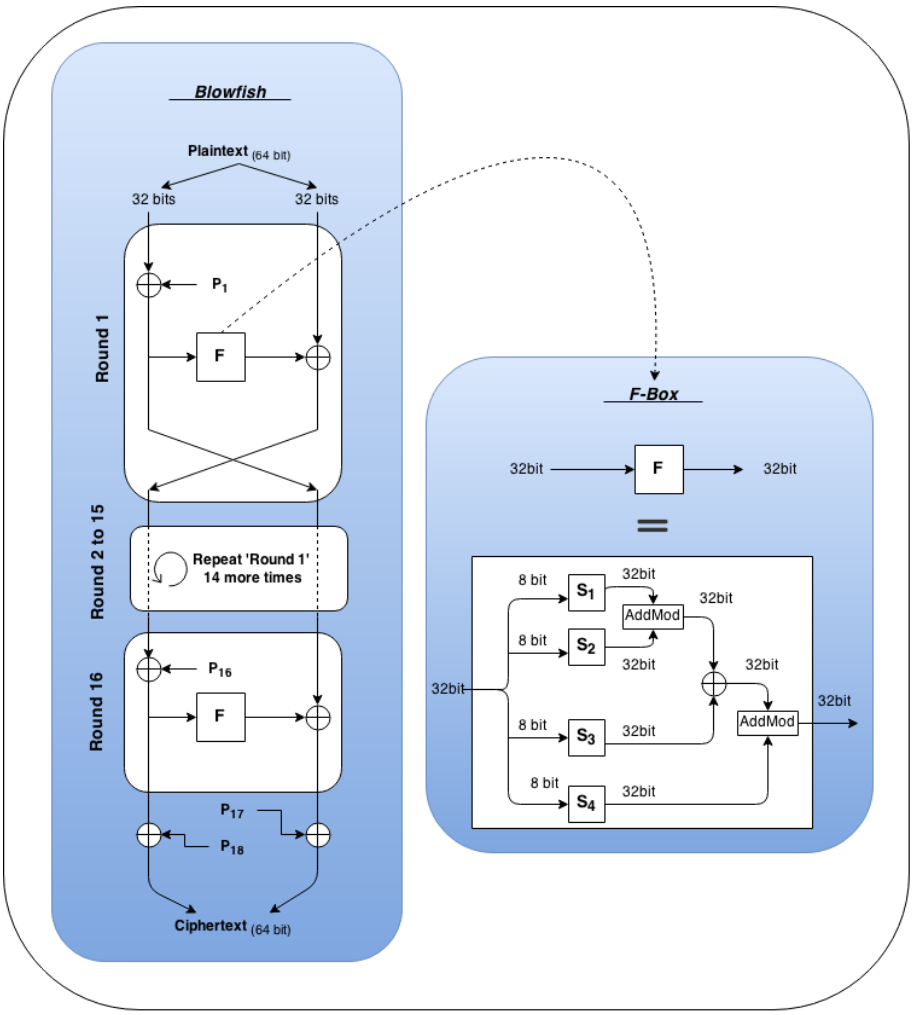

Figure 11: Block cipher Blowfish (left) and schematic of the F-Box as used in the Blowfish (right). This figure contains S-boxes are denoted with S_x, XOR ports illustrated as $\oplus$ and Addmod boxes that represents the addition of two 32bit values modulo 2^{32}. S-boxes are initiated with pseudo-random data derived from hexadecimal digits of π [BFK+96]. Key lengths of up to 576 bits are supported by a P-array that gets dived in 18 parts of 32 bit used in the 16 different rounds and final two XOR operation. Encryption and decryption use the same scheme in which decryption uses P_1, P_2 .. P_18 in reversed order [Sch94a]

Extending CBC to CMC

Though CBC ensures the confidentiality of encrypted data, it does not provide guarantees to data integrity. If a plaintext is known (e.g. due to low entropy) it is possible for an attacker to change every second plaintext block to a freely chosen one by changing the previous ciphertext block to a random value as shown in bellows explanation using AES-CBC from [Lel13].

Attack on CBC:

1. Known value's

p_i : Original plaintext block i.

c_i : Ciphertext block i.

x_i : Chosen plaintext block for i.

2. Notice that $DEC(c_i, key) = c_{i-1} XOR p_i$ since $p_i = DEC(c_i, key) XOR c_{i-1}$, where i is assumed to have a previous block.

3. Derive that $DEC(c_i, key)$ is a known ciphertext block and c_{i-1} can be manipulated so that $p_i = x_i$ by using $c_{i-1} = DEC(c_i, key) XOR x_i$.

4. Change p_i to x_i.
$p_i = DEC(c_i, key) XOR c_{i-1} = DEC(c_i, key) XOR DEC(c_i, key) XOR x_i = x_i$

This attack requires a previous ciphertext block to be changed to a random value, leaving at most half of the blocks of a known plaintext to be manipulated. This can, however, have practical implications as this allows for the injection of shellcode spread out over multiple blocks using JMP institutions [Lel13]. To protect against this type of attack, CryptDB supports the use of CBC-mask-CBC (CMC) [HR03] as an extension on CBC to prevent meaningfully manipulation. In CMC, the CBC ciphertext is masked by XORing with $2(c_0 \oplus c_{i-1}$ and re-encrypted using CBC mode starting from the last block. This provides the guarantee that if the underlying block cipher (e.g. AES or Blowfish) is a strong pseudorandom permutation (PRP) then ciphertext well be a tweakable PRP. The previously mention attack can no longer be performed as a cipher c_i cannot be changed using a random c_{i-1}, as this included in the applied mask.

5.5.1.5 *OPE : Order Preserving Encryption*

OPE is an order-preserving encryption scheme used for sorting operations [AKSX04]. It allows for the deduction of relations between encrypted data values without the need for (intermediate) server-side decryption, maintain data confidentiality. Relations can be derived by the fact that if value X_1 is encrypted under OPE as C_1 and X_2 as C_2 than it holds that if $(X_1 < X_2)$ then $(C_1 < C_2)$ regardless of the private key. This way the server can perform operations as ORDER BY, MIN, MAX, SORT without the need of a private key. The "leakage" of order to the server makes it less secure than RND, but more functional. CryptDB implements its OPE based on Boldyreva et al. [BCLO09] and has provable security. Boldyreva et al. [PLZ13] implementation relatively slow as it took 25 ms per encryption of 32 integers on an Intel 2.8 GHz Q9550 processor according to CryptDB development team that choose to implemented a AVL binary search trees [AVL62] for batch encryption gaining more that 350% performance increase allowing for 7 ms per encryption [BCLO09]. OPE implemented by CryptDB is proven to be Indistinguishability Under Ordered Chosen-Plaintext Attack (IND-OCPA) secure like Xiao et al.'12 [XYH12] and more secure as previous schemes offering security guarantees against IND-OCPA and leakage of information besides order, making it the "Ideal" OPE scheme as claimed by its authors [PLZ13]. The use of search trees for order-preserving encryption can be explained by the following AVL binary tree example adjusted from [PLZ13] shown in 12.

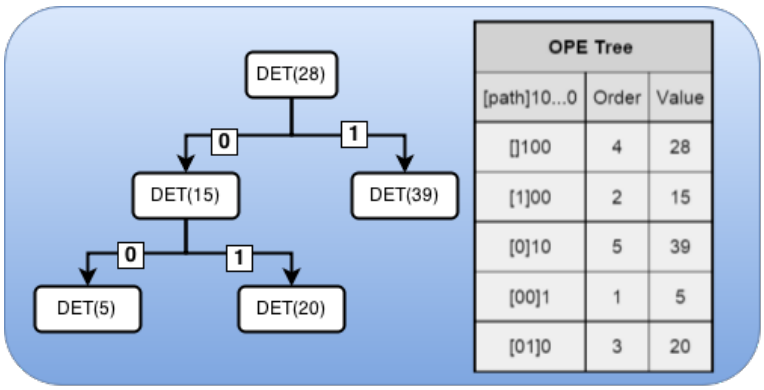

Figure 12: Overview of OPE's encryption model. OPE is constructed as a tree of which the nodes are encrypted using DET (hexadecimal value) illustrated by DET(value), to visualize the tree's balancing. Pointers to children are labelled with 0 or 1 to indicate the path encoding based on [PLZ13]

Suppose we have a list of containing five values: 39, 28, 15, 5, and 20. A potential order preserving encryption function $E(x)$ for these values could be : $E(39) \rightarrow 5, E(28) \rightarrow 4, E(20) \rightarrow 3, E(15) \rightarrow 2, E(5)$ and $\rightarrow 1$ as their ordering from high to low would remain the same. This can be done by a deterministic encryption scheme whose security property is that of a pseudo-random function [Golo4]. The problem for this model is however that a client should be able to add and remove values without having a prior knowledge of present and future values.

Encrypted values are stored as a binary tree at the server by having each cipher store a link to direct nodes. OPE search, addition and removal of values can be done by the client proxy by in logarithmic time in the total number of encoded values [PLZ13]. A client's proxy searches the ALV tree in such a scenario by decrypting and comparing values stored at the server. An example of this can be the addition of the cipher $E(32)$, which place can be found by the client after just two requests. First a client request and decrypts $E(28)$ after which $28 < 32$ the client requests the right place $E(39)$ which is smaller than 32 and has no more children. The client then knows the location of $E(39)$ in the list and adds it as the 6th element encoded by path $[011]$. CryptDB implements and extends on this idea by using hashes as pointers of lower nodes according to a Merkle tree structure illustrated in figure 13

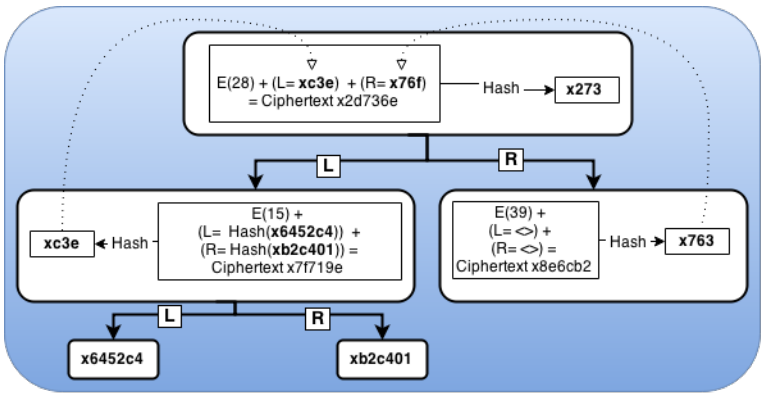

Figure 13: OPE tree of figure 12 adjusted to a Merkle tree based on hashes of the cipher values

CryptDB's uses hash as path indicators in its OPE three model in order to prevent the server from learning the tree's structure. In figure 13 this is illustrated by having the first node encrypted as $x2d736e$ containing the cipher of 28 and the hashes of its direct children. If a client wants to search the OPE tree of figure 13 for a search value S_i, it can request the root of the tree. The server returns this ($x2d736e$) which allows the client to decrypt this into 28 and a left ($xc3e$) and right ($x761$) hash. If S_i is not equal to 28 the client request a search for either the left or right has depended on whether S_i is larger or smaller than evaluated note's value. In this case, we assume S_i to be smaller and the client requests $xc3e$ from the server. Note that at before this point the server was unaware whether that node even contained children and only known what to search for when the client returns a hash. The server can now hash nodes of which it does not know the order until it finds one with the corresponding hash, which it then returns to the client. A search can continue this way until the search criteria are met, only revealing data connection that are strictly necessary to perform a tree based search. Note that hashes are not based on the hashes of children contained within each node as hashed based on children would lead to both information leakage and unnecessary computations. Adding and removing element from the tree can therefore be done in logarithmic time [Szy04] similar to that of the ALV tree, having CryptDB only perform rebalancing of the tree after a certain offset to minimize server overhead.

5.5.1.6 *JOIN : Cross-Column Equality*

JOIN and OPE-JOIN are two CryptDB specific encryption schemes introduced Popa et al. [PZ12] that allows for equality joins between two columns. In CryptDB, all columns have a unique key that prevents cross column correlations to so limits information leakage. This key structure limits query functionality for DET and OPE based operations as they cannot be extended to multiple attributes. CryptDB solves this by introducing the JOIN scheme for DET, which enables a server to draw equality checks cross column and the OPE-JOIN scheme for OPE, which allow the server to perform cross column joins by order relations. JOIN and OPE-JOIN work by having respectively two DET and OPE columns change their private to joined shared private key, without intermediate decryption. JOIN and OPE-JOIN

reduce the number of private key's and, therefore, provide weaker confidentiality guaranty than RND, DET, and OPE. Note that JOIN and OPE-JOIN still minimizes leakages under the demented functionality as they will only be revealed to a server when their functionality is requested by a client at run time, as further explained in section 5.5.2. CryptDB is able to join columns keys by implementing a cryptographic scheme based on Elliptic Curve Diffie-Hellman (ECDH) [JMV01] which is a combination of elliptic curve cryptography [HM11] and Diffie-Hellman key exchange [Mer78] [DH76]. This scheme's details are set out and including proof of guarantees and justification for transitivity relations between joined columns in [PZ12]. CryptDB defines JOIN and OPE-JOIN at the hand of the a cryptographic primitive labelled JOIN-ADJ (adjustable join) which is a deterministic function on an elliptic curve. JOIN is then defined as the concatenation of JOIN-ADJ(x) and DET(x). Because each column has different JOIN key cross column equality checks cannot be performed. To allow for two columns to be joined a client can provide the server with a key to adjust the JOIN-ADJ values of these columns to give them matching keys, thus allowing for cross-column equality checks. To illustrate the basic concept of key matching, we will set out a simplified example JOIN-ADJ's key joining at the hand of table 3.

Column A (Key k_1)	**Column B (Key k_2)**	**Column C (Key k_3)**
$C(2)_{k1} = P^{k_1 * M_{k0}(2)}$	$C(7)_{k2} = P^{k_2 * M_{k0}(7)}$	$C(9)_{k3} = P^{k_3 * M_{k0}(9)}$
$C(4)_{k1} = P^{k_1 * M_{k0}(4)}$	$C(2)_{k2} = P^{k_2 * M_{k0}(2)}$	$C(7)_{k3} = P^{k_3 * M_{k0}(7)}$

Table 3: Simplified example of column encryption using the JOIN-ADJ operator used for both JOIN and OPE-JOIN which adaption of JOIN including tree ordering. JOIN-ADJ is defined as $C(value)_k x = P^{k_x * M_{k0}(value)}$ where k_x is the initial key, P is a point an elliptic curve and M being a pseudo-random mapping function with key k_0. Key k_0 is the same for all columns and derived from the CryptDB proxy's master key. Column A, B and C are encrypted with their respective secret key's k_1 ,k_2 and k_3, where $k_1 \neq k_2 \neq k_3$.

In table 3 we have three columns in which value are encrypted with a function JOIN-ADJ of the same elliptic curve EC using the same secret base point G [JMV01]. Each column is encrypted using a different secret key k_x, which is a value between zero and the order of the base point on that elliptic curve. Due to these differend secret column keys equal values as 2 in column A and B and 7 in column B and C are encrypted to different ciphers. Because the server does not possess the secret key's k_1 and k_2 it is unable to perform cross-column equality checks between column A and B without the client's permission. Let's assume in this scenario that a client what's the server to search for values occurring in both column A and B (i.e. 2), without allowing the server to decrypt these columns to their plain values. The client can issue an update based on ECDH to the server generating a shared key for columns A and B based on key k_1 and k_2, without the need for server-side decryption. The issue a update of two columns with one having a key k_1 and the other key k_2 a client has to send $\Delta k = k_1 / k_2$ to the server for updating k_1's column. values encrypted with k_2 are updated using : $C_{k2}(value)^{\Delta k} = P^{k_2 * M_{k0}(value) * (k_1 / k_2)} = P^{k_1 * M_{k0}(value)} = C_{k1}(value)$. In our example this would mean that Column A and B are joined by a client sending $\Delta k = k_1 / k_2$ replacing the encryption key of column B with that of A after which both

columns share k_1 allowing for the server to observe that $C(2)_k x$ is present in both column A and B, but not C. In this example, we showed the how two secret column keys can be used to generate a shared secret key using the elliptic curve based join. This allows for a server to update encrypted values without the need for decryption, leaving all computational demanding tasks at the server as a client only has to calculate $\Delta k = k_1 / k_2$, which can be used by the server without revealing secret key k_1 or k_2. This principle also applies to the joining of more than two columns, in which joining can happen pairwise repeating the process until all columns share a secret key. CryptDB's implementation is slightly differed as it includes optimizations for both performance, security guarantees and inclusion order preservation tokens using an NIST-approved elliptic curve, as explained in by Popa and Zeldovich [PZ12] dedicate to implementing an improved adjustable join.

5.5.2 *Query-based Encryption Levels*

CryptDB implements adjustable query-based encryption, which is based in the idea that the maximum level of data confidently is related to the by the client required functionality (SQL queries). CryptDB's privacy level is dynamic and can decrease for each data item at runtime when this is needed for the execution of a particular SQL query [PRZB11b]. Queries that only need the primitive operators RND, HOM or SEARCH, are for example more secure than those that require DET, JOIN or OPE as shown in figure 6. CryptDB implements this trade-off between functionality and security by only providing the server with less secure operators when this is strictly required for the execution of a client's query. To achieve this CryptDB works with an onion layered encryption for each value, as is illustrated in 14.

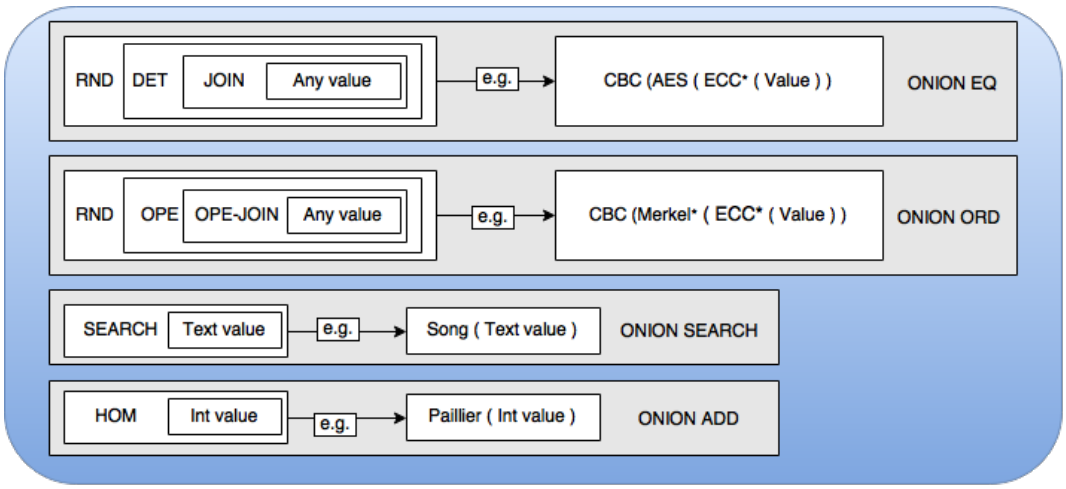

Figure 14: "The four onions used by CryptDB are based on the primitive SQL operator they support, namely Equality, Order, Search, and Addition. Each onion layer is based on the functionality and confidentiality it is required to provide an abstract level (RND, DET, OPE, JOIN, OPE-JOIN, SEARCH and HOM) [PRZB11a]. The default implementations of these unions is shown on the right and is explained in the previous section of 5.5.1. Cryptographic schemes annotated with " $*$ " represent CryptDB variants based on that scheme, not to be confused with their schoolbook example.

Each database value has an equality and order onion encrypted with the several layers of protection. The outer layer of these onions is by default RND, which provides the strongest confidentiality guarantees and the least functionality as equality checks cannot be made. When an operation is a needed that is prevented by the current level of encryption, layers are permanently removed by the client by providing the DBMS with a key to creating access. For example, a client wants to perform an equality check on columns A which currently has all values encrypted with equality onions containing the RND encryption level. RND prevents equality checks in the equality onion and permanently removed by the client leaving DET as the top layer, which will suffice for equality checks. The core principal to this process is that a client can dynamically determine whether a layer should be removed allowing for additional query functionality at the cost of confidently guarantees. Note though DET provides weaker confidentiality (encrypted values can match for equality) as RND, it deliberately does so to provide this functionality. To dynamical determine which encryption layers are to be maintained each user connects to the DBMS using a CryptDB proxy. This proxy layer keeps track of the current encryption layers of all columns and databases schema. User queries are sent through the proxy that makes each table and variable anonymous using given maximum usable encryption level. The CryptDB proxy monitors client queries and determines whether the DBMS should be allowed to adjust encryption layers. If so, it sends an additional query containing a key that allows the DBMS server to update that particular encryption layer. Using an onion structure only results in just a small computational overhead compared to the use of Postgres, because every layer only gets removed (decrypted) once and only if needed [PRZB11b].To explain

this layered encryption model further we set out several examples in section 5.5.2.1 until 5.5.2.4 capturing the main idea behind this in practical scenarios.

5.5.2.1 *Example 1 : The Database from Client- and Server-side*

In this example we will shows how the CryptDB proxy modifies a database that is created client side *(Table 4)* and changes it to an obscured variant to be stored at the server *(Table 5)*. Table 4 is labelled 'Citizens' and consist of 2 columns obtained from the TLPRD (Table 15). Column 1 contains string (text) values and Column 2 integers, this causes column 1 to get a OnionSearch and column 2 to get a OnionAdd encryption layer server side at table 5.

Table name : Citizens	
Column 1 : NameOwner	Column 2 : BSN
String value : 'Karuna Spengers'	Integer value : 176265557
String value : 'Yasemin van der Schee'	Integer value : 984840734

Table 4: SubDB : A simple DB consisting of a table labelled 'Citizens' created by a CryptDB client

Table name : 0x65BA839A (Obscured "Citizens" table name in CryptDB)							
Obscured NameOwner (C1) columns				Obscured BSN (C2) columns			
C1-IV	C1-OnionEq	C1-OnionOrder	C1-OnionSearch	C2-IV	C2-OnionEq	C2-OnionOrder	C2-OnionAdd
0xDF7	0x416	0x73B	0xFEB	0xB4F	0xA89	0xDA9	0x7C4
0x7C8	0x5B7	0x51CD	0x829	0x5B7	0xCE1	0xBFE	0xED6

Table 5: Encrypted SubDB : The by CryptDB's proxy modified version of table 4 for the DBMS server. Note that C1 is based on a string attribute and contains an Onion-Search and that C2 contains a OnionAdd because it's based on an integer attribute

Table 5 is a modified version of Table 4 in which each column has been encoded to four by the CrypDB proxy located at the client. One initialization vector (IV) and three onion encryption tables (OnionEq, OnionOrder and either OnionSearch or OnionAdd). Each of the columns encrypted by CryptDB in table 5 is encrypted with a unique private key, allowing for separate column decryption and key management. Column and table names are obscured by the CryptDB proxy that holds the key's and encryption level of each onion table. The proxy provides updates to the servers database if additional functionality is required by a client's query, as is further explained in Example 2.

5.5.2.2 *Example 2 : Query-based Encryption Update*

To illustrate the use of the onion structure, we will give a simple example using table 4 referred to as SubDB. In SubDB, we have two items that consist of a text and an integer value. We encrypt both the text and the integer values with an Equality and Order onion, which can be applied to any value. For the text values, we also implement a Search onion and for the integers values the integer specific Add onion. There haven't been any query request, so we assign the highest possible encryption layer to each onion visualized by the

top layers in figure 14. The resulting encryption table with mainly RND encryption layers is shown in 6.

	Column 1 : NameOwner	*Column 1 : NameOwner*	*Column 1 : NameOwner*	*Column 2 : BSN*	*Column 2 : BSN*	*Column 2 : BSN*
	OnionEq	OnionOrder	OnionSearch	OnionEq	OnionOrder	OnionAdd
Item 1:	RND	RND	SEARCH	RND	RND	HOM
Item 2	RND	RND	SEARCH	RND	RND	HOM

Table 6: Onion encryption Scheme shown in figure 14 applied on the SubDB of table 4. Note that the inclusion of a Search or Hom onion is depended on the column's data type

Table 4 is encrypted and stored according to the layer structure illustrated by table 6 on a DBMS server. The user now wants to search the database to know whether "Karuna Spengers" is in the Column "NameOwner". The query the client uses for this tasks is Q2.1.

$$SELECT * FROM\ SubDB\ WHERE\ 'NameOwner' = 'Karuna\ Spengers' \quad \text{(Q2.1)}$$

The CryptDB proxy receives query Q2.1 and knows that 'NameOwner' is in columns 1. Column 1 is encrypted with OnionEq at level RND and, therefore, prevents the use of equality checks. A relief of the OnionEq encryption is required to perform this search operation so the proxy send UPDATE query Q2.2 containing the RND key for that onion to the DBMS.

$$UPDATE\ SubDB\ SET\ Column1OnionEq = Decrypt\_RND\ (key, Column1OnionEq) \quad \text{(Q2.2)}$$

Update Q2.2 releases the RND encryption lowering the encryption of the equality onion to the DET encryption level, as shown in table 7.

	Column 1 : NameOwner	*Column 1 : NameOwner*	*Column 1 : NameOwner*	*Column 2 : BSN*	*Column 2 : BSN*	*Column 2 : BSN*
	OnionEq	OnionOrder	OnionSearch	OnionEq	OnionOrder	OnionAdd
Item 1:	**DET**	RND	SEARCH	RND	RND	HOM
Item 2	**DET**	RND	SEARCH	RND	RND	HOM

Table 7: Updated version of the encryption layers of table 6.

After the CryptDB updates Column1OnionEq to the DET encryption level, it can perform equality checks using the encrypted search query Q2.3.

$$SELECT * FROM\ SubDB\ WHERE\ Column1OnionEq\ = \{Karuna\ Spengers'\}_k \quad \text{(Q2.3)}$$

In query Q3, CryptDB encrypts the value 'Karuna Spengers' using the encryption key k for the Equality Onion of column 1. Columns 1 is then at a privacy level determent by the

security of the DET encryption scheme. The encryption level can be lowered further to that of the JOIN scheme, but never completely removed leaving values never in plaintext at the DBMS.

5.5.2.3 *Example 3 : Query Resulting in Equality Leakage*

In Example 2 we explained how CryptDB proxy can update union layers to release additional functionality. In this example, we illustrate how such an update affects the data base's values and causes the leakage of equality. Let's assume that a CryptBD's proxy client uses table 6 from Example 1 and updates it with a third person using query Q3.1. Note that the added person happens to have the same name (Karuna Spengers) as the first person in that table, but with a different BSN number "3473126943".

$$INSERT\ INTO\ Citizens\ (NameOwne, BSN)\ VALUES\ ('KarunaSpengers','3473126943'); \tag{Q3.1}$$

The CryptDB proxy executes query Q3.1 and modifies it using the encrypted table and column key. It then sends the encrypted query Q3.2 to the server.

$$INSERT\ INTO\ 0x65BA839A\ (C1-IV, C1-OnionEq, C1-OnionOrder, C1-OnionSearch, C2-IV, C2-OnionEq, C2-OnionOrder, C2-OnionAdd)\ VALUES\ ('0xK9E','0x3C2','0xB5D','0x42B','0xC4F','0x8AF','0xE24','0x28B'); \tag{Q3.2}$$

Query Q3.2 results in the extension of table 5 at the server, as shown in table 8.

Table name : 0x65BA839A (Obscured table name (Citizens) in CryptDB)							
Obscured NameOwner (C1) columns				Obscured BSN (C2) columns			
C1-IV	C1-OnionEq	C1-OnionOrder	C1-OnionSearch	C2-IV	C2-OnionEq	C2-OnionOrder	C2-OnionAdd
0xDF7	0x416	0x73B	0xFEB	0xB4F	0xA89	0xDA9	0x7C4
0x7C8	0x5B7	0x51C	0x829	0x5B7	0xCE1	0xBFE	0xED6
0xK9E	**0x3C2**	**0xB5D**	**0x42B**	**0xC4F**	**0x8AF**	**0xE24**	**0x28B**

Table 8: The extended version of the encrypted SubDB of table 5 before the OnionEq update of example 2. Note that row three has been added containing similar values to row 1 and that there is no similarity visible between the three rows.

In table 8 all encryption layers are their default level as shown in table 6. If we now apply the query update Q2.2 of example 2 we see updated values for column 1 at the OnionEq changing from RND to DET encryption as shown in table 9.

Table name : 0x65BA839A (Obscured "Citizens" table name in CryptDB)							
Obscured NameOwner (C1) columns				Obscured BSN (C2) columns			
C1-IV	C1-OnionEq	C1-OnionOrder	C1-OnionSearch	C2-IV	C2-OnionEq	C2-OnionOrder	C2-OnionAdd
0xDF7	**0x75C**	0x73B	0xFEB	0xB4F	0xA89	0xDA9	0x7C4
0x7C8	**0x7BF**	0x51C	0x829	0x5B7	0xCE1	0xBFE	0xED6
0xK9E	**0x75C**	0xB5D	0xFEB	0xC4F	0x8AF	0xE24	0x28B

Table 9: Updated table 8, where query Q2 has been used to lower the OnionEQ layer from RND to DET encryption. Changed values are shown in Bold format.

In table 9 we can see that all values of the OnionEq of the 'NameOwner' column (C1-OnionEq) have been modified while leaving all other columns unchanged. It is now clear the see for the server that value '0x75C' occurs multiple times in column 'C1-OnionEq' indicating equality while not releasing the original string. Note that while Equality is now leaked to the server it does so strictly to provide the functionality required by query Q2.1 while leaving it entirely unknown to the server whether or not other columns have matching values. In this example we have ('Karuna Spengers','3473126943') and ('Karuna Spengers','176265557'), for the server this is visible as ('0x75C','0xA89') and ('0x75C','0x8AF') where determining whether ($0x8AF \equiv 0x75C$) holds is a hard problem defined by RND as explained in 5.5.1.

5.5.2.4 *Example 4 : Queries that Require Homomorphic Properties*

One of the main selling points of CryptDB is that it allows for standard SQL queries over encrypted data [PRZB11a]. One of the difficulties of this is often found in queries based on both additive and multiplicative proprieties, as this breaks partial homomorphic encryption schemes. In this example, we show how CryptDB can solve queries that require both addition and multiplication. For this example, we will use a simple database consisting of one table named 'Cars' containing two columns 'Brand' and 'Catalog price', shown in table 10.

Table : Cars	
C1 : Brand	C2 : Catalog price
'Volkswagen'	40381
'Volkswagen'	10374
'Opel'	21125

Table 10: CarDB : A simple DB consisting of a table labelled 'Cars' created by a CryptDB client

The CarDB is stored at a server using the CryptDB proxy as table 11.

Table : 0x65BA839A (Obscured "Cars" table name)							
C1 : 0xCB362AA3 (Obscured "Brand" columns)				C2 : 0x7ACC5311 (Obscured "Price" columns)			
C1-IV	C1-OnionEq	C1-OnionOrder	C1-OnionSearch	C2-IV	C2-OnionEq	C2-OnionOrder	C2-OnionAdd
0xE92	0xAB9	0x534	0x92A	0x5CC	0xA89	0x8C0	0xF60
0x963	0x59C	0x54C	0x829	0x12C	0xDE5	0xEEC	0x3FB
0x0AF	0x6BF	0x609	0x036	0x620	0x6F1	0xE99	0xAD0

Table 11: Encrypted CarDB : The by CryptDB's proxy modified version of table 10 for the DBMS server

An example of queries that contain both addition and multiplication are those that contain the operation 'average'. To calculate the average of a list one needs to sum a list and divide it by the number of elements in of that list, which is equal to the inverse multiplication. If we want to know the average Catalog price' of a car in table 10 we can use the following query in the CryptDB clients proxy:

$$SELECT\ AVG\ (Catalogprice)\ AS\ average\ FROM\ Cars; \tag{Q4.1}$$

The clients proxy receives query Q4.1 and encrypts the column and table name for the DBMS server as Q4.2 were 0x7ACC5311 matches 'Price' and 0x65BA839A matches 'Cars'.

$$SELECT\ AVG\ 0x7ACC5311\ AS\ average\ FROM\ 0x65BA839A; \tag{Q4.2}$$

The DBMS server will not able to perform the AVG operation as CryptDB implemented an additive partial homomorphic instead of full homomorphic encryption for the homomorphic (Hom) layer in the ADD onion. CryptDB uses a partial homomorphic encryption scheme because full homomorphic encryption preforms to slow, as explained in section 4.2. The client is, however, able to call a UDF that performs query Q4.3 and Q4.4, returning their results to the clients proxy.

$$SELECT\ SUM\ '0x7ACC5311 - OnionAdd'\ AS\ columnsum\ FROM\ 0x65BA839A; \tag{Q4.3}$$

Query Q4.3 provides the summation of values in the 'Price' column by using the addition onion of column 0x7ACC5311 (Price). The addition onion is able to provide the the encrypted summation of table 11, without additional UPDATE functions or decryption as the OnionAdd allows for:

$$\begin{aligned} &Encrypted(40381) + Encrypted(10374) + Encrypted(21125) \equiv \\ &0xF60 + 0x3FB + 0xAD0 \ \equiv \ 0x7C1 \ \equiv Encrypted(71879) \equiv \\ &Encrypted(40381 + 10374 + 21125); \end{aligned} \tag{Q4.3}$$

Note that the DMBS server never needs to decrypt values or gains insight in the outcome of the summation. The DMBS server also calculated the number of elements of the given column, which it can do without decryption as the number is visible. Empty values can be handled by an additional count including equality checks for zero values, which can be extended by additional value calculations to obscure zero value presence in a column. In

this example, there are however no empty values, and we perform just the single count using query Q4.4.

$$COUNT\ (*)\ FROM\ '0x7ACC5311 - OnionAdd'; \qquad \text{(Q4.4)}$$

The DMBS server then returns the values of Q4.3 and Q4.4 as a result of Q4.2. Note that the outcome of Q4.4 is not encrypted as COUNT does not implement as secure encryption module.

$$SELECT\ AVG(0XE24AE)\ AS\ average\ From\ 0xB237AE = (0x7C1, 3). \qquad \text{(Q4.2 Result)}$$

The client's proxy server receives the values of Q4.3 (0x7C1) and Q4.4 (3). The proxy server is then able to calculate the average value of the 'Price' column with only one additional calculation and one decryption: decrypt(0x7C1)/3 = 23960.

5.5.3 *Performance and Query Support*

CryptDB has shown to support most queries obtained in 10 days from an MIT production (MySQL) server running on a shared web application hosting service operated by MIT's Student (sql.mit.edu) [PRZB11a]. From the 128,840 columns in the database over 99.5% percent could remain in encrypted form when processing the 126 million queries obtained from sql.mit.edu, as shown in table 12.

Application	Total columns	Columns targeted for Encryption	Not supported	Percentage Supported	Needs HOM	Needs Search	Lowest level RND	Lowest level DET	Lowest level OPE
phpBB	563	23	0	100	1	0	21	1	1
HotCRP	204	22	0	100	2	1	18	1	2
grad-apply	706	103	0	100	0	2	95	6	2
TPC-C	92	92	0	100	8	0	65	19	8
sql.mit.edu (without in-proxy processing)	128,840	128,840	1094	99.1	1,019	1,125	80,053	34,212	13,131
sql.mit.edu (with in-proxy processing)	128,840	128,840	571	99.5	1,016	1,135	84,008	35,350	8,513

Table 12: Overview of CryptDB's result on the benchmarks of application traces found in [PRZB11a]. Not supported indicates that CryptDB cannot executed the applications queries over those columns under encryption. All columns that can be used under encryption for the provided benchmark queries are categorized by their lowest used encryption layer steady-state.

In table 12 we can see that the most data (at least 99.5%) columns can be encrypted using CryptDB under the assumption of allowed in proxy processing. In-proxy processing allows CryptDB to evaluate predicates at the proxy instead of the server as this can provide higher security guarantees in some (nested) cases. In this table, we can see that for actual server data of sql.mit.edu more than 60% of the columns could remain as an RND encryption layer providing maximum confidentiality guarantees if HOM and SEARCH are not included. The weakest security layer of OPE as for all tested application traces not required for over 90% of the encrypted columns showing that the inclusion of stronger confidently protecting encryption onion layers is still useful in a high (126 million) query

traffic scenario. Most queries in this benchmark consisted of simple operators like insert, update, delete, nested queries, indexes and transactions that proved supported by CryptDB under encryption. Queries based on complex operators like trigonometry and operations that require combining incompatible encryption schemes (e.g. $(C_1(a) + C_1(b) > C_2(c))$) are not supported [PRZB11a].

Popa et al. [PRZB11a] performed a benchmark comparison of an unmodified MySQL server against the same setup with a CryptDB proxy as front-end. The benchmark was done using the TPC-C described at table 12 without additional proxy pressing and no active encryption layer adjustments. These setting are justifiable given that the decryption of columns is only done once per layer as CryptDB only allows for the decrease in security of layers limiting the encryption overhead. The average overhead in throughput of CryptDB compared to the unmodified Mysql server is between the 21 and 26 percent depending on the amount of server cores.

5.5.4 *Limitations*

CryptDB's encryption model is based on the combination of multiple well-known encryption schemes providing it with the capabilities to handle SQL operations on encrypted data is shown in section 5.5.3. CryptDB's functionality does come at a cost in both terms of performance, query support and compatibilities compared cryptographic MySQL solutions.

QUERY SUPPORT CryptDB does not have support for analytical queries over encrypted data [PRZB11a] and cannot work with complex operators like trigonometry or those that required the combing of incompatible encryption schemes as earlier mentioned in section 5.5.3. CryptDB does not provide an exact list of which queries are supported, and it is important to check this before considering CryptDB as a solution.

PERFORMANCE CryptDB can process queries with a relatively small overhead in terms of query throughput. This overhead is in the order of 25% as described in section 5.5.3. Overall latency can however be 6 times higher in the CryptDB proxy ($0.60ms$) compared to the MySQL server ($0.10ms$) as CryptDB requires its proxy to perform a notable amount of parsing and processing [PRZB11a]. It is, therefore, necessary for a CryptDB client to possess more processing power then when no proxy is used, leading to higher client requirements. Additionally it should be taken into consideration that new queries provided to a server might trigger a decryption of union layers, as explained in section 5.5.2.2, which will cause a temporary latency at the server depended on the number of elements in the affected columns.

SECURITY CryptDB does not provide guarantees regarding the integrity, freshness, or completeness as its main focus is based confidentiality. This means that CryptDB requires a "untrusted" server to be trusted with these aspects of security [PZB11].

SUPPORT CryptDB is developed as a proof of concept at MIT and does not provide a broad range of support. It latest update was on 5 Feb 2014 [oT15] and is only compatible with Ubuntu Linux LTS 12.04.x 64bit and MySQL 5.5.14. I found that updating this

version of CryptDB (or any previously released), MySQL or the use of a general update command (apt-get update) will cause CryptDB to stop working and require a clean install. CryptDB's comes with no official support, but does have a mailing list for users "cryptdb-users@lists.csail.mit.edu" and hold an archive of provided answers "https://lists.csail.mit.edu/mailman/listinfo/cryptdb-users" [oT15]. From the end of 2013 till June 2015 over 80 Questions question have been submitted with only 25 of them (partially) answered by community members or CryptDB's developers. On a site [whia] explaining the installation of CryptDB's latest version are also comments on over 14 different installations and setup issues regarding CryptDB that are not or partially solved. Personally I also encounter difficulties setting up CryptDB encountering update and VM incompatibilities, crashes and modules that are included, but not operational or updated since more than 2 years, like the web-interface and training mode. CryptDB is currently still in a development stadium containing several unsolved implementation related issues. CryptDB hasn't received any updates in its Github repository since begin 2014 [oT15] raising the question if active support for CryptDB is still present.

Part IV

REQUIREMENTS

6

CASE DESCRIPTION

Current and future technology allows for Intelligent Transport Systems (ITS). Intelligent Transport Systems can improve the safety of traveling by providing feedback about the environment, traveling ease by preventing traffic jams or making toll systems easier accessible. The amount of vehicle that include ITS like vehicle to vehicle (V2V) and Vehicle-to-Infrastructure (V2I) are expected to increase in the Netherlands, with the Dutch government promoting ITS development and presenting itself as test country for self-driving vehicles [zv15].

The Dutch Motor Vehicle Authority RijkDienstWegverkeer (RDW) collects and manages data on vehicles in a Database on License Plate Registration (DLPR). Due to increasing developments in ITS and other innovations in future smart cars RDW consider that the amount of requests to the DLPR may significantly increase in size in the near future [DLS+13]. Cloud providers offer scalable and secure data solutions, and the RDW wants to know what the effects would be if they were to move their DLPR to the cloud. Because parts of the data regarding license plate registration of Dutch citizens is considered personal information by Dutch law, RDW is required to keep those parts hidden from the cloud provider [Per; Rij14]. Encryption tools can be a solution to ensure data privacy, but these come at a cost of efficiency or availability [CLHK11]. One of the most important aspects in this scenario is that a cloud-stored database still has to be usable by the RDW and other authorized parties while not leaking private data to the cloud provider, as set out in the sections 6.1 Access, 6.2 Actors and 6.3 The Database. To define the technical criteria for such a model we set out use cases obtained from interviews with the RDW, which can be found in section 6.4 Use-cases.

6.1 ACCESS

In the current situation, the RDW hosts the DLPR on a private server and manages all the data access of other parties. Actors can acquire public data from the DLPR in an automated process by accessing the RDW's homepage and entering a registered license plate number [RDWa], this will be referred to as a "Public Query". To acquire or modify private data from the DLPR actors have to send a data request with identification to the RDW [RDWa], this will be referred to a "Private Query". For Dutch citizens this can be done by using their digital passport DigiD [Kal09] or mail, this will be referred to as "Special Query" or "Special Request". A subset of the license plate registration database of 2012 that exclusively contains

non-private data has been made publicly available on the Azure cloud and is referred to as the Azure-DLPR [RDWb]. In table 13 we provide an overview the three different type of query classifications and the desired processing party according to our case description of section 6. This shows that only Private Queries are moved from being processed by the RDW to be processed in the Cloud as Public Queries are already cloud based, and Special Queries require access to private information in plaintext. These private queries are defined in the Use-cases of section 6.4.1 and are explicitly covered by our solution design in section 7.2.

Type of Query	**Current Processor**	**Desired Processor**	**Explanation**
Public	Cloud	Cloud	Public queries are queries that exclusively involve public attributes.
Private	RDW	Cloud	Private Queries are queries that involve 1 or more private attribute. See the Use-cases of section 6.4.1 for a full overview of all included cases.
Special	RDW	RDW	Special queries are queries that require the active involvement of the RDW.

Table 13: Overview of the three different type of queries included in this thesis as used in our scenario models of figure 15 and 16.

6.1.1 *Current Scenario*

To simplify the current situation, we assume that both the Azure-DLPR and the non-sensitive data request on the DLPR are handled by the same database and in the cloud, as there are no privacy restrictions on this data. This cloud hosted database will be referred to as the Public Subset of Database on License Plate Registration (PS-DLPR), which is a subset of the DLPR containing only non-private data. Data request regarding private data is in this scenario still send to the RDW as is currently the case. This simplified version of the current situation of data-storage including the cloud is illustrated figure 15.

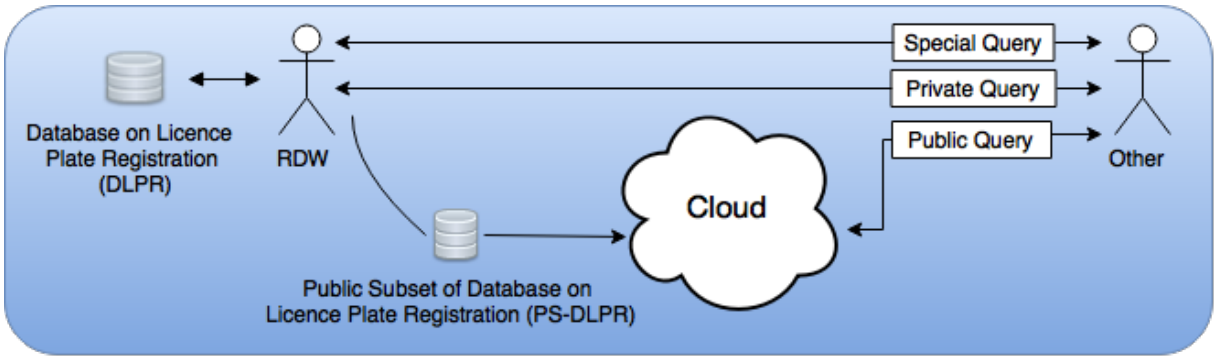

Figure 15: The current Hybrid-Cloud-Scenario : Full DB (DLPR) at RDW & Public subset (PS-DLPR) at the Cloud

6.1.2 *Public Cloud Scenario*

If the RDW would move its DLPR database to the cloud by contracting a cloud provider, a new scenario would arise in which an actor other than the RDW has a level of access to the private attributes within the DLPR. The cloud provider is not allowed to read private attributes within the DLPR without permission. To enforce the confidentiality of private data the RDW applies encryption on the DLPR before moving it to the cloud. The cloud provider than hosts the DLPR database and manages the access to it by other actors according to privacy restrictions for those organisations set by the RDW. The RDW distributes key's to all actors allowing them to read or write data they have been granted access to. Direct access to the cloud is only allowed for predefined public and private query request. Special query request like those that require verification are still sent to the RDW as illustrated in figure 16.

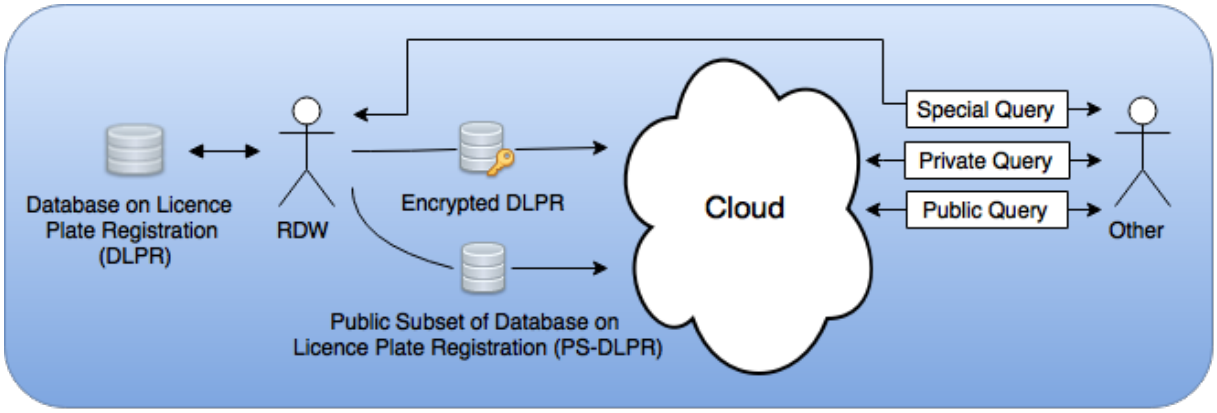

Figure 16: Public-Cloud-Scenario : Full DLPR including sensitive information in encrypted format stored in the cloud allowing for encrypted private queries. In this scenario the PS-DLPR remains hosted in the cloud in order supported the plain public queries.

6.2 ACTORS

The actor defined by others in figure 15 and 16 represents a group of actors that together with the RDW form all actors that require access to the DLPR. All actors combined provided almost 40 percent more data requests than in 2012 resulting in 2.548.500.000 online and batch data requests to the RDW in 2013 [Weg13]. The most request came from the Police (59.4 %), which requested 59.4 percent of all online information request as is illustrated in figure 17. The other data requesting actors with unique and significant amount data request we have taken into account in our use cases are : citizens (7.4%), insurance companies (2.0%) and the tax authority (0.4%).

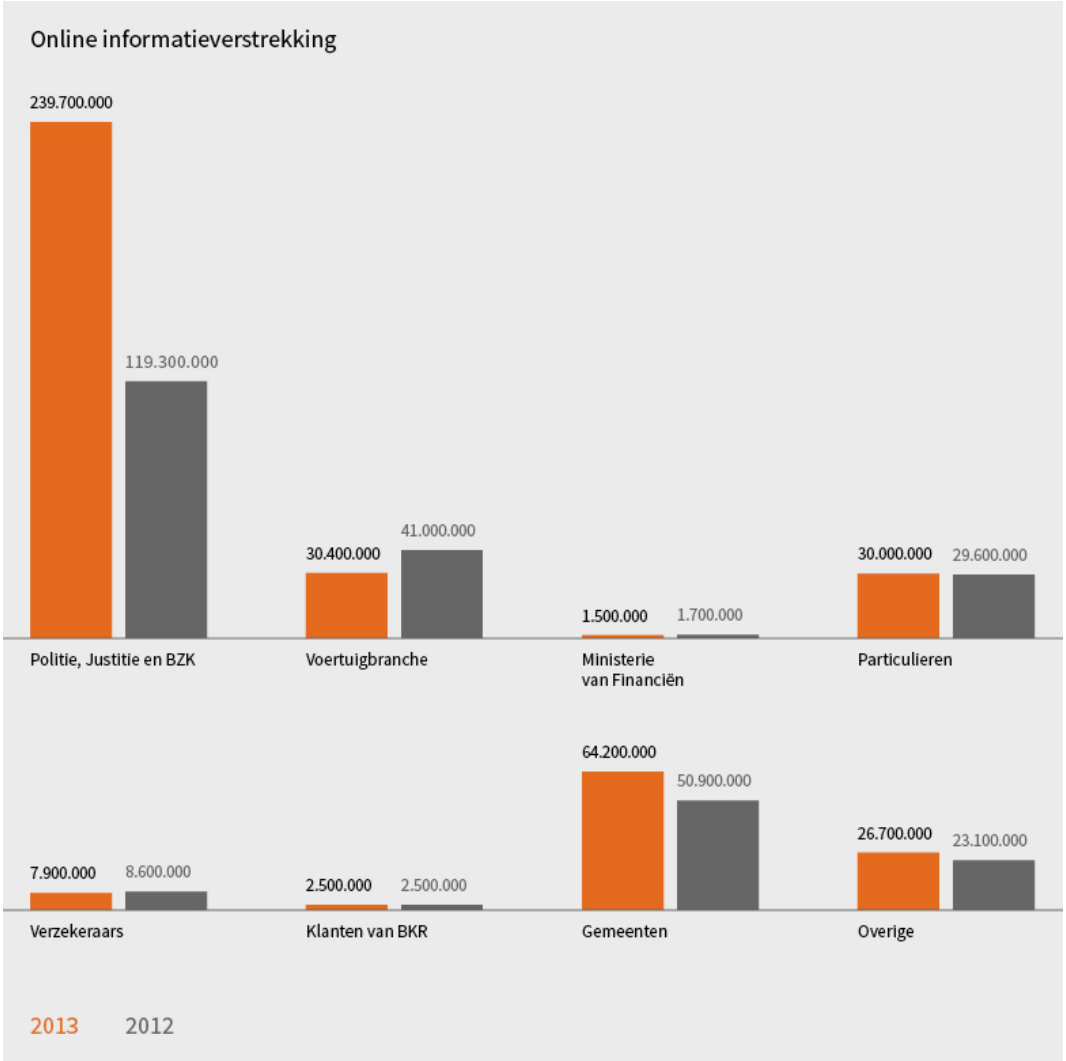

Figure 17: Overview of the number of individual data requests send to the RDW by different parties [Weg13]

The actors included in our research exists of the previously mentioned police and citizens combined with the Dutch motor vehicle authority (RDW) and the cloud provider hosting the database. Here follows a short list of all these actors and their role in the public cloud solution, as described in the scenario in figure 16.

THE RIJKDIENSTWEGVERKEER(RDW)
: The Dutch motor vehicle authority requires full access to the complete DLPR database to use and update it. It should also be possible for the RDW to assign, limit and revoke access to all or some attributes within the database for individual stakeholders.

CLOUD PROVIDER
: The cloud provider where the database is stored should not be able to compromise the confidentiality of private information stored or requested by others actors.

THE POLICE

The Dutch police are required to have read access to attributes belonging to a specific individual, based on license plate number. The police are also required to be able to update a car's legal status attribute given a license plate number.

CITIZENS

Citizens are allowed to retrieve all public information corresponding to a specific license plate number. Citizens are also allowed to retrieve private information registered about their car. To retrieve private information it is required for citizen to authenticate themselves by means of an electronic ID called DigiD or by sending a paper request to the RDW, which contains their BSN, name, address, date of birth and license plate number [RDWa]. The RDW retrieves the information for them by crosschecking private data which causes these type of request to be labelled as a Non-Standard Data Request as they can not be handled by a party other than the RDW.

6.3 DATABASE

In section 6.1: Access, we described that the DLPR consists of a private and a public part that can be found in the Azure-DLPR and PS-DLPR. The private part is not public available and is restricted according to Dutch law as explained in section 6.3.1: Sensitive Data. Because the DLPR contains private attributes and consists of several dozen attributes we introduce an experiment database in section 6.3.2 that will be used in the experimental setups of this thesis.

6.3.1 *Sensitive Data*

The private part of the DLPR database contains information that is not freely accessible as it can compromise personal information by itself or in combination with other obtainable information. The RDW applies the followings Dutch laws in order to classify which attributes are classified as sensitive and, therefore, have restricted access, article 42a of the "De Wegenverkeerswet 1994", article 7 of the "Kentekenreglement" and article 1 of the "Wet bescherming persoonsgegevens" (Wbp) [Rij14]. The Information within the license plate database classified as sensitive are the following:

- "Elk gegeven betreffende een geidentificeerde of identificeerbare natuurlijke persoon"[Rij14]. All information regarding to an identified or identifiable person. This includes for example a person's full name, address and social service number.
- "Gegevens waarvan de verstrekking een nadelig effect kan hebben op de concurrentiepositie van een onderneming, waaronder in elk geval worden verstaan voertuigidentificerende gegevens in combinatie met gegevens ten aanzien van rechtspersoon omtrent:"[Rij14]. Information that when released may have an adverse effect on the competitiveness of an enterprise. This includes but is not limited to vehicle identifying information in combination with information regarding the legal person:
  - Name, address and domicile.

  - Date of establishment and abolition of a legal person.
  - Legal person related numbers and coding.
  - Legal status.

- "Gegevens waarvan de verstrekking het risico van handelen in strijd met een wettelijk voorschrift met zich brengt"[Rij14]. Information that when released may risk acting in conflict with a statutory regulation. This includes but is not limited to information regarding:
  - Identification and registration a vehicle.
  - Theft of a vehicle.
  - Liability regarding the vehicle.
  - License plate certificate.
  - Vehicles status.

6.3.2 *Experimental Database*

The complete database contains over 50 attributes and has over the 10 million entries [Mar14]. Most of these categories and entries are superfluous when investigating the cryptographic abilities of this database. To create a smaller, better comprehensibly database we introduce the sampled Toy License Plate Registration Databases (TLPRD) in our research examples. The TLPRD database contains fabricated public and private values for interesting attributes and will resemble the original database in format and challenges.

The TLPRD contains seven attributes of the License Plate Registration Databases. The attributes included in the TLPRD are labelled Private if they are considered sensitive data and are labelled public otherwise. Public information is non-sensitive data and can be retrieved from the RDW by anybody based on a license plate request [Weg14]. The freely interpreted DLPR attributes included in the TLPRD database are:

Category	**Data type**	**Classification**	**Explanation**
LicensePlateNumber	String	Public	License Plate sign combination.
NameOwner	String	Private	Name of the current owner of the vehicle.
BSN	Int	Private	BSN stands for Burger Service Number and is the national identification number of a Dutch citizen.
APKDate	String	Public	Date until the APK validity expires.
Colour	String	Public	First colour registered for the car.
Brand	String	Public	Identifying mark of producers of the car.
CatalogPrice	Int	Public	Registered value of the car.
CO2Emission	Int	Public	Weighted CO_2 emission in gram/kilometre for that model.
Status	String	Public	Insurance relevant status of the car.
Legal	String	Public	The legal status of the car that is either "STOLEN" or "OK".

Table 14: Attributes included in the Toy License Plate Registration Databases

The TLPRD database exists of 25 entries and solely contains fabricated data that should in no way be linked to real persons or property. Besides random data, we also implemented 3 data constructions to support the explanation of confidentiality requirements (CR), to which we will refer in section 6.5.4. These three construction can be found in the TLPRD table 15 and are as follows:

1. CR2-Vertical: Jan Jansen is a value in $NameOwner$ (Private) that occurs multiple times in the TLPRD referring to different people according to their BSN (Private) number.
2. CR2-Horizontal: Alen Dooper (Private) is a person who is into unique cars. His car is a Golden (Public) KIA (Public) and is the only one in our data set with that particular set of public attributes.
3. CR2-Indirect Horizontal: The attributes Status (Public) and Legal (Public) are unique public attributes because the small entropy of their values as the majority (50%+) is "OK".

NameOwner	BSN	LicensePlateNumber	CatalogPrice	Colour	Brand	CO2Emission	APKDate	Status[3]	Legal[3]
Karuna Sprengers	176265557	00VFR7	40381	GREY	VOLKSWAGEN	213	2016-07-25T00:00:00	OK	OK
Yasemin van der Scheer	984840734	01VBS5	10374	GREY	VOLKSWAGEN	139	2016-06-19T00:00:00	OK	OK
Ara Warmerdam	74198658	01BXVH	21124	BEIGE	SKODA	165	2016-06-11T00:00:00	OK	OK
Donald Hoogstad	616998144	00WJ09	16444	GREY	OPEL	137	2015-10-11T00:00:00	OK	THEFT
Yoeri Kors	178173290	01BHBN	29260	GREY	TOYOTA	173	2015-08-19T00:00:00	OK	OK
JanJansen[1]	**755686308**[1]	02BVSR	16639	GREY	OPEL	137	2016-08-16T00:00:00	OK	OK
JanJansen[1]	**133451605**[1]	0057WE	11835	GREEN	VOLKSWAGEN	129	2015-08-28T00:00:00	OK	OK
JanJansen[1]	**206185297**[1]	02BJPK	12510	BLACK	PEUGEOT	109	2016-06-06T00:00:00	OK	OK
Bastien Peerdeman	196917050	02BLGG	30678	BLACK	PEUGEOT	192	2016-07-08T00:00:00	OK	OK
AlenDooper[2]	956454768	01BNBL	9894	**Gold**[2]	**KIA**[2]	120	2016-05-27T00:00:00	OK	THEFT
Ted Holster	555724335	02BPDT	11835	RED	FIAT	129	2016-03-05T00:00:00	OK	THEFT
Donald Hoogstad	616998144	02BPSF	19185	BLUE	POLO	176	2014-11-10T00:00:00	APK EXPIRED	OK
Shilan Rorije	722643149	00VGH5	27310	N.v.t.	VOLKSWAGEN	109	2016-05-27T00:00:00	OK	OK
Philipp van den Muijsenberg	669048211	00VGS7	26315	BLUE	VOLKSWAGEN	130	2014-10-30T00:00:00	OK	OK
Tetske van der Wijk	751753761	*00VJT5*	*11835*	*BLUE*	*SKODA*	*145*	*2017-10-17T00:00:00*	*OK*	*OK*
Tetske van der Wijk	751753761	*02BRRK*	*11540*	*RED*	*SEAT*	*110*	*2017-02-05T00:00:00*	*OK*	*OK*
Tetske van der Wijk	751753761	*02BRRX*	*50972*	*BLUE*	*TOYOTA*	*110*	*2015-10-11T00:00:00*	*APK EXPIRED*	*OK*
Justian Kreeft	145378295	02BVDB	28856	RED	FORD	110	2018-08-27T00:00:00	OK	OK
Randa Snellen	647299239	01WK44	28860	GREY	LAND ROVER	114	2016-03-05T00:00:00	OK	OK
Caterina Kolk	316485334	01WL12	11550	N.v.t.	SUZUKI	99	2015-06-02T00:00:00	OK	OK
Mellany van Doleweerd	308719527	0287WH	11835	RED	VOLKSWAGEN	99	2017-06-16T00:00:00	OK	THEFT
Lein Herlaar	651586777	0059WE	87707	GREY	PEUGEOT	134	2014-11-21T00:00:00	APK EXPIRED	OK
Viola van 't Klooster	378297942	0065WE	40329	PAARS	VOLVO	148	2017-02-19T00:00:00	OK	OK
Dara Haarman	777707548	0057WT	22139	GROEN	OPEL	178	2016-05-02T00:00:00	OK	OK
Cleo Bosland	846917529	0059WT	23450	ZWART	CITROEN	198	2015-08-09T00:00:00	OK	OK

Table 15: **Toy License Plate Registration Databases (TLPRD)**. Variables labels with 1,2 and 3 are explicitly used for our explanatory purposes as explained in section 6.3.2.

6.4 FUNCTIONAL REQUIREMENTS

6.4.1 *Functional Requirements From Use-Cases*

In this section, we set out the functional requirements for our solution design of section 7.2. We constructed these requirements at the hand of use cases obtained by interviews with the RDW and our interpretation of relevant database functionality. These use cases cover the basic functionality required to perform a search, update or computational query on an encrypted database using a proxy server. The four use cases from which we derived our functional requirements can be found in table 16.

Use Case	**Functionality**	**Importance**	**Motivation**
1	Search (Section 6.4.2)	Very High	Performing a search operation allows for targeting specific entries and is crucial for functionality of use cases 2, 3 and 4. In this use case we set out the following scenario: A license plate number is send to the database (TLPPRD) and corresponding personal information ($NameOwner$ and BSN) is retrieved.
2	Update (Section 6.4.3)	High	Update functionality allows for a database to be kept up-to-date with events and we consider this to be one of the most important functionalities of a database. In this use case we set out the following scenario : If a car ($LicensePlateNumber$) changes owner, new personal data ($NameOwner$ and BSN) have to be assigned to that cars in that database (TLPRD).
3	Summation (Section 6.4.4)	Low	The ability to perform summations over integer values allows for some calculations to be performed on the database. In this use case we assume that for explanatory purposes there is need to calculate the sum of the $CatalogPrice$ values of all cars registered to a specific BSN number. We include this use case for expansionary purposes and therefore classified it's importance as Low.
4	Average (Section 6.4.5)	Low	The ability to calculate the average of a set of integer values in the database allows an extension on use case 3. In this use case we assume that for explanatory purposes there is need to calculate the sum of the $CO2Emission$ values of all cars registered to a specific BSN number. We include this use case for expansionary purposes and therefore classified it's importance as Low.

Table 16: Overview of included use cases

We set out each use case in 10 functional requirements, covering all aspects regarding the desired query formats, (Proxy and Cloud) server functionality and time constraints to achieve a practical solution. In figure 18 we explain this construction at the hand of an overview.

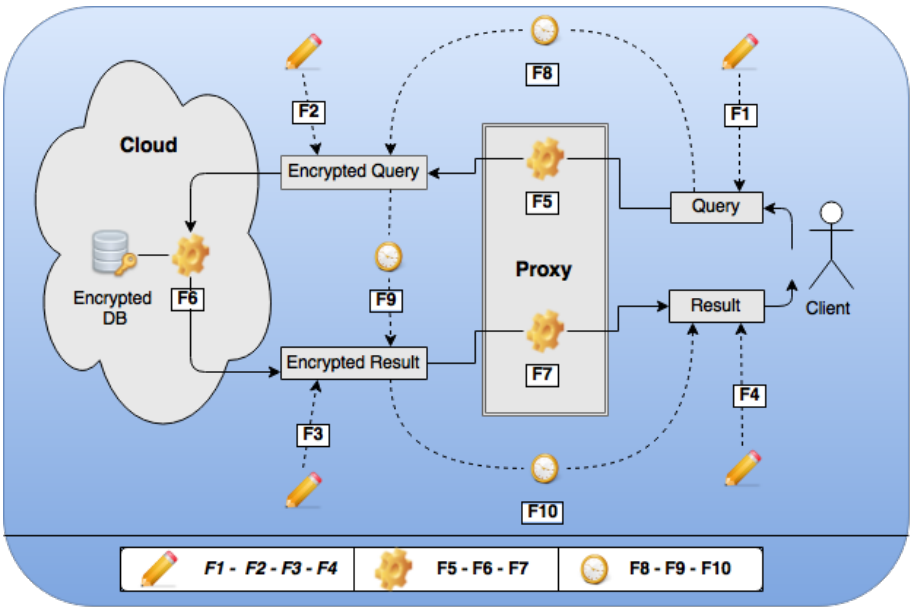

Figure 18: Overview of how all (F1-F10) functional requirements are derived for each use case. Functional Requirements F1-F4 define the formats in which queries and their results are required to be delivered (Pencil illustration). Functional Requirements F5-F7 define the processes required by the cloud or proxy in order to transform a query of format F1-F3 to a result in format F2-F4 (Gearwheel illustration). Functional requirements F8-F10 define the time constrains for processes F5-F7 (Clock illustration).

Each use case assumes an encrypted TLPRD stored and processed in the (untrusted) cloud, which gets addressed (indirectly) by a client using MySQL from a trusted environment. The Proxy server is assumed to possess all secret key's and capacities needed to effectively interact with the encrypted TLPRD. Each functional requirements as set out in figure 18 is defined in table 17 .

Functional Requirement	**Scope**	**Description**
F1 : Query	Format	The format in which a Client must send a MySQL query to the Proxy.
F2: Encrypted Query	Format	The format in which the Proxy must relay the MySQL query of F1 to the Cloud.
F3: Encrypted Result	Format	The format in which the Cloud must send the result of the MySQL query of F2 back to the Proxy.
F4: Result	Format	The MySQL format in which the Proxy must relay the query of F3 to the Client.
F5: Proxy Encryption	Functionality	The Functionality required from the Proxy in order to generate the query from F2 based on input of query F1.
F6: DB Functionality	Functionality	The Functionality required from the encrypted database (without decryption) by the Cloud in order to generate the query from F3 based on the input of query F2.
F7: Proxy Decryption	Functionality	The Functionality required from the Proxy in order to generate the query from F4 based on input of query F3.
F8: Proxy Encryption Time	Time	The time constrain(s) for the required functionality of F5.
F9: DB Functionality Time	Time	The time constrain(s) for the required functionality of F6.
F10: Proxy Decryption Time	Time	The time constrain(s) for the required functionality of F7.

Table 17: Overview of how we define the functional requirements of figure 18 for each use case.

6.4.2 *Use Case 1 : Performing a Database Search*

There are three specific search operation that this use case requires to be functional on the TLPRD, illustrate in the format (Search-criteria $\rightarrow$ required result).

1. $Licenseplatenumber \rightarrow$ List($NameOwner, BSN$).
2. $BSN \rightarrow$ List($CatalogPrice$).
   (Needed for use case 3 in section 6.4.4)
3. $BSN \rightarrow$ List($CO2Emission$).
   (Needed for use case 4 in section 6.4.5)

$LicensePlateNumber$ search is the most performed operation on RDW's license plate database. In this search a license plate number is send to the database and corresponding personal data ($NameOwner$ and BSN) is retrieved. Additionally we included BSN search in support of the functionally of use cases 3 and 4. In those use cases a BSN value is send and all corresponding BSN entries return their $CatalogPrice$ and $CO2Emission$ values to the cloud provider for further processing. We define the functional requirements for this three functions at the hand of our functional requirement model of section 6.4.1. Each requirement is encoded with U1-FX for Use Case 1, where X is the number referring to the type of requirement found in table 17.

U1-F1: QUERY FORMAT

A client sends search queries to the Proxy server in the format :

```
SELECT * FROM DBTable WHERE Attribute = Target
```

Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constraints.

U1-F2: ENCRYPTED QUERY FORMAT

The Proxy sends search queries to the Cloud server in the format :

```
SELECT * FROM DBTable WHERE C(Attribute) = C(Target)
```

Where C(x) annotates a cipher generated by an encryption scheme C with input x.

Clarification: We desire that Cloud can use MySQL syntax as this provides clear compatibility constraints.

U1-F3: ENCRYPTED RESULT FORMAT

The Cloud sends search query results to the Proxy server as a list of database rows containing all encrypted attributes

Clarification: We request complete rows instead of only attributes of interested to limit information leakage to the Cloud.

U1-F4: RESULT FORMAT

The Proxy sends search query results to the Client as a list of database rows containing only the following attributes in plaintext: $NameOwner$, BSN, $CatalogPrice$ and $CO2Emission$.

Clarification: The client only requests information from $NameOwner$, BSN, $CatalogPrice$ and $CO2Emission$ attributes. Other values can remain encrypted and will be discarded.

U1-F5: PROXY ENCRYPTION

The proxy is required to be able to encrypt the fallowing attributes :

1. $LicensePlateNumber \rightarrow$ C($LicensePlateNumber$).
2. $BSN \rightarrow$ C(BSN).

Clarification: The attributes $LicensePlateNumber$ and BSN require the same encryption schemes C and key as used in the encryption TLPRD. It is not necessary for $LicensePlateNumber$ and BSN to use the same C and key, as these are attribute depended.

U1-F6: DB FUNCTIONALITY

The encrypted DB is required to support equality checks on encrypted data (e.g. deterministic encryption) for the following encrypted attributes :

1. Proxy C($LicensePlateNumber$) == Cloud C($LicensePlateNumber$).
2. Proxy C(BSN) == Cloud C(BSN).

Clarification: The Cloud should be able to determine if an encrypted BSN and $LicensePlateNumber$ obtained from F5 are equal to any encrypted BSN or $LicensePlateNumber$ value stored in the encrypted DB.

U1-F7: PROXY DECRYPTION

The proxy is required to be able to decrypt the fallowing attributes :

1. C($NameOwner$) $\rightarrow$ $NameOwner$.
2. C(BSN) $\rightarrow$ BSN.
3. C($CatalogPrice$) $\rightarrow$ $CatalogPrice$.
4. C($CO2Emission$) $\rightarrow$ $CO2Emission$.

Clarification: Proxy should be able to decrypt the encrypted $NameOwner$, BSN, $CatalogPrice$ and $CO2Emission$ values, so the client can directly process results and does not require a decryption key.

U1-F8: PROXY ENCRYPTION TIME

Each encryption in F5 must be performed in under 0.1 seconds. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess sufficient processing power and RAM. See table 21 on page 109 for hardware specifications.

Clarification: This requirement's upper-bound in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

U1-F9: DB FUNCTIONALITY TIME

Each operation in U1-F6 must be performed in linear time to the number of database rows.

Clarification: A database other than the TLPRD (e.g. LPRD) can consist of several million rows. We deem it required that search queries can be performed on large databases without an expensive time complexity.

U1-F10: PROXY DECRYPTION TIME

Each decryption in U1-F7 must be performed in under 1 second. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possesses at most 6GB RAM and processing power in the order of a Duo CPU 3.00GHz processor.

Clarification: These requirement's upper-bound in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

6.4.3 *Use Case 2 : Performing a Database Update*

An important operation performed on the RDW's license plate database is that of updating exciting attributes. If a car ($LicensePlateNumber$) changes owner new personal data ($NameOwner$ and BSN) have to be assigned to that cars data in the (encrypted) TLPRD. We define the update functionality for the at the hand of our functional requirement model of section 6.4.1. Each requirement is encoded with U2-FX for Use Case 2, where X is the number referring to the type of requirement found in table 17.

U2-F1: QUERY FORMAT

A client sends update queries to the Proxy server in the format :

```
UPDATE DBTable SET Attribute_1 = Target_1, Attribute_2 = Target_2,
WHERE Condition;
```

General update query where Attribute equals a database column in TLPRD and Target a variable in that column.

Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constraints.

U2-F2: ENCRYPTED QUERY FORMAT

The Proxy sends update queries to the Cloud server in the format :

```
UPDATE DBTable SET C($BSN$) = c(Target_1), C($NameOwner$) = C(Target_2),
WHERE Condition;
```

Where C(x) annotates a cipher generated by an encryption scheme C with input x. Condition is a search criteria on $LicensePlateNumber$, which implies the functionality of Use Case 1.

Clarification: We desire that Cloud can use MySQL syntax as this provides clear compatibility constraints.

U2-F3: ENCRYPTED RESULT FORMAT

Not applicable in use case 2, because an update query is not required to send a result to the proxy server.

U2-F4: RESULT FORMAT

Not applicable in use case 2, because an update query is not required to send a result to the client.

U2-F5: PROXY ENCRYPTION

The proxy is required to be able to encrypt the following attributes :

1. $LicensePlateNumber \rightarrow$ C($LicensePlateNumber$).
2. $BSN \rightarrow$ C(BSN).
3. $LicensePlateNumber \rightarrow$ C($LicensePlateNumber$).

Clarification: The attributes $LicensePlateNumber$, BSN and $NameOwner$ require the same encryption scheme C and key(s) as used in the encryption of the TLPRD. It is not required for $LicensePlateNumber$, BSN and $NameOwner$ to use the same C and key(s), as these are attribute depended.

U2-F6: DB FUNCTIONALITY

The encrypted $NameOwner$ and BSN columns must support the replacement of individual values without breaking encryption or decryption assumptions of other attributes or values.

1. Cloud C($NameOwner$) $\rightarrow$ Proxy C($NameOwner$).

2. Cloud C(BSN) $\rightarrow$ Proxy C(BSN).

Clarification: Encryption assumption of other attributes might also not be affected as this might cause updates to prevent correct processing of other values. An example can occur in CBC. CBC produces a depended chain in which a missing/changed value might negatively affect the ability of other blocks to decrypt correctly.

U2-F7: PROXY DECRYPTION

Not applicable in use case 2, because an update query is not required to send a result to the proxy.

U2-F8: PROXY ENCRYPTION

Each encryption in F5 must be performed in under 0.1 seconds. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess a sufficient processing power and RAM. See table 21 on page 109 for hardware specifications.

Clarification: This requirement's upper-bound in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

U2-F9: DB FUNCTIONALITY TIME

Each operation in U2-F6 must be performed in linear time to the number of database rows.

Clarification: A database other than the TLPRD (e.g. LPRD) can consist of several million rows. We deem it required that update queries can be performed on large databases without an expensive time complexity.

U2-F10: PROXY DECRYPTION

Not applicable in use case 2, because an update query is not required to send a result for decryption to the proxy.

6.4.4 *Use Case 3 : Calculating a Summation*

An operation that can be performed on the RDW's license plate database is the calculation of the sum of a set of values. Currently, this functionality is less important, but with the future inclusion of sensor data (Section 1.1) this might become an interesting property. In this use case, we assume that for explanatory purposes there is the need to calculate the sum of the $CatalogPrice$ values of all cars registered to a particular BSN number. We define this functionality at the hand of our functional requirement model of section 6.4.1. Each requirement is encoded with U3-FX for Use Case 3, where X is the number referring to the type of requirement found in table 17.

U3-F1: QUERY FORMAT

A client sends a summation query to the Proxy server in the format :

```
SELECT SUM(Attribute) AS "Sum" FROM DBTable WHERE Condition;
```

General MySQL summation query where Attribute equals a database column in TLPRD and Target a variable in that column [Mys].

Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constraints.

U3-F2: ENCRYPTED QUERY FORMAT

The Proxy sends summation queries to the Cloud server in the format :

```
SELECT UDF(C(CatalogPrice)) AS "Sum" FROM TLPRD WHERE Condition;
```

Where UDF is a user defined MySQL function that should operate on C($CatalogPrice$) as SUM would on $CatalogPrice$. C(x) annotates a cipher generated by an encryption scheme C with input x. And Condition is a search criteria on BSN, which implies the functionality of Use Case 1. *Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constrains.*

U3-F3: ENCRYPTED RESULT FORMAT

The Cloud sends the summation query results to the Proxy server as an encrypted value in the format C($CatalogPrice$).

Clarification: We require the result encrypted in the same format (same key, encryption scheme) as the $CatalogPrice$ in the TLPRD, as this allows results to be used for further calculations.

U3-F4: RESULT FORMAT

The Proxy sends summation query results to the Client as a $CatalogPrice$ in plaintext.

Clarification: The client requests the sum of multiple $CatalogPrice$ values. It is therefore assumed that returning the result in the same format allows the client to process this result further without requiring additional information.

U3-F5: PROXY ENCRYPTION

The Proxy server is required to be able to transform a query of format U3-F1 to a query of format U3-F2. This requires the Proxy server to be able to generate a MySQL UDF that can provide the functionality of SUM under the encryption model used for $CatalogPrice$.

Clarification: A UDF is required instead of SUM because encrypting additive homomorphic encryption schemes like Pallier [Pai99] require additions to be performed as multiplications. Note there is by default no multiplicative variant of SUM in MySQL [Mys].

U3-F6: DB FUNCTIONALITY

The encrypted $CatalogPrice$ column must support the additive homomorphic property (section..) to allow the Cloud server to perform summation without requiring a secret key. (C($CatalogPrice$), ..., C($CatalogPrice$)) $\rightarrow$
C($CatalogPrice$ + ... + $CatalogPrice$). *Clarification: We require the cloud provider to perform all computationally demanding operations. It is therefore required that a cloud provider can perform summations without computational support from the proxy server or the need to posses the decryption key.*

U3-F7: PROXY DECRYPTION

The proxy is required to be able to decrypt the fallowing attributes :

1. $C(CatalogPrice) \rightarrow CatalogPrice$.

Clarification: Proxy should be able to decrypt the encrypted $CatalogPrice$ values as the result of a summation of encrypted $CatalogPrice$ values is in the same format. We desire proxy side decryption so that a client is able to directly process results and does not require a decryption key.

U3-F8: PROXY ENCRYPTION TIME

Each encryption in F5 must be performed in under 0.1 seconds. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess a sufficient processing power and RAM. See table 21 on page 109 for hardware specifications.

Clarification: This requirement's upper-bound in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

U3-F9: DB FUNCTIONALITY TIME

Each operation in U3-F6 must be performed in linear time to the amount of database rows (Search) and number of $CatalogPrice$ elements required for the summation (SUM).

Clarification: A database other than the TLPRD (e.g. LPRD) can consist of several million rows. We deem it required that summation queries can be performed on large databases without an expensive time complexity.

U3-F10: PROXY DECRYPTION TIME

Each decryption in U3-F7 must be performed in under 1 second. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess at most 6GB RAM and processing power in the order of a Duo CPU 3.00GHz processor.

Clarification: This requirement's upper-bounds in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

6.4.5 *Use Case 4 : Calculating an Average*

An operation that can be performed on the RDW's license plate database is the calculation of the average of a set of integer values. Currently, this functionality is less important, but with the future inclusion of sensor data this might become an interesting property. In this use case, we assume that for explanatory purposes there is the need to calculate the sum of the $CO2Emission$ values of all cars registered to a particular BSN number. We define this functionality at the hand of our functional requirement model of section 6.4.1. Each requirement is encoded with U4-FX for Use Case 4, where X is the number referring to the type of requirement found in table 17.

U4-F1: QUERY FORMAT

A client sends a summation query to the Proxy server in the format :

```
SELECT AVG(Attribute) AS "Sum" FROM DBTable WHERE Condition;
```

General MySQL summation query where Attribute equals a database column in TLPRD and Target a variable in that column [Mys].

Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constraints.

U4-F2: ENCRYPTED QUERY FORMAT

The Proxy sends summation queries to the Cloud server in the format :

```
SELECT UDF(C(CO2Emission)) AS "Sum" FROM TLPRD WHERE Condition;
```

Where UDF is a user defined MySQL function that should operate on C($CO2Emission$) as AVG would on $CO2Emission$. C(x) annotates a cipher generated by an encryption scheme C with input x. And Condition is a search criteria on BSN, which implies the functionality of Use Case 1. *Clarification: We desire that Clients can use MySQL syntax as this provides clear compatibility constrains.*

U4-F3: ENCRYPTED RESULT FORMAT

The Cloud sends the query results to the Proxy server as an encrypted value in the format C($CO2Emission$). *Clarification: We require a result encrypted in the same format (same key, encryption scheme) as the $CO2Emission$ in the TLPRD as this allows results to be compatible with with calculations that are compatible with the $CO2Emission$ encryption format.*

U4-F4: RESULT FORMAT

The Proxy sends summation query results to the Client as a $CO2Emission$ in plaintext. *Clarification: The client requests the sum of multiple $CO2Emission$ values. It is therefore assumed that returning the result in the same format allows the client to process this result further without requiring additional information.*

U4-F5: PROXY ENCRYPTION

The Proxy server is required to be able to transform a query of format U4-F1 to a query of format U4-F2. This requires the Proxy server to be able to generate a MySQL UDF that can provide the functionality of AVG under the encryption model used for $CO2Emission$. *Clarification: A UDF is required instead of AVG because encrypting full homomorphic encryption schemes like Gentry's FHE [Geno9] requires intermediate bootstrapping operations.*

U4-F6: DB FUNCTIONALITY

The encrypted $CO2Emission$ column can be used in 2 different ways in order to allow for the calculation of an average. To comply with this requirement at least on the two following sets of conditions have to be satisfied:

SET 1 :

1. The encrypted $CO2Emission$ column supports the full homomorphic property (summation and division).
2. The encrypted $CO2Emission$ column supports the counting of the number of elements.

SET 2 :

1. The encrypted $CO2Emission$ column supports the additive homomorphic property (summation).
2. The encrypted $CO2Emission$ column supports the counting of the number of elements.
3. The division of the results of condition 1 and condition 2 can be done by the Proxy server

Clarification: We require the cloud provider to perform all computationally demanding operations. Since the computational cost of summation and count is dependent on the number of elements in the summation, we classify them as (potentially) computationally demanding. The calculation of an average value (sum/count) only required one division, which computational cost can be seen as constant. We consider this cost to be insignificant (in respect to encryption en decryption requirements) and, therefore, allow for Set 2, in which the proxy server provides computational support for that divisions.

U4-F7: PROXY DECRYPTION

The proxy is required to be able to decrypt the following attribute :

1. C($CO2Emission$) $\rightarrow$ $CO2Emission$).

Clarification: Proxy should be able to decrypt the encrypted $CO2Emission$ value as the average of encrypted $CO2Emission$ values is in the same format. We desire proxy-side decryption so that a client can directly process results and does not require a decryption key.

U4-F8: PROXY ENCRYPTION TIME

Each encryption in U4-F5 must be performed in under 0.1 seconds. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess a sufficient processing power and RAM. See table 21 on page 109 for hardware specifications.

Clarification: This requirement's upper-bounds in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

U4-F9: DB FUNCTIONALITY TIME

Each operation in U4-F6 must be performed in linear time to the amount of database rows (Search) and number of $CO2Emission$ elements required for calculating an average (AVG).

Clarification: A database other than the TLPRD (e.g. LPRD) can consist of several million rows. We deem it required that queries for an average can be performed on large databases without an expensive time complexity.

U4-F10: PROXY DECRYPTION TIME

Each encryption in U4-F7 must be performed in under 0.1 seconds. This is under the assumption that the to be encrypted value is at most 100Kb in size and the proxy server possess a sufficient processing power and RAM. See table 21 on page 109 for hardware specifications.

Clarification: This requirement's upper-bound in terms of both hardware and timing are based on intuition and only serve to enforce practical feasibility of our model and should be interpreted according.

6.5 SECURITY REQUIREMENTS

6.5.1 *Threat Model and Assumptions*

Our threat scenario is based upon the implication of the Patriot Act as described in section 3.5.1. We consider that our private database in stored at a cloud provider that is forced by a foreign government to release the database and a list of performed queries to an untrusted party (e.g. the U.S. government). The malicious government does not attempt to actively manipulate the database and is only concerned with an attempt to obtain sensitive data (Personal data) from both a copy of the license plate database(s) and performed queries. We have made the following Threat Assumptions (TA) regarding the intended and reach of a malicious entity against which our solution design of section 7.2 should suffice.

TA-1 All Cloud hosted databases are available to the malicious entity.

TA-2 All by the Proxy performed queries are available to the malicious entity.

TA-3 All attributes of one entry (his/her own) are known to the malicious entity.

TA-4 The malicious entity only tries to break the confidentiality (of personal data).

TA-5 The malicious entity only observes and does not actively try to change, delete or add data. Unless this would result in a scenario in which it is trivial that the confidentiality of private attributes is compromised.

In the threat model of section 6.5.2 we set out five type of threats derived from these five treat assumption, against which our solution design of section 7.2 must provide sufficient confidentiality guarantees.

6.5.2 *Confidentiality Requirements Model*

We consider our private (personal) database secure from confidentiality oriented perspective if five requirements are met. We constructed these five requirements based on our visualisation of this threats scenario illustrated in figure 19, which is based on our threat assumptions of section 6.5.1.

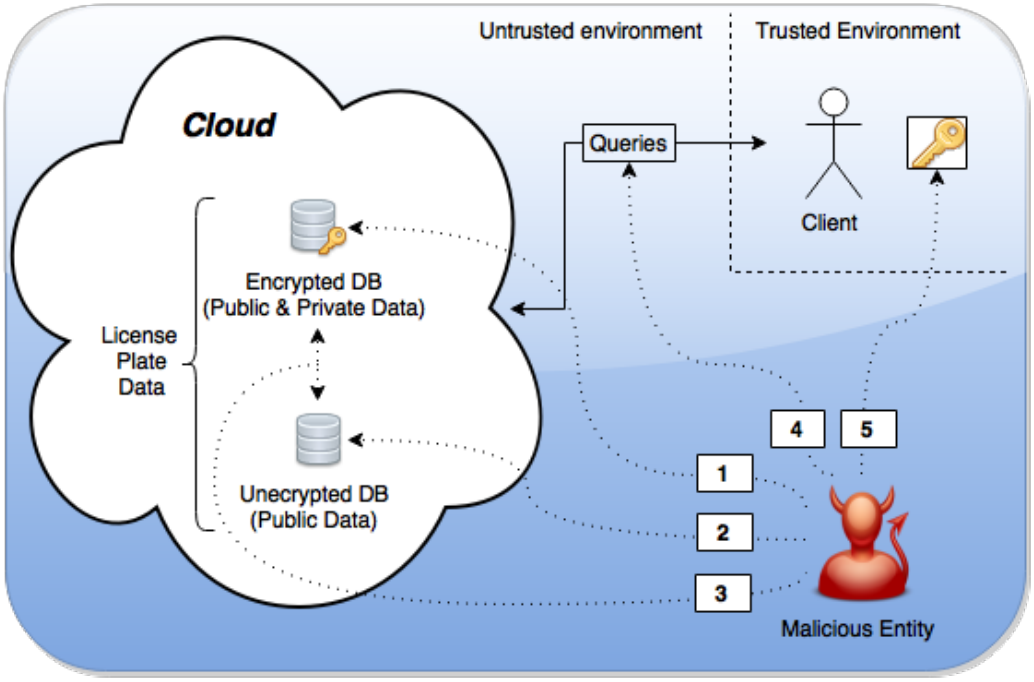

Figure 19: Overview of the aspects we have taken into account in order to formulate confidentiality requirements.
1: Breaking Encryption : Leakage through breaking encryption(s).
2: Static Analysis : Leakage trough non-event based DB correlations.
3: Dynamic Analysis : Leakage trough event based DB correlations.
4: Query Analysis : Leakage through requested or performed queries.
5: Key Theft : Leakage or theft of the secret key(s).

We consider our private (personal) database secure from confidentiality oriented perspective if the following Confidentiality Requirements (CR) based on figure 19 are met:

6.5.3 *CR-1 : Breaking Encryption*

It should be infeasible for the malicious entity to decrypt any encrypted value without a private key. We consider this to be the case if all deploy encryption models comply with current day encryption standards and key lengths. Recommendations as proposed by the NIST with a suspected lifetime of 15 years (2030) are deemed satisfactory for this requirement [Gir].
Clarification : Personal data might require lifetime (50+ years) security guarantees. We decide to not make explicit assumptions on the time spend to which the encryption of the involved attributes should remain relevant and leave this open for future analysis. For the purpose of this thesis we assumed ***(Assumption CA-1 section 7.3.1)*** *that 15 years of protection is sufficient.*

6.5.4 *CR-2 : Static Analysis*

It should not be possible for a malicious entity to derive sensitive data from obtaining both the Encrypted DB and Unencrypted DB in "cold" state (e.g. a database bump file of both DB's). We set out the conditions for this scenario at the hand of figure 20.

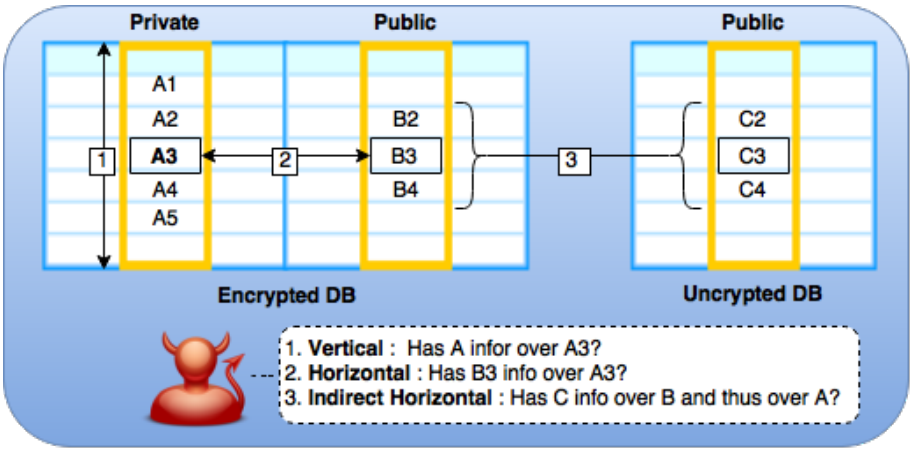

Figure 20: Model illustrating the static correlations of private values (A3) in respect to other values (A1-5) and attributes (B, C).
Vertical: Correlation with other private values of the same attribute (A1-A5).
Horizontal: Correlation with public attributes in the same database (B3).
Indirect Horizontal: Indirect correlations with public attributes of another database (C2).

We will explain our these conditions at the hand of figure 20 at the hand of 3 conditions that have to be satisfied in order to comply with requirement CR-2.

VERTICAL

We require that (private) attributes are encrypted in such a way that their plain value cannot be derived from observing other encrypted values of that attribute. We consider this the case if deterministic encryption is exclusively used on attributes that do not contain collisions of values registered to different persons (BSN).
Clarification: In section 6.5.1 we set out assumption TA-1 and TA-3 from which we can derive that a malicious entity know the encrypted database and its own values (in plain format). A malicious entity could use this information do decipher (private) attributes of other entries if they would have equal values and where determinacy encrypted. An example of this could revealing entries with the same legal status (Legal) or name (NameOwner) breaking confidentiality as set out in our naive approach in section 5.2 and is illustrated in the TLPRD of section 6.3.2 (Label 1) .

HORIZONTAL

There should not be any observable correlation between plain public and encrypted private attributes in the same database. We, therefore, require all private and public attributes in the same database have to be encrypted.

Clarification: When encrypted private and plain public data are stored in the same table sensitive correlation may occur. When, for example, a set of public attributes becomes unique for a private value it allows for that value to be derived. This example is illustrated in the TLPRD of section 6.3.2 (Label 2) in the entry of "Alen Dooper". Even if Alen Dooper's name is encrypted, it is possible to connect his name to that entry when his cars public specifics are known. This can be done by simply looking for the unique set of public value's Clour (Golden) and Brand (Kia). This allows an adversary to derive which cipher decrypts to "Alen Dooper", what compromises the encryption of that private attributes value.

INDIRECT HORIZONTAL

The Encrypted DB and Unencrypted DB should not have any observable correlations. We consider this the case if all encrypted values of shared attributes map to different ciphers breaking any data repetition patterns. This means that unlike private attributes public attributes may never rely on deterministic encryption.
Clarification: The previous example (2) would also occur when plain public data can be mapped to encrypted public data stored with private data. This would, for example, be the case for the Attribute Legal in the TLPRD, when deterministic encrypted is used as explained in 5.2 and illustrated in the TLPRD of section 6.3.2 (Label 3)

6.5.5 *CR-3 : Dynamic Analysis*

It should not be possible for a malicious entity to derive sensitive data from events that affect both the data in the Encrypted DB and Unencrypted DB. This type of sensitive data might, for example, be obtained from updates that effect attributes in both the Encrypted DB and Unencrypted DB (e.g. Update Legal for a particular license plate number). We illustrate this in figure 21 to give a clear overview of this type of correlation.

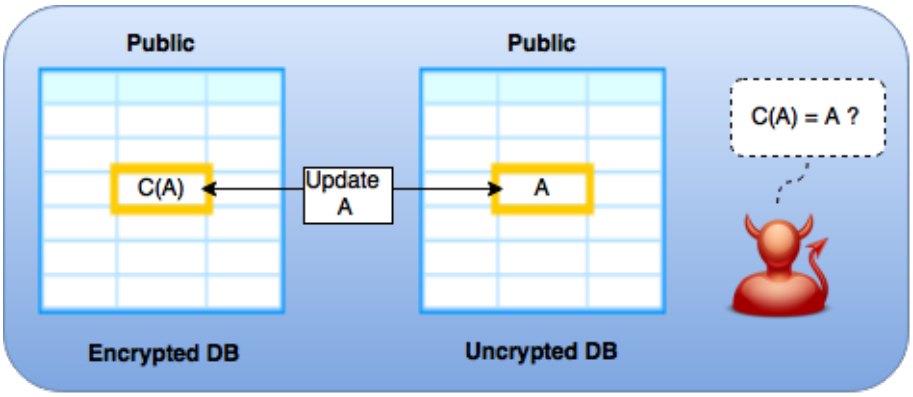

Figure 21: Model illustrating the dynamic correlations between the Encrypted DB and Unencrypted DB caused by updates. Both DB's share public attribute A, which is encrypted in the Encrypted Db (C(A) and in plaintext (A) in the Unencrypted DB. Update A cause both value C(A) and A to be updated (simultaneously) leaking the information that C(A) and A might be connected to each other.

To prevent dynamic analyses, we require queries that effect both to the Unencrypted DB and Encrypted DB cannot be linked. We consider this the case if those updates only affect the Encrypted DB in real time, synchronising the Unencrypted DB and Encrypted DB only when a certain threshold of changes have been made (e.g. 100+). synchronization will then cause multiple updates at once obfuscates correlation between update attributes across databases.
Clarification: Both databases share their public attributes, resulting in that changes in these characteristics affect both databases. We don't want to encrypt public attributes in the encrypted database to reveal their plaintext value as this would compromise private attributes (see CR-2.3). Our requirement forces the unencrypted DB to be less updated than the encrypted database, but we assume ***(Assumption CA-2 section 7.3.1)*** *that update traffic is high enough for synchronizations to happen within a reasonable time interval (e.g. 1 day)*

6.5.6 *CR-4 : Query Analysis*

It should be impossible for the malicious entity to derive sensitive information from obtaining or performing queries. We separate this requirement into two sets of conditions that all have to be satisfied, differentiating between conditions are dependent on the specification of a particular query (**Single Query**), or those that depends on the fact that sending queries allows for a pattern to be formed (**Multiple Queries**).

SINGLE QUERY

We require all queries of use cases 1 till 4 of section 16 within an untrusted environment (Proxy → Cloud) to prevent the leakage of sensitive information. We illustrated the conditions we derived for this in figure 22.

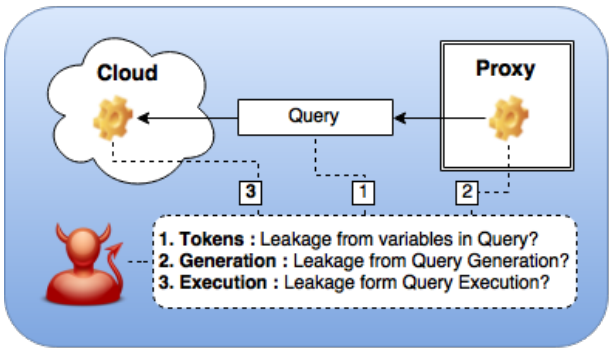

Figure 22: Model of CR-4 single query analysis, in which sensitive leakage of the following three attack vectors is covered. **Token :** Leakage that can occur through variables obtained in the query (e.g. plaintext variables). **Generation :** Leakage through manipulation of query generation (e.g. sending wrong information to influence encryption generations). **Execution :** Leakage through effects of queries on the database (e.g. updating non-deterministic encrypted values to deterministically encrypted values)

In figure 22 we set out how we constructed the confidentiality guaranties that we require for the tokens, generation, and execution of our queries. This requirement (Single Query) is satisfied if the following conditions are met:

TOKEN
: Each query send from the Proxy to the Cloud may not contain (private) attributes in plaintext.
Clarification: All attributes set to the cloud have the be encrypted in order to assure that the cloud is unable to correlate plaintext values with encrypted values

GENERATION
: The proxy generates each query for the Cloud exclusively based on Client input and its internal state (key's).
Clarification: This prevents the cloud from potentially manipulating the encryption processes performed by the proxy

EXECUTION
: The execution of each query may not change the (previous) decryption or encryption assumptions of the Encrypted DB.
Clarification: This would could compromise previously made confidently assumptions.

MULTIPLE QUERIES
: Secure queries that on themselves do not leak confidential information may do so in larger numbers. Correlation might, for example, occur if the Police always queries for cipher A on the same days that offender B (cause of A) gets a notice for speeding. Over a longer period, these correlations might be derivable (breaking confidentiality) by a malicious entity that is assumed to know all queries **(Assumption TA-2 section 6.5.1)**. We illustrated this type of threat in figure 23 to clarify this statement.

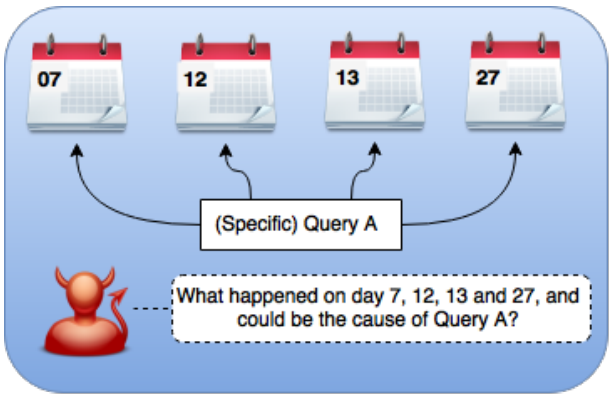

Figure 23: Model of CR-4 multiple query analysis, in which a specific query A has been issued on 4 different dates. Correlations might be derived if additional information about these 4 days is known (e.g. speeding tickets for person B). The strength of these type correlations grows if query data over a longer period of time is collected (e.g. years).

To prevent multiple queries from forming (strong) patterns we require periodical re-encryption of all encrypted attributes in the Encrypted DB, preventing ciphertext matching cross re-encryption periods.
Clarification: Re-encryption of the Encrypted DB has to be done in a trusted environment and with a frequency that has to be determined depending the speed in which pattern may occur, and the acceptance range of the patterns. In this these we assume that performing a re-encryption once a day is sufficient ***(Assumption CA-3 section 7.3.1)***.

6.5.7 *CR-5 : Key Theft*

The secret key may not be obtainable by a malicious entity. We consider this the case if the secret key can remain in a trusted environment at all time.
Clarification: We like to keep the secret key in a trusted environment as this provides maximum security guarantees. Other options like placing a secret key in secure hardware at an untrusted location (Cloud), might be acceptable we prefer not to rely on this as this would open the possibility to side channel analyse attacks.

Part V

ANALYSIS

7

CASE STUDY

In chapter 1.2 we set out the purpose of our research, focusing on enforcing confidentiality while processing personal data in the cloud. We defined a specific scenario based on the RDW's processing of license plate data and researched literature on the legal implications and cryptographic possibilities. In this chapter, we set out a solution for our toy scenario at the hand of the following three sections.

DERIVED APPLICABLE LEGAL FRAMEWORK :
: In section 7.1 we explain that Dutch personal data may be processed by a (foreign) cloud provider. We also set out the legal restrictions to this type of processing and how we have taken this into account in the proposed solution.

PROPOSED SOLUTION :
: In section 7.2 we set out a new solution design for the TLPRD based on the case description of Part iv.

COVERAGE OF REQUIREMENTS :
: In section 7.3.2 we will give an overview of how are proposed solution covers all the functionality and security requirements, including an assessment of its feasibility.

7.1 APPLICABLE LEGAL REQUIREMENTS

In our RDW based license plate database scenario, we work with personal data in the cloud that leads to legal implications (Section 3.3) which constrain our database design. In this section, we explain the legal implications relevant to our use case and take into account our proposed solution 7.2.
Personal data is considered sensitive data within the EU and may only be processed under well-defined constraints formulated in the Dutch WBP law, which is based on EU Directive 95/46/EC. Within the coming years both these rules will be replaced by the GDPR, which implements more and stronger cloud forced privacy requirements on the processing of personal data (Section 3.3.3). Cloud providers that comply with the WBP can show this by obtaining either a Safe Harbor certification or compliance with the EU module clauses (Section 3.4). The RDW uses cloud services provided by Microsoft Azure for public data [RDWb]. Microsoft Azure is compliant with the EU Model Clauses [Azu] and can, therefore, be considered as an option to host and process personal data.
Using a certified cloud provider is not sufficient as the WBP states that personal data still

requires to be appropriately encrypted, whether it is stored, in transit or processed. The European Commission's Article 29 working group's vision on cloud computing states that a data provider may not consider a by a (certified) cloud provider managed encryption to ensure the confidentiality of personal data [soFRotEC12]. An appropriate encryption solution does not allow the cloud provider to obtain a private key used to provide reasonable protection of data confidentiality. The definition of reasonable confidentiality is ambiguous and in this thesis defined at the hand of threat requirements (Section 6.5). The proposed encryption model in section 7.2.2 is, therefore, independent of any active cooperation from a cloud provider and generates and keeps private keys at a by the client (RDW) chosen location.

7.2 SOLUTION DESIGN

In section 5 and 5.5 we set out four encryption models: Naive Approach, C-SDA, GhostDB and CryptDB. These models use different approaches to secure data confidentiality for a database consisting only of private data. None of these models focus on the risk of combining private and public data, which requires additional confidentiality guarantees based on preventing database correlations (Dynamic and Static) as set out in section 6.5.2. We, therefore, propose a new solution which is based on CryptDB. We base our solution on CryptDB because it's deployment model requires no additional hardware in untrusted environment (CR-5) and provides MySQL support for queries requiring deterministic (DET) and partial homomorphic (HOM) operations, as is required by our functional requirements. In section 7.2.1 we set out the current limitations of CryptDB regarding our case description, providing a clear overview of the additional confidentiality guarantees our model aims to implement to comply with our functional and confidentiality requirements. We then explain our proposed solution at the hand of a new deployment model from a higher level of abstraction in section 7.2.2 and a new proxy based encryption model in section 7.2.2 to explain the cryptographic contractions used on the TLPRD.

7.2.1 *CryptDB's Coverage of Requirements*

In section 5.5 we explained the cryptographic model of CryptDB and its ability to ensure confidential interaction with an encrypted database. In this section, we set out the degree to which CryptDB covers our functional and confidentiality requirements. We divided this section into the following parts:

- **Coverage of Functional Requirements** (Section 7.2.1.1)
  Here we explain which functional requirements of section 6 are covered by CryptDB (Release of February 2014) when the deployment model of section 5.5 is used for the TLPRD.
- **Coverage Confidentiality Requirements** (Section 7.2.1.2)
  Here we explain which confidentiality requirements of section 6 are covered by CryptDB (Release of February 2014) when the deployment model of section 5.5 is used for the

TLPRD. Additionally we assume the PS-DLPR to be active in a public cloud as assumption for our confidentiality requirements.

- **Overview of Possible Improvements** (Section 7.2.1.3)
  Here we provide an overview of the shortcomings of CryptDB and which adjustments can be made to overcome these limitations. These improvements are included in our proposed solution and encryption model.

7.2.1.1 *Coverage of Functional Requirements*

In sections 5.5.1.1 (RND), 5.5.1.4 (DET) and 5.5.1.2 (HOM) we explained that CryptDB uses pseudorandom, deterministic and (partial) homomorphic cryptographic models in a way that theoretically provides all desired functionality for our functional requirements (Format, Functionality and Time) of section 6.4.1, excluding F8, F10 as those are server dependent. In order to test the CryptDB required by our use cases we built a test setup illustrated in figure 24.

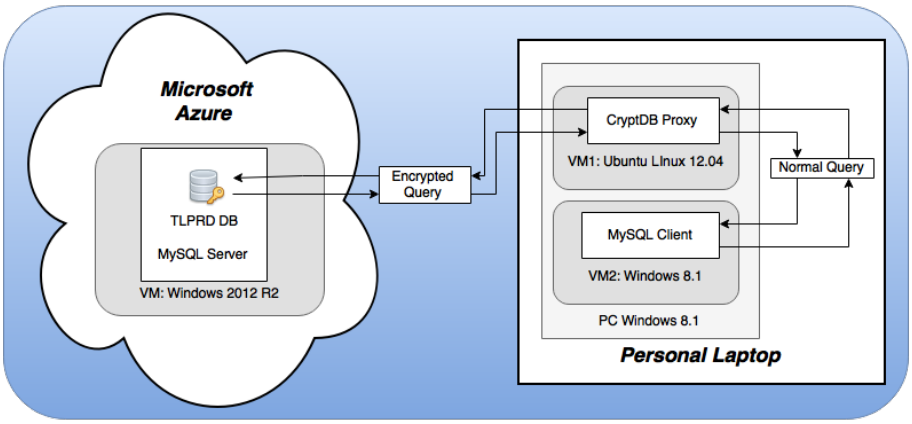

Figure 24: The setup used for the testing of all use case queries on CryptDB

To test the required functionality of CryptDB we used a personal laptop (Lenovo Yoga 2: Core i5, 6 RAM) and one-month free trial account for the Microsoft Azure cloud. On our laptop, we used two virtual machines, one running Windows 8.1 serving as a MySQL client, and one running Ubuntu 12.04 serving as the secure proxy by running CryptDB. In the Azure cloud, we also made a virtual machine which ran Windows 2012 R2 and hosted the MySQL server storing the TLPRD database. We tested this setup for every use case with 25 randomly generated queries compatible with the F1 functional requirement. We manually checked whether the result was in format F4, correct and complete. We verified that this was the case. We also ran a set of 40 queries in random order existing of 10 queries of each of the four use cases as a test, at which we didn't find any conflicts. Both tests confirmed that CryptDB's current implementation supports the by us precisely defined MySQL query formats as can be expected from CryptDB's analyses on supporting

determinacy, and homophobic encryption explained in section 5.5.1. We did not perform a benchmark, as our setup was not optimized (Limited trial account Azure and limited hardware allocations in VMs) and extensive real life benchmarking of CryptDB is present in CryptDB's release paper [PRZB11a]. Our conclusion from this setup is that CryptDB's model forms a feasible solution for the earlier stated use cases when looking at queried support summarized in table 18.

Use Case	Case Description	Supported by CryptDB
1	Preforming a Database Search	Yes
2	Preforming a Database Update	Yes
3	Calculating an Summation	Yes
4	Calculating an Average	Yes

Table 18: Overview of CryptDB's support for the provided use cases.

7.2.1.2 *Coverage Confidentiality Requirements*

CryptDB only takes one single database into account and ignores the possibility of shared data with a public database or an attributes entropy when applying deterministic encryption. These assumptions results in that CryptDB is unable to satisfy all our confidentiality requirements as explained in table 19. All not satisfied requirements (CR-2,3 and 4) are the direct result of either enabling use case 1 (SEARCH) or 2 (UPDATE)) using CryptDB on a database that contains both private and public data.

Confidentiality Requirements	Satisfied	Motivation
CR-1 : Breaking Encryption	Yes	All used encryption's (RND, DET, DET* and HOM) are based on the discrete logarithm problem and are classified by NIST to be infeasible to break until at least 2030 when a key a relative key length of 256 bits is taken (See table 21).
CR-2 : Static Analysis	No(1+2+3)	Not satisfied since the requirement Horizontal is not met.
Vertical	No (1)	Not satisfied since the attribute $Name$ gets deterministically (DET) encrypted in use case 2. This causes different persons (BSN) to have the same cipher when having the same $Name$ value, which are required to be private.
Horizontal	Yes (2)	Satisfied since all attributes in the TLPRD are encrypted (Section 5.5).
Indirect Horizontal	Yes (3) Though not enforced	Satisfied since there is no observable correlation between an encrypted public data and not encrypted public data. This is the case since only RND or HOM encryption's are used on the public attributes of the TDLPR. Note however that is possible to break this guarantee as CryptDB does allow RND attributes to decrypted to DET ciphers when such a query occurs without additional checks. This is not included in our scenario but should be considered when use cases get extended.
CR-3 : Dynamic Analysis	No	The PS-DLPR and TLPRD cannot be linked through query patterns. This is the case since updates can occur synchronized on both databases.
CR-4 : Query Analysis	No (4+5+6+7)	Not satisfied since the requirement for Multiple Queries is not met.
Single: Token	Yes (4)	Satisfied since all F2 functional requirements of each use case require a query to the cloud to have no plaintext variables , which is guaranteed [PRZB11a] by using the CryptDB proxy.
Single: Generation	Yes (5) Though not enforced	Satisfied since all F5 functional requirements of each use case require a queries to the cloud the be generated by the proxy without information provided by the cloud. Note however that though this is the case for our use cases this is not enforced by CryptDB as queries are allowed to request intermediate computational support for multiplications. This requirement will no longer hold when new use cases require this support intermediate instead of last step (e.g. nested constructions using SUM or AVG).
Single: Execution	Yes (6)	Satisfied since all F6 functional requirements of each use case provides no conflict with other requirements as tested in section 7.2.1.1.
Multiple Queries	No (7)	Not satisfied since leakage through observing multiple queries over a period of longer than one day time is not being prevented. Re-encryption can be done manually, but this would require CryptDB to operate, inefficient as an excessive amount of encryption layers have to re-applied since it is only allowed to decrypt on run-time. Additionally this would cause lag the first time as specific type of query is applied since an columns first require a decryption update. In CryptDB's released benchmark [PRZB11a] it is also stated that support for re-applying encryption layers can be considered as possible improvement.
CR-5 : Key Theft	Yes	Satisfied since all F6 functional requirements can be satisfied without requiring a secret key. No secret key is there for required to be located outside the trusted environment when CryptDB runs from a trusted environment.

Table 19: Coverage of confidentiality requirements under use case 1, 2, 3 and 4 by using CryptDB.

7.2.1.3 *Overview of Relevant Improvements to CryptDB's Model*

In sections 7.2.1.1 and 7.2.1.2 we explained the degree to which CryptDB is a suitable solution for processing the TDLPR given our requirements of section 6. Though CryptDB provides all functionality required, it does not satisfy all of our defined confidentiality guarantees. Additionally CryptDB's architecture is more complex using more encryption layers than required, negatively affecting performance when initiating or re-encrypting a database. We, therefore, composed the following list of four adjustments we incorporated in our proposed deployment and encryption model in respectively sections 7.2.2 and 7.2.3.

1. The attribute $Name$ should allow for modifications without reducing CryptDB's definition of deterministic encryption (CR-2 Vertical).
2. CryptDB allows public attributes stored with private attributes to be updated to determinacy encryption without additional checks. A proxy should prevent this deterministic encryption for public attributes that have patterns caused by duplicate value's (CR-2 Indirect Horizontal).
3. A proxy should ensure that the PS-DLPR cannot reveal sensitive information about TLPRD by allowing synchronized access to both databases (CR-3).

4. A proxy should limit patterns caused by the repetitive use of the same query as this could correlate sensitive information to public information or events (CR-4 Multiple Queries).

7.2.2 *Proposed Deployment Model*

In our deployment model, we have taken into account that there are two separate databases that share attributes and should remain separate. We used CryptDB's deployment model and extend that with periodical re-encryption and query restriction in order to address CryptDB's limitation regarding confidentiality (CR-Indirect Horizontal, CR-3 and CR-4 Multiple Queries) as set out in section 7.2.1.3. In this section, we explain our proposed solution for the RDW based on the requirements and database assumptions of section 6. We first provide a complete overview of our deployment model in figure 25 and then explain all elements of figure 25 and our design choices in the reminder of this section.

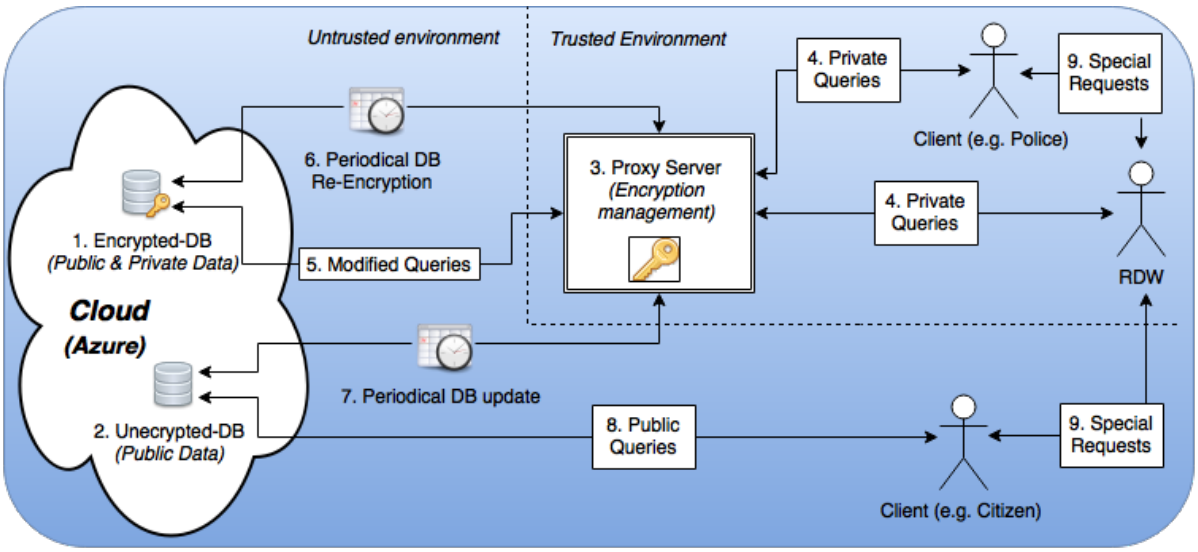

Figure 25: Overview of proposed deploy model for secure license plate data processing in the cloud. Explanations of, and motivation behind, all included elements can be found in the form of an enumeration in section 7.2.2.

In figure 25 we illustrated an overview of our proposed deployment model constructed with nine elements. We will here explain these nine elements and motivate their design.

1. ENCRYPTED DB :
   The encrypted DB consists of the TLPRD, which contains both personal and public information and resembles the DLPR database of the RDW. All attributes of the TLPRD are encrypted and placed in the cloud by a secure Proxy Server. The encryption methods used by this Proxy Server are set out in section 7.2.3.

Clarification: the Encrypted DB can be seen as the complete license plate database currently hosted by RDW.

2. UNENCRYPTED DB :
The unencrypted database is a subset of the Encrypted DB only including the public attributes from the TLPRD.
Clarification : The Unencrypted DB can be seen as the public license plate database, which is hosted by Azure [Mar14].

3. PROXY SERVER :
The Proxy server is a by the RDW or other trusted party hosted server, which can manage all private keys. The Proxy Server encrypts and decrypts queries between the client and the cloud.
Clarification : This proxy server is based on the deployment model of CryptDB and can be considered a similar tool with different settings. For more details on the use of encryption proxies see CryptDB section 5.5.

4. PRIVATE QUERIES :
Private queries are queries that contain or require at least one private attribute and, therefore, require access to the encrypted DB. They are sent by a client to the Proxy Server that modifies (encrypts) them before sending them to the untrusted cloud. All supported private queries were set out in the use cases of section 6.4 under the label "F1: Query Format".

5. MODIFIED QUERIES :
Modified queries are private queries that are modified by the Proxy Server in such a way that they do not reveal plaintext values and are compatible with "1. Encrypted DB". These queries are defined as "F2: Encrypted Query Format" in section 6.4.

6. PERIODICAL DB RE-ENCRYPTION :
In security requirement CR-3 (Section 6.5.5) we described the risk of "Dynamic Analysis" in which the cloud provider can correlate the Encrypted DB and Unencrypted DB through observed (update) query patterns. To limit the time spent in which queries may cause patterns, we propose to re-encrypt the Encrypted DB every day. Re-encryption can be established by temporarily preventing changes made by update queries to the Encrypted DB. This can, for example, be done in a time (e.g. at night) when Encrypted DB traffic is low as mentioned in CA-3 of section 7.3.1. In this time the Encrypted DB can be downloaded, decrypted, re-encrypted and shuffled by Proxy Server using different private keys. This will be done to prevent correlations between queries performed during different encryption periods as even deterministically encrypted values will then map to other ciphers. We recommend Periodical DB Re-Encryption to occur in the following four steps:

6.1 TEMPORARILY NO UPDATES
Every day the Encrypted-DB will not allow updates for a limited amount of time (e.g. 1 hour) during a period in which traffic is low (e.g. at 1 am if most Dutch public services are closed).

6.2 DB TO PROXY

During "1. Temporary No Updates" the complete Encrypted-DB is send to the Proxy Server. We assume that this step can be performed within an acceptable amount of time. We make this assumption because we were able to download 100MB from Azure (azurespeed.com) in under 10 seconds (10MB/s) at home using optic fiber cable Internet. For a database of 10GB, this could theoretically take around 15 minutes or less.

6.3 RE-ENCRYPTION

During "6.1 Temporary No Updates" the Proxy Server decrypts and re-encrypts all values in Encrypted-DB, after which all rows get randomly reordered. We assume this step to be performable within an acceptable time frame based on CryptDB's performance of encrypting 1 KB with AES in CBC in 0.008 ms and decrypting it in 0.007 ms [PRZB11a]. From CryptDB's results, we derive that for 10 GB in AES-CBC encryption plus decryption would take 150 seconds (0.015 ms per 1 KB) if no other time factors are taken into account. Additionally dedicated encryption hardware at the proxy server could be considered to achieve a good re-encryption speed.

6.4 DB TO CLOUD

The proxy server sends the re-encrypted DB back to the Cloud that replaces the previous DB with the newly encrypted one, after which everything returns to its normal state. We tested Azure's upload speed using azurespeed.com and got an average upload speeds of 2 MB/s on Dutch Azure servers. We consider this a sufficient upload speed and assume that this rate can be improved by contractual agreements with Azure.

7. PERIODICAL DB UPDATE :

A single row update performed simultaneously on both the encrypted database and the unencrypted database, would reveal unwanted correlations to the cloud provider as set out in security requirement CR-4 (Section 6.5.6). To prevent this, only the Encrypted DB is updated real-time. The Unencrypted DB is only updated periodically (e.g. once a day), obfuscating all correlations between the two databases caused by (update) queries.

8. PUBLIC QUERIES :

Public queries are search queries that are sent to the unencrypted database in the cloud and require no modification by the Proxy Server. This is limited to search queries that only require the involvement of public attributes that are allowed to be out of date by a small offset, e.g. 1 day, equal to at least the time frame in which the periodical database update (7. Periodical DB update) gets performed.

9. SPECIAL REQUESTS :

Special requests are either queries that cannot be processed in the cloud or requires the active involvement of the RDW. An example of this is the look-up of a particular license plate number by the RDW in their private database to assist the police.

Clarification : This part of our model is out of our scoop, but we included it to illustrate the completeness of a deployed solution.

7.2.3 *Proposed Encryption Model (Proxy)*

In this section, we will set out the encryption models used in our solution for TLPRD to comply with our functional requirements of section 16 and our security requirements of section 6.5.2. We set out this section in the following parts:

- **Attribute Depended Encryption's** (Section 7.2.3.1).
  Here we explain the need for attribute dependent encryption choices and how we structure these choices to comply with all previously mentioned requirements.
- **DET* : RND Depended on a DET Attribute** (Section 7.2.3.2).
  Here we explain our newly introduced encryption model for use case 2 (U2-F6).
- **Implementation of Encryption Models** (Section 7.2.3.3).
  Here we explain the implementations of our encryption model and its expected performance in terms of encryption efficiency, decryption efficiency, and cryptographic strength.

7.2.3.1 *Attribute Depended Encryption's*

In our deployment model of 7.2.2 we assumed the encrypted database allowed for the required query functionality without the cloud provider possessing a private key. This is not a trivial problem as also stated by the European Commission's Article 29 working group, which warned that providers of personal data should keep in mind that encryption methods enforcing confidentiality against the cloud provider limit the cloud provider's ability to process that data [soFRotEC12]. The main reason for this, as explained in section 5.5.1, is that the inability to decrypt values limits the type of queries that can be applied dependent on the kind of encryption and deployment model. The encryption methods and the way they are deployed should, therefore, be dependent on the functionality required from the cloud provider. Not all encryption methods provide unrestrained confidentiality guarantees as set out in CryptDB's explanation in section 5.5. It is therefore essential that the type of data is taken into consideration when selecting appropriate encryption models. In order to determine an encryption scheme for each attribute in the TLPRD (Table 15) we have set out a flowchart in figure 26 in order to assure that all encrypted attributes of the TLPRD comply with both our functional requirements of section 16 and our security requirements of section 6.5.2.

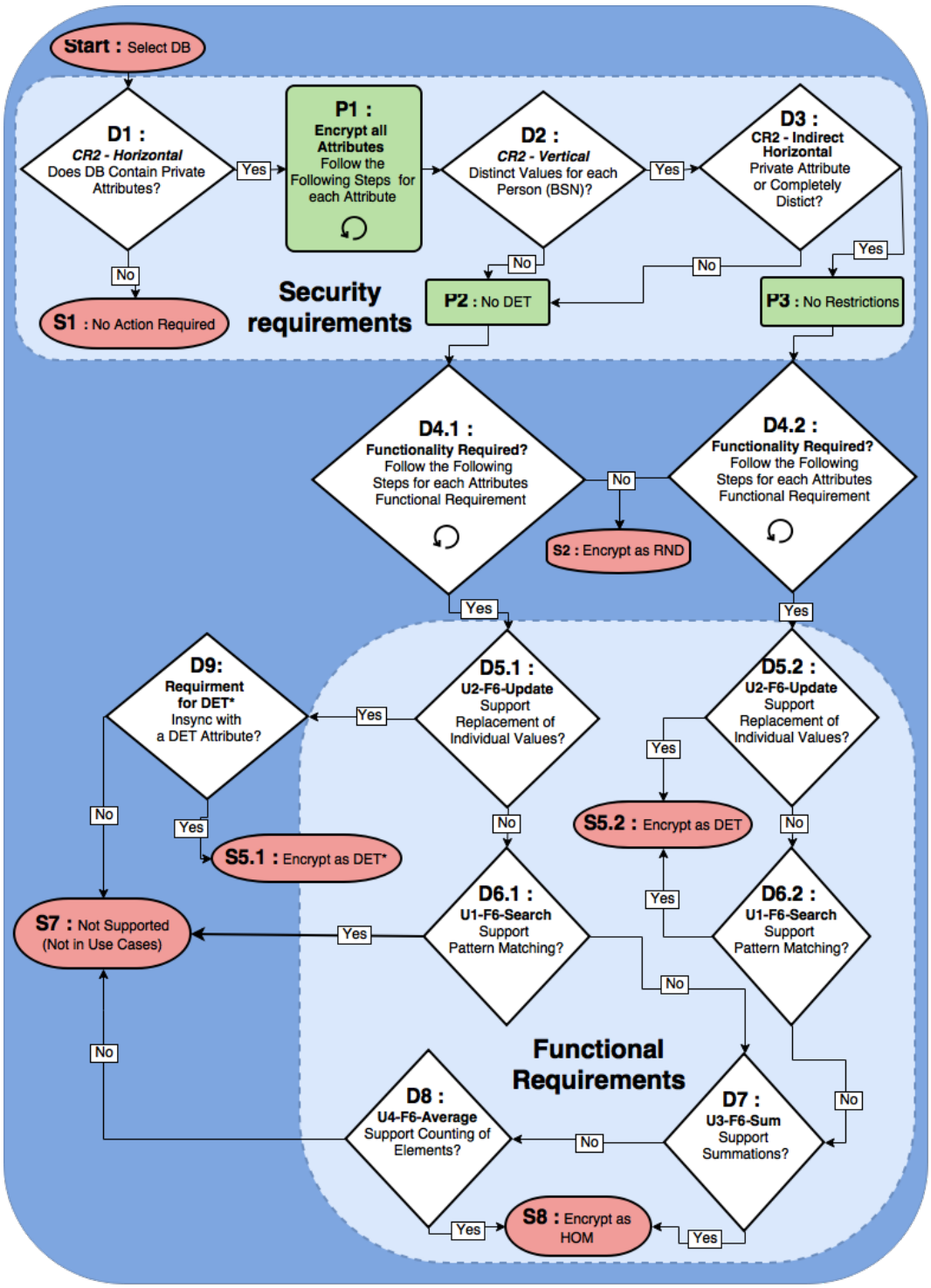

Figure 26: Flowchart for selecting an encryption model for each attribute in TLPRD based on our functional requirements of section 16 and our security requirements of section 6.5.2. All steps are labelled either Decisions (**D**, Diamonds), Process-state (**P**, Rectangle) or Solution (**S**, Circular). A detailed explanation on how to derive an appropriate encryption model for each attribute is given in section 7.2.3.1 together with the definition of the decryption models RND, DET, HOM and DET*.

The flowchart of figure 26 shows the steps that a database (e.g. TLPRD) has to follow in order to comply with both our F6 functional requirements of section 16 and our security requirements of section 6.5.2. These steps support the decision-making process of an appropriate encryption model for each attribute of a database. In the first steps (D1, D2, and D3) we assure that each attribute complies with our security requirements. This is done by determining whether encryption is required and if so, what limitations to encryption models apply. In step D4.1 and D4.2, we require that all following steps are followed for each functional requirements involved for every attribute to detect conflicting requirements. Through binary (Yes, No) encoding we enforce each route to lead through a targeted functional requirement ending in the applicable solution node. This path is then created by following "No" until the correct requirement is found.

For example, the attribute BSN is being evaluated in step D4.2 requiring functional requirements U1-F6 (Pattern Matching) and U2-F6 (Individual Replacement). This then leads to two trails starting from point D4.2, *(Path: D5.2, Yes, S5.2 : DET)* and *(Path: D5.2, No, D6.2, Yes, S5.2 : DET)* both ending in S5.2 (DET). This means that both functional requirements can be met using DET encryption. It is also possible for different paths of the same attribute to lead to different solutions. We refer to different solutions for the same attribute or solutions that end in S7 (Not Supported) as "Conflicting Paths". If a attributes has "Conflicting Paths" one or more functional requirements cannot be satisfied.

We applied the flowchart of figure 26 on the TLPRD assuring that our encryption model complies with functional requirements U1-F6, U2-F6, U3-F6 and U4-F6 and with confidentiality Requirement: CR2 (Horizontal, Vertical and Indirect Horizontal). This resulted in the TLPRD encryption model as provided in table 20.

Attribute	Private (Table 14)	Distinct (Table 15)	Required Functionality (Section 6.4.1) *(Path in Figure 26)*	Conflicting Paths	Encryption Model
NameOwner	Yes	No	Requirement U2-F6 :Support for individual replaceable elements. *(Path: P1,D2,P2,D4.1,D5.1,D9,S5.1* ***DET****)*	No	DET*
BSN	Yes	Yes	1. Requirement U2-F6 : Support for individual replaceable elements. *(Path: P1,D2,D3,P3,D4.2,D5.2,S5.2:* ***DET****.)* 2. Requirement U3-F6 :,Support for pattern matching (Defined in U1). *(Path: P1,D2,D3,P3,D4.2,D5.2,D6.2,S5.2:* ***DET****.)* 3. Requirement U4-F6 : Support for pattern matching (Defined in U1). *(Path : P1,D2,D3,P3,D4.2,D5.2,D6,2,S5.2:* ***DET****.)*	No	DET
LicenseplateNumber	No	Yes	1. Requirement U1-F6 :,Support for pattern matching. *(Path: P1,D2,D3,P3,D4.2, D5.2,D6.2,S5.2:* ***DET****.)* 2. Requirement U2-F6 :Support for pattern matching. *(Path: P1,D2,D3,P3,D4.2,D5.2,D6.2,S5.2:* ***DET****.)*	No	DET
Brand	No	No	None *(Path P1,D2,P2,D4.1,S2 :* ***RND****.)*	No	RND
Colour	No	No	None *(Path P1,D2,P2,D4.1,S2 :* ***RND****.)*	No	RND
CatalogPrice	No	No	Requirement U3-F6 Set 2 : Support additive homomorphism. *(Path: P1,D2,P2,D4.1,D5.1, D6.1,D7,S8 :* ***HOM****.)*	No	HOM
CO2Emission	No	No	1. Requirement U4-F6 Set 2 :,Support for additive homomorphism. *(Path: P1,D2,P2,D4.1,D5.1,D6.1,D7,D8, S8 :* ***HOM****.)* 2. Requirement U4-F6 Set 2 :,Support for counting of elements. *(Path: P1,D2,P2,D4.1,D5.1,D6.1,D7,D8, S8 :* ***HOM****.)*	No	HOM
APKDdate	No	No	None *(Path P1,D2,P2,D4.1,S2 :* ***RND****.)*	No	RND
Status	No	No	None *(Path P1,D2,P2,D4.1,S2 :* ***RND****.)*	No	RND
Legal	No	No	None *(Path P1,D2,P2,D4.1,S2 :* ***RND****.)*	No	RND

Table 20: Overview of the determination process of the appropriate encryption model for each attribute of the TLPRD, based on the flowchart of figure 26. All functional requirements for each attribute can be met without causing a conflict by either ending in S7 or having different path endings.

7.2.3.2 *DET* : RND Depended on a DET Attribute*

In figure 26 and table 20 we set out the encryption schemes that we propose to implement for the attributes of TLPRD. In section 5.5 we explained the DET, RND and HOM models, as defined by CryptDB. We found that exclusive use of these three models would not suffice to satisfy all functional and confidentiality requirements.

CR2-VERTICAL → RND

With only the use of RND, HOM and DET the problem arises that *NameOwner* (Private, Not distinct) would require encryption by RND in order to break all patterns between different people with the same names (e.g. Jan Jansen) as required by CR2-Vertical.

U2-F6 → OBTAIN RND'S IV

The problem of using RND on *NameOwner* is that this conflicts with functional requirement U2-F6, which states that *NameOwner* has to be individually replaceable. Values cannot be individually replaced under RND encryption as we have no random Initialization Vector (IV) to be used in RND's CBC.

F2-F8 → IV NOT STORED AT PROXY

This is the case because IVs are required to be hosted (Encrypted) in the cloud, as

we do desire a Proxy to have no more computational load than strictly necessary changes to the database. Functional requirement F2-F8 specifies this by stating that proxy performed encryptions have to be performable under a time limit, defined as a constant. This means that the time required to obtain an IV has to be independent of the number of elements in the database. It is therefore not possible for the proxy to host an IV list, as this would require a search complexity correlated to the number of rows in the DB (Complete DPLR contains several million rows).

CR-4 $\rightarrow$ IV NOT OBTAINABLE FROM CLOUD

The Proxy Server is also not allowed to request an IV for encryption purposes as set out in CR-4, as this would allow the cloud to manipulate the encryption process. We do not consider this a problem for decryption, from a confidentiality oriented point of view, since the cloud receives no direct feedback.

TAKING A NEW IV

RND can only obtain a new IV that is not retrieved from the cloud by by generating new IV for each time the attribute $NameOwner$ is being updated. This would require RND to have a IV specific for $NameOwner$ while preventing other RND encrypted attributes from being linked to the encrypted $NameOwner$ attribute. This could be considered a solution (RND*), but this would require the extension of the database with an additional IV column exclusive to one attribute. A larger database will require more upload and download time negatively affecting the time required for "Periodical DB Re-encryption" (section 7.2.2). A more efficient solution is the by us newly proposed DET* encryption model.

CONSTRUCTION OF DET*

DET* is RND with the adaptation of using another attribute as the initialization vector. Because $NameOwner$ is always updated in sync with BSN (DET) and requires no additional functionality. We can use DET* to give $NameOwner$ the same encryption pattern as BSN. Though entries with the same BSN number will still show the same NamerOwner encryption, there is no longer a correlation between people with the same name who have different BSN numbers. Having correlations based on a BSN number is acceptable (Compliance with CR2- Vertical), as these are unique for every individual person. This prevents leakage of personal data to other persons as they will have a different BSN number (unique). Decryption of DET* depends on the attribute to which it is linked (BSN). Functional requirement F6 states that this is always the case as $NameOwner$ and BSN are always updated as a pair (In sync). We can thus allow for $NameOwner$ to be encrypted using DET* with a BSN as DET link in this scenario. The advance over the previously mention RND* is that DET* requires no additional IV column to be added to the database, saving database space.

7.2.3.3 *Implementation*

In contrast to CryptDB's default implementation we do not require multiple onion layers for each attribute since we know all desired functionalities. Therefore, we propose only to

use four different implementations that we derived from CryptDB as illustrated in figure 27.

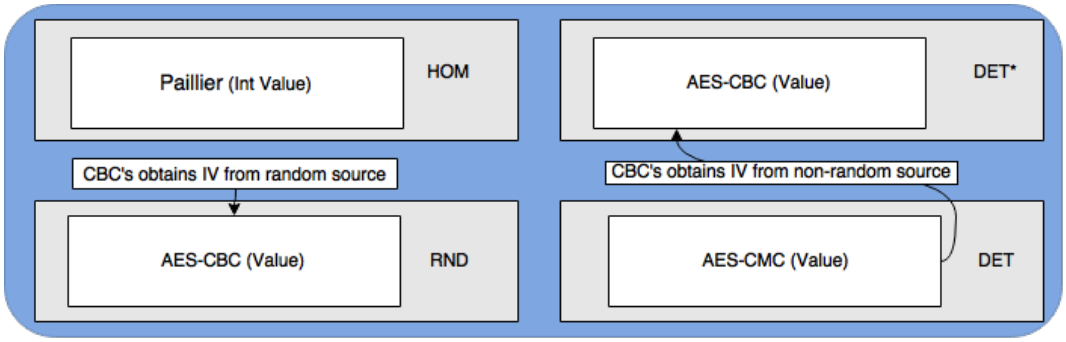

Figure 27: We propose to encrypt all attributes in the TLPRD (table 15) with the following encryption structures derived from CryptDB and labelled according to table 20. This means that we encrypt each row using a separate row key with either RND, DET, DET* or HOM. The IV of DET* is explained in 7.2.3.2 and for RND's IV we include one IV column in the DB for the first RND column, which in turn provides the IV for the next.

We deploy the Encryption Models of figure 27 on the TLPRD according to the model of table 20. To achieve a sufficient level of security according to CR-1, we require each encryption to have the following key length.

Encryption	Bits of Security	Secure Until	Encryption time	Decryption time
AES-CBC (100KB)	256 (Symmetric 256)	>2030	0.008 ms	0.007 ms
AES-CMC (100KB)	256 (Symmetric 256)	>2030	0.016 ms	0.015 ms
Pallier (32-bit integer)	112 (Factoring Modulus 2048)	2030	9.7 ms	0.7 ms

Table 21: Performance of AES-CBC, AES-CMC and Pallier achieved by a CryptDB's Proxy benchmark on a machine with eight 2.4 GHz AMD Opteron 8431 6-core processors and 64 GB of RAM [PRZB11b]. 'Secure Until' shows the until which year each encryption is deemed infeasible to break according to the NIST [Gir].

7.3 ANALYSIS OF OUR PROPOSED SOLUTION

In section 6 we set out use cases and confidentiality requirements that we designed for the TLPRD based on a problem description for the RDW. In sections 7.2.2 and 7.2.3 we set out how we would deploy a CryptDB inspired solution and what encryption structures we would recommend. In this section, we perform a theoretical analyse of how well this solution covers the previously stated requirements. We end this section with a short comparison of our solution to those previously described in section 7.2, and explain why we believe this solution to be preferable.

7.3.1 *Coverage of Confidentiality Requirements*

In table 22 we set out all confidentiality requirements of section 6.5 and how our proposed solution of section 7.2 satisfies them. We can thus conclude that our system is able to enforce sufficient confidentiality under the following (CA) assumptions:

CA-1 Encryption should be infeasible to break until 2030 (See CR-1).

CA-2 There is sufficient database traffic (Updates) in order to obfuscate a single update by performing them in a bench (See CR-3).

CA-3 Observing query patterns within the time frame of 1 day is acceptable (See CR-4: Multiple).

CA-4 Only queries are sent to the cloud according to our functional requirement.

Confidentiality Requirements	Satisfied	Motivation
CR-1 : Breaking Encryption	Yes	All used encryption's (RND, DET, DET* and HOM) are based on the discrete logarithm problem and are classified by NIST to be infeasible to break until at least 2030 when a key a relative key length of 256 bits is taken (See table 21)
CR-2 : Static Analysis	Yes (1+2+3)	Satisfied since the requirements of Vertical, Horizontal and Indirect Horizontal are met.
Vertical	Yes (1)	Satisfied since only the attribute BSN is deterministically (DET) encrypted (Table 20).
Horizontal	Yes (2)	Satisfied since all attributes in the TLPRD are encrypted (Table 20).
Indirect Horizontal	Yes (3)	Satisfied since there is no observable correlation between an encrypted public data and not encrypted public data. This is the case since RND and HOM reveal no correlation and DET* (7.2.3.2) only correlates to the private attribute BSN (Table 20).
CR-3 : Dynamic Analysis	Yes	The Unencrypted DB and Encrypted DB can be linked through query patterns. This is the case since updates only occur in real time on the Encrypted DB (Updating the Unencrypted DB later in batch) obfuscating correlations (Section 7.2.2).
CR-4 : Query Analysis	Yes (4+5+6+7)	Satisfied since the requirements of Single Token, Single Generation, Single Execution and Multiple Queries are met.
Single: Token	Yes (4)	Satisfied since all F2 functional requirements of each use case require a query to the cloud to have no plaintext variables (Section 16).
Single: Generation	Yes (5)	Satisfied since all F5 functional requirements of each use case require a queries to the cloud the be generated by the proxy without information provided by the cloud (Section 7.2.3.1).
Single: Execution	Yes (6)	Satisfied since all F6 functional requirements of each use case provides no conflict with other requirements as shown in table 20.
Multiple Queries	Yes (7)	Satisfied since leakage through observing multiple queries over a longer period of time of by re-encryption as explained in section 7.2.2
CR-5 : Key Theft	Yes	Satisfied since all F6 functional requirements can be satisfied without requiring a secret key (Table 7.2.3.1). No secret key isthereforerequired to be located outside the trusted environment (Proxy).

Table 22: Overview of all confidentiality requirements of section 6.5 and how our solution of section 7.2 satisfies them.

7.3.2 *Coverage of Functional Requirements*

In table 23 we set out all functional requirements of section 6 and how our proposed solution of section of section 7.2 satisfies them. We can thus conclude that our system is able to perform all desired functionality with sufficient processing efficiency under the (FA) assumptions that:

FA-1 Client, Proxy, and Cloud support MySQL.

FA-2 The Proxy knows all secret keys.

FA-3 The Proxies hardware's performance is equal or greater than revered to in table 21.

FA-4 It is acceptable for the cloud to perform queries in a time complexity linear to the size of the database.

FA-5 It is acceptable to prevent update queries during a limited time (e.g. 1 hour) each day (See section 7.2.2).

FA-6 It is acceptable for the public database to be at most one day out of date with the encrypted database (section 7.2.2).

Use Case 1 : Search			
Functional Requirement	**Type**	**Satisfied**	**Motivation**
U1-F1 : Query	Format	Yes	Client is assumed to have MySQL.
U1-F2 : Encrypted Query	Format	Yes	Proxy has MySQL and U1-F5 allows for U1-F2 given U1-F1.
U1-F3 : Encrypted Result	Format	Yes	Azure supports MySQL and U1-F6 allows for U1-F3 given U1-F2.
U1-F4 : Result	Format	Yes	Proxy has MySQL and U1-F7 allows for U1-F4 given U1-F3.
U1-F5 : Proxy Encryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U1-F1.
U1-F6 : DB Functionality	Functionality	Yes	Azure is able to perform pattern matches on LicenSePlateNumber and BSN (DET) attributes.
U1-F7 : Proxy Decryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U1-F3.
U1-F8 : Proxy Encryption	Time	Yes	U1-F5 requires 2 DET encryption's. Given table 18 this is feasible in under 0.1 second.
U1-F9 : DB Functionality	Time	Yes	Azure can pattern match on DET columns in linear time to the amount of database rows.
U1-F10 : Proxy Decryption	Time	Yes	U1-F7 requires at most 4 decryption operation at once. The most expensive of these operations is HOM. Given table 18 it is feasible to decipher four time a HOM encrypted value in under 0.1 second.

Use Case 2 : Update			
Functional Requirement	**Type**	**Satisfied**	**Motivation**
U2-F1 : Query	Format	Yes	Client is assumed to have MySQL.
U2-F2 : Encrypted Query	Format	Yes	Proxy has MySQL and U2-F5 allows for U2-F2 given U2-F1.
U2-F3 :Encrypted Result	Format	Yes	No requirement
U2-F4 : Result	Format	Yes	No requirement
U2-F5 : Proxy Encryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U2-F1.
U2-F6 : DB Functionality	Functionality	Yes	Azure is able to replace encrypted $NameOwner$ (DET*) and BSN (DET) values without breaking any assumptions. (See also section,8.2.2). The only dependency in TLPRD regarding $NameOwner$ and BSN is that each $NameOwner$ entry is dependent on 1 BSN value of same row. Updates of $NameOwner$ and BSN are paired (U2-F5) preventing distortion of this decency.
U2-F7 : Proxy Decryption	Functionality	Yes	No requirement
U2-F8 : Proxy Encryption	Time	Yes	U2-F2 requires the encryption of one DET* and DET value. Given table 18 this is feasible in under 0.1 second.
U2-F9 : DB Functionality	Time	Yes	U2-U9 performance is the sum of U1-U9 combined with the constant time it takes to replace 2 entries. Since U1-U9 is linear in time, so is U2-U9.
U2-F10 : Proxy Decryption	Time	Yes	No requirement

Use Case 3 : Summation			
Functional Requirement	**Type**	**Satisfied**	**Motivation**
U3-F1 : Query	Format	Yes	Client is assumed to have MySQL.
U3-F2 : Encrypted Query	Format	Yes	Proxy has MySQL and U3-F5 allows for U3-F2 given U3-F1.
U3-F3 : Encrypted Result	Format	Yes	Azure supports MySQL and U3-F7 allows for U3-F4 given U3-F3.
U3-F4 : Result	Format	Yes	Proxy has MySQL and U3-F7 allows for U3-F4 given U3-F3.
U3-F5 : Proxy Encryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U3-F1. The proxy is able to change SUM to a MySQL compatible multiplicative variant as this is support by MySQL.
U3-F6 : DB Functionality	Functionality	Yes	Azure is able to perform a summation on the encrypted CatalogPrice (HOM) as multiplication (Section 5.5.1.2) is (indirectly) supported in MySQL
U3-F7 : Proxy Decryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U3-F3.
U3-F8 : Proxy Encryption	Time	Yes	U3-F2 required the encryption of one HOM and BSN value. Given table 18 this is feasible in under 0.1 second.
U3-F9 : DB Functionality	Time	Yes	U3-U6 time complexity is the sum of U3-U9 with the time complexity of counting the elements in SUM (U3-F5) a number of elements equal or less the number database rows. Since both are compliant with F9 so is their sum U3-F9.
U3-F10 : Proxy Decryption	Time	Yes	U3-F7 required the decryption of 1 summation value encrypted with HOM. Given table 18 this is feasible in under 0.1 second.

Use case 4 : Average			
Functional Requirement	**Type**	**Satisfied**	**Motivation**
U4-F1 : Query	Format	Yes	Client is assumed to have MySQL.
U4-F2 : Encrypted Query	Format	Yes	Proxy has MySQL and U4-F5 allows for U4-F2 given U4-F1.
U4-F3 : Encrypted Result	Format	Yes	Azure supports MySQL and U4-F7 allows for U4-F4 given U4-F3.
U4-F4 : Result	Format	Yes	Proxy has MySQL and U4-F7 allows for U4-F4 given U4-F3.
U4-F5 : Proxy Encryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U4-F1. The proxy is able to change AVG to a MySQL compatible combination of SUM and COUNT.
U4-F6 : DB Functionality	Functionality	Yes	U4-F6 (Set 1) Azure is able to perform a summation and count on HOM (See also U3-F6)
U4-F7 : Proxy Decryption	Functionality	Yes	Proxy knows all secret keys and requires no additional information besides U4-F3.
U4-F8 : Proxy Encryption	Time	Yes	U4-F2 required the encryption of 1 CO2Emission (HOM) and 1 BSN (DET) value. Given table 18 this is feasible in under 0.1 second.
U4-F9 : DB Functionality	Time	Yes	U4-U6 time complexity is the sum of U1-U9 combined with the time complexity of counting (U4-F5) the number of elements in SUM. Since both are compliant with F9 so is their sum U4-F9.
U4-F10 : Proxy Decryption	Time	Yes	U4-F7 required the decryption of 1 summation value (HOM). Given table 18 this is feasible in under 0.1 second.

Table 23: Overview of all functional requirements of section 6 and how our solution of section 7.2 satisfies them.

8

CONCLUSION

In this thesis, we presented an overview of models that enforce data confidentiality in a cloud computing scenario. From this overview, we concluded that these models are insufficient to process license plate data because none of them provide the confidentiality guarantees or functionality required for our problem sketch of the RDW. We constructed a new solution based on CryptDB that can be used to provide both the needed confidentiality and functional requirements while being compliant with Dutch data protection law. For the following two research sub-question, we summarize the legal boundaries regarding personal data and cryptographic possibilities for the RDW. Our main research answer then combines these sub-answers explaining that dependent on the type of attributes encryption methods can be used to securely process Dutch personal data in the cloud, allowing for at least search and update queries and including the ability to calculate summations and an average.

Sub-1: Can encryption methods be used to allow the processing Dutch personal data in the cloud from a legal perspective?

In Section 7.1 we explain that it is indeed possible to use encryption methods to allow the processing of Dutch personal data in the cloud. The Dutch law Wet Bescherming Persoongegeven (WBP) requires two conditions for this to be met. Condition one is that Dutch personal data may only be placed or processed by a cloud provider that complies with EU Directive 95/46/EC. A common way for cloud providers to show compliance with EU Directive 95/46/EC is by obtaining either a Safe Harbor or EU Model Clause certification. This is however not sufficient as the EU commission's Article 29 Data Protection Working Party's (A29WP) on cloud computing and the CBS [Jus15] states that an organization has to verify whether a cloud provider upholds the Directive 95/46/EC. One of the suggested ways this can be achieved is by securely encrypting all personal data, leaving the definition of secure to be determend by the data providing party. A newly upcoming European law (GDPR), to be taken into effect in 2016, specifies this encryption requirement by stating that secure encryption models may not share its security key with a cloud provider, excluding cloud maintained encryption services.

Sub-2: Is it feasible for the RDW to use encryption to securely process their personal data in the cloud?

We consider it feasible for the RDW to securely process personal data in the cloud using encryption. This conclusion is based on the fact that we were able to construct a theoretical model, based on the combination of multiple encryption schemes, that satisfies all requirements and assumptions made in Section iv. These requirements are based on the need to perform database search and update operations extended with the ability to calculate summations and an average. We believe this model to be efficient as a proxy is able to encryption or decryption any of the given queries in under 0.1 second and the cloud is still able to perform all operation in a time linear to the number of database entries.

Main: To what extent can current encryption methods be applied, in practice, to enforce data confidentiality of Dutch personal data processed in the cloud?

We conclude from our literature review that there are functional fully homomorphic encryption methods that allow for data to be manipulated without limiting the amount and type of operations that can be performed under encryption. These methods are however infeasible for any practical implication regarding real-time database traffic, due to their time complexity and computational overhead. Other methods that rely on secure hardware or the combination of multiple encryption models like CryptDB have proven to be able to support a wide variety of MySQL statements and are able to perform at a limited overhead in the order of 25 percent compared to a non-encrypted MySQL implementation. Though methods like CryptDB suffice in basic database needs, they are unable to efficiently perform fully homomorphic and require computational support from a trusted environment when combinations of additive and multiplicative are used. However, we can conclude that given a well-defined framework for operational needs, these type of models can be a feasible solution, depending on the desired functionality.

From a confidentiality oriented perspective, we found that all investigated encryption models prevented the revealing of plaintext values. However, these models did not include sufficient safeguards against data analysis attacks, which form a risk in the case of combined public and private information. We, therefore, constructed a new encryption model and concluded that two countermeasures have to be implemented. The first countermeasure requires that an encrypted database gets re-encrypted periodically to limit queries from forming patterns over time. The second countermeasure require the limitation of deterministic encryption based on the meaning and entropy of an attribute, preventing the plaintexts of ciphers to be derived through deductive reasoning. In our solution, we showed that these additional confidentiality requirements can be satisfied using the attribute depended on encryption structure exploited by CryptDB. Encryption models can, therefore, suffice as enforcement of the confidentiality of Dutch personal data in a cloud computing scenario, allowing for at least search and update queries, including the ability to calculate summations and an average.

9

FUTURE WORK

PROOF OF CONCEPT

In this research, we have shown that a proxy based encryption model using attribute dependent on encryption can efficiently enforce confidentiality in a cloud computing scenario involving both public and private data. However, due to time constraints we were unable to implement this solution as a proof of concept. Further development of this solution, therefore, requires the testing of this model in terms of performance under desired hardware constraints.

GENERALIZATION

The research in this thesis focused on a specific database and a defined set of use cases. Further research is required to generalize the proposed solution to formalize a general encryption model, adaptive to both required functionally and meaning of the handled data. For this, we like to refer to both our model, for its framework regarding confidently, and that proposed by CryptDB, for its adaptive onion encryption structure. An extended framework formalizing a general encryption model should also include further research to all risks and assumptions regarding the analyses of encrypted data. This research is needed to provider provable guarantees in generalized scenario's, as we considered all aspect that are not relevant to our specific scenario to be out of the scoop.

SUPPORT

In this research, we spent a lot of time trying to implement CryptDB's open source implementation. Due to limited support, compatibility and documentation we encountered several complications when trying to setup and run CryptDB. After several weeks, we were able to setup and configure CryptDB using virtual machines running Linux, of which we included a manual and link to our VMware image in Dropbox [Sli15]. A future implementation of a proxy based cryptographic solution is advised to include a clear tutorial and provide a higher level of compatibility by being based on well (cross platform) supported programming languages and packages.

Part VI

BIBLIOGRAPHY

BIBLIOGRAPHY

[AAC15] Mohamed Alsharnouby, Furkan Alaca, and Sonia Chiasson. Why phishing still works: user strategies for combating phishing attacks. *International Journal of Human-Computer Studies*, 2015.

[ABB+07] Nicolas Anciaux, Mehdi Benzine, Luc Bouganim, Philippe Pucheral, and Dennis Shasha. Ghostdb: querying visible and hidden data without leaks. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 677–688. ACM, 2007.

[Act01] Patriot Act. Uniting and strengthening america by providing appropriate tools required to intercept and obstruct terrorism (usa patriot act) act of 2001. *Public Law*, 107:56, 2001.

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[AG06] Larry Abramson and Maria Godoy. The patriot act: Key controversies. *National Public Radio. Online: http://www. npr. org/news/specials/patriotact/patriotact provisions. html*, 2006.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.

[Alb12] JP Albrecht. Draft report on the proposal for a regulation of the european parliament and of the council on the protection of individual with regard to the processing of personal data and on the free movement of such data (general data protection regulation)(com (2012) 0011-c7-0025/2012–2012/0011 (cod)). *European Parliament, Committee on civil liberties, justice and home affairs*, 17, 2012.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. Cryptology ePrint Archive, Report 2014/094, 2014. `http://eprint.iacr.org/`.

[Aut14] Dutch Motor Vehicle Authority. Homepage rijks dienst voor wegverkeer, Dec 2014. URL: `http://www.rdw.nl` [cited 10.08.2015].

[AVL62] GM Adelson-Velskii and Evgenii Mikhailovich Landis. An information organization algorithm. In *Doklady Akademia Nauk SSSR*, volume 146, pages 263–266, 1962.

[Azu] Microsoft Azure. Eu model clauses. URL: `http://azure.microsoft.com/en-gb/support/trust-center/compliance/eu-model/` [cited 21.06.2015].

[Bar11] Chiesa Alessandro Barak, Boaz. Computing blindfolded: New developments in fully homomorphic encryption. In *New Developments in Cryptography*. Boston University, 2011.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009*, pages 224–241. Springer, 2009.

[BFK+96] Aiden A Bruen, Mario A Forcinito, Alan G Konheim, Chey Cobb, Adam Young, Moti Yung, and David Hook. Applied cryptography: protocols, algorithms, and source code in c. 1996.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography*, pages 325–341. Springer, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *Advances in Cryptology-CRYPTO'94*, pages 341–358. Springer, 1994.

[Bod12] Irene Bodle. Eu data protection law and the patriot act in the cloud, 2012. URL: `http://www.webanalyticsworld.net/2012/03/eu-data-protection-law-and-the-patriot-act-in-the-cloud.html` [cited 11.04.2015].

[Boo13] Shelly Boose. Cloud computing adoption by federal agencies increases 400%, December 2013. URL: `http://www.tripwire.com/state-of-security/latest-security-news/cloud-computing-adoption-federal-agencies-increases-400/` [cited 12.07.2015].

[BP02] Luc Bouganim and Philippe Pucheral. Chip-secured data access: Confidential data on untrusted servers. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 131–142. VLDB Endowment, 2002.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Report 2011/344, 2011. `http://eprint.iacr.org/`.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *EUROCRYPT*, volume 7881, pages 315–335. Springer, 2013.

[Che14] Li Chen. Multikey homomorphic encryption from ntru, 2014.

[CLHK11] I-Hsun Chuang, Syuan-Hao Li, Kuan-Chieh Huang, and Yau-Hwang Kuo. An effective privacy protection scheme for cloud computing. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 260–265. IEEE, 2011.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology–CRYPTO 2011*, pages 487–504. Springer, 2011.

[CNS99] Jean-Sébastien Coron, David Naccache, and Julien P Stern. On the security of rsa padding. In *Advances in Cryptology-CRYPTO'99*, pages 1–18. Springer, 1999.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2012*, pages 446–464. Springer, 2012.

[Cod82] Edgar F Codd. Relational database: a practical foundation for productivity. *Communications of the ACM*, 25(2):109–117, 1982.

[Col] Cameron Coles. Only 1 in 100 cloud providers meet proposed eu data protection requirements. URL: `https://www.skyhighnetworks.com/cloud-security-blog/only-1-in-100-cloud-providers-meet-proposed-eu-data-protection-requirements/` [cited 20.05.2015].

[Coma] European Commission. Factsheet on the "right to be forgotten" ruling (c-131/12). URL: `http://ec.europa.eu/justice/data-protection/files/factsheets/factsheet_data_protection_en.pdf` [cited 13.05.2015].

[Comb] European Commission. Model contracts for the transfer of personal data to third countries. URL: http://ec.europa.eu/justice/data-protection/document/international-transfers/transfer/index_en.htm [cited 31.05.2015].

[Com12] European Commision. Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation). *COM (2012) 11 final, 2012/0011 (COD), Brussels, 25 January 2012*, 2012.

[Coo05] Katherine K Coolidge. Baseless hysteria: The controversy between the department of justice and the american library association over the usa patriot act. *Law Libr. J.*, 97:7, 2005.

[DA] Marc Dautlich and Stephan Appt. Data protection officers. will eu businesses face an obligation to appoint one? URL: http://www.out-law.com/en/articles/2015/january/data-protection-officers--will-eu-businesses-face-an-obligation-to-appoint-one/ [cited 20.05.2015].

[Dav66] Kahn David. The codebreakers: the story of secret writing.., 1966.

[Des00] Anand Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In *Advances in Cryptology-CRYPTO 2000*, pages 394–412. Springer, 2000.

[dG] Jochem de Groot. Artikel 29: Azure voldoet aan de strenge eu privacy wetgeving. URL: http://www.azureblog.nl/tag/eu-model-clause/ [cited 31.05.2015].

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[Dir95] EU Directive. 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the EC*, 23(6), 1995.

[DLS+13] Chyi-Ren Dow, Cheng-Min Lin, Waleed W Smari, Chien-Chung Wu, and Kuo-Kun Tseng. Ict innovations in future smart cars. *International Journal of Vehicular Technology*, 2013, 2013.

[DPUotDGfJS] Freedom Data Protection Unit of the Directorate General for Justice and Security. Frequently asked question relating to transfers of personal data from the eu.eea to thirth countries. URL: http://ec.europa.eu/justice/policies/privacy/docs/international_transfers_faq/international_transfers_faq.pdf [cited 16.06.2015].

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2002.

[Ebe93] Hans Eberle. A high-speed des implementation for network applications. In *Advances in Cryptology-CRYPTO'92*, pages 521–539. Springer, 1993.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[EMST78] William F Ehrsam, Carl HW Meyer, John L Smith, and Walter L Tuchman. Message verification and transmission error detection by block chaining, February 14 1978. US Patent 4,074,066.

[ERB03] Mohamed Y Eltoweissy, Abdelmounaam Rezgui, and Athman Bouguettaya. Privacy on the web: Facts, challenges, and solutions. *IEEE Security & Privacy*, 1(6):0040–49, 2003.

[Exp] Export.gov. Welcome to the u.s.-eu safe harbor. URL: `http://www.export.gov/safeharbor/eu/eg_main_018365.asp` [cited 21.05.2015].

[Fra] Natacha Franke. Update on eu data protection regulation: One-stop-shop and general principles. URL: `http://www.considerati.com/blog/update-on-eu-data-protection-regulation-one-stop-shop-and-general-principles/` [cited 20.05.2015].

[Fra87] Benjamin Franklin. *Poor Richard's Almanack, 1733–1758*. Benjamin Frank, 1987.

[ftAiGS14] IAGS Institute for the Analysis if Global Security. How much did the september 11 terrorist attack cost america?, 2014. URL: `http://www.iags.org/costof911.html` [cited 10.04.2015].

[G+13] Top Threats Working Group et al. The notorious nine: cloud computing top threats in 2013. *Cloud Security Alliance*, 2013.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *Advances in Cryptology–CRYPTO 2010*, pages 116–137. Springer, 2010.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology-CRYPTO'97*, pages 112–131. Springer, 1997.

[GH11] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology–EUROCRYPT 2011*, pages 129–148. Springer, 2011.

[GHS11] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. Cryptology ePrint Archive, Report 2011/566, 2011. http://eprint.iacr.org/.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography–PKC 2012*, pages 1–16. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology–EUROCRYPT 2012*, pages 465–482. Springer, 2012.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[GHS12d] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[Gir] Damien Giry. Bluekrypt cryptographic key length recommendations. URL: http://www.keylength.com/en/4/ [cited 27.06.2015].

[GLM01] Barbara Crutchfield George, Patricia Lynch, and Susan J Marsnik. Us multinational employers: Navigating through the âĂIJsafe harborâĂİ principles to comply with the eu data privacy directive. *American Business Law Journal*, 38(4):735–783, 2001.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[GMP13] Glenn Greenwald, Ewen MacAskill, and Laura Poitras. Edward snowden: the whistleblower behind the nsa surveillance revelations. *The Guardian*, 9:2013, 2013.

[Gol04] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2004.

[Gre] Andy Greenberg. An mit magic trick: computing on encrypted databases without ever decrypting them. URL: http://www.forbes.com/sites/andygreenberg/2011/12/19/ [cited 17.04.2015].

[Gun02] Rohan Gunaratna. *Inside Al Qaeda: global network of terror*. Columbia University Press, 2002.

[HBB13] Andreas Hülsing, Christoph Busold, and Johannes Buchmann. Forward secure signatures on smart cards. In *Selected Areas in Cryptography*, pages 66–80. Springer, 2013.

[Hen01] Mike Hendry. *Smart card security and applications*. Artech House, Inc., 2001.

[HM11] Darrel Hankerson and Alfred Menezes. Elliptic curve discrete logarithm problem. In *Encyclopedia of Cryptography and Security*, pages 397–400. Springer, 2011.

[HMMT14] Jan Hajny, Lukas Malina, Zdenek Martinasek, and Ondrej Tethal. Performance evaluation of primitives for privacy-enhancing cryptography on current smart-cards and smart-phones. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 17–33. Springer, 2014.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic number theory*, pages 267–288. Springer, 1998.

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *Advances in Cryptology-CRYPTO 2003*, pages 482–499. Springer, 2003.

[HS14a] Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014*, pages 554–571. Springer, 2014.

[HS14b] Shai Halevi and Victor Shoup. Helib-an implementation of homomorphic encryption, 2014.

[ISO] ISO. *ISO/IEC 27017 - Information technology - Security techniques - Code of practice for information security controls based on ISO/IEC 27002 for cloud services (DRAFT)*. International Organization for Standardization and International Electrotechnical Commission. URL: `http://www.iso27001security.com/html/27017.html`.

[ISO13] ISO. *ISO/IEC 27002:2013 Information technology - Security techniques - Code of practice for information security controls*. International Organization for Standardization and International Electrotechnical Commission, 2013. URL: `http://www.iso27001security.com/html/27002.html`.

[ISO14] ISO. *ISO/IEC 27018:2014 Information technology - Security techniques - Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors*. International Organization for Standardization and International Electrotechnical Commission, 2014. URL: `http://www.iso.org/iso/catalogue_detail.htm?csnumber=61498`.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.

[Jus15] Justitia.nl. Safe harbor, 2015. URL: `http://www.justitia.nl/privacy/safeharbor.html` [cited 30.07.2015].

[JVJ12] Bansidhar Joshi, A Santhana Vijayan, and Bineet Kumar Joshi. Securing cloud computing environment against ddos attacks. In *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, pages 1–5. IEEE, 2012.

[Kal09] Shalinie Kalika. *" De electronische ontwikkeling van DigiD met betrekking tot het persoonsnummerbeleid": advies aan GBO. Overheid (Ministerie van Binnenlandse Zaken en Koninkrijksrelaties)*. Erasmus Universiteit, 2009.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology-CRYPTO'99*, pages 388–397. Springer, 1999.

[KV10] Ronald L Krutz and Russell Dean Vines. *Cloud security: A comprehensive guide to secure cloud computing*. John Wiley & Sons, 2010.

[LATV13] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. Cryptology ePrint Archive, Report 2013/094, 2013. `http://eprint.iacr.org/`.

[Lel13] Jakob Lell. Practical malleability attack against cbc-encrypted luks partitions, December 2013. URL: `http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions/` [cited 30.07.2015].

[lg] M law group. New draft european data protection regime. URL: `http://mlawgroup.de/news/publications/detail.php?we_objectID=227` [cited 21.05.2015].

[LV11] Benoit Libert and Damien Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re- Encryption. *IEEE Transactions on Information Theory*, 57:1786–1802, 3 2011.

[Mao03] Wenbo Mao. *Modern cryptography: theory and practice*. Prentice Hall Professional Technical Reference, 2003.

[Mar14] Microsoft Azure Marktplace. Voertuig open data, 2014. URL: `https://datamarket.azure.com/dataset/opendata.rdw/vrtg.open.data` [cited 10.08.2015].

[Mat05] Ulf T Mattsson. Database encryption-how to balance security with performance. *Available at SSRN 670561*, 2005.

[MCG08] Carlos Aguilar Melchor, Guilhem Castagnos, and Philippe Gaborit. Lattice-based homomorphic encryption of vector spaces. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 1858–1862. IEEE, 2008.

[Mer78] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[MG09] Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[MG11] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[Mic01] Daniele Micciancio. Improving lattice based cryptosystems using the hermite normal form. In *Cryptography and Lattices*, pages 126–145. Springer, 2001.

[Moo15] Susan Moore. Gartner says worldwide cloud infrastructure-as-a-service spending to grow 32.8 percent in 2015, May 2015. URL: http://www.gartner.com/newsroom/id/3055225 [cited 26.07.2015].

[Mor13] Liam Morris. Analysis of partially and fully homomorphic encryption. *Department of Computer Science, Rochester Institute of Technology, Rochester, New York*, 2013.

[Mys] Mysql.com. 12.16.1 group by (aggregate) functions. URL: https://dev.mysql.com/doc/refman/5.0/en/group-by-functions.html [cited 12.07.2015].

[Nag] Floortje Nagelkerke. Strengthening sanctions for violation of the dutch data protection act. URL: http://www.nortonrosefulbright.com/knowledge/publications/124016/ [cited 28.05.2015].

[NEN13] NEN,ISO,IEC. *NEN-ISO/IEC 27001:2013 Information technology - Security techniques - Information security management systems - Requirements*. NEN, 2013. URL: https://www.nen.nl/NEN-Shop/Norm/NENISOIEC-270012013-en.htm.

[NTTM15] Mohsin Nazir, Prashant Tiwari, Shakti Dhar Tiwari, and Raj Gaurav Mishra. Cloud computing: An overview. *Book Chapter of Cloud Computing: Reviews, Surveys, Tools, Techniques and Applications-An Open-Access eBook published by HCTL Open*, 2015.

[NZMK15] Aws Naser, Mohamad Fadli Zolkipli, Mazlina Abdul Majid Mohamad, and Nusrat Ullah Khan. Security scheme for protecting cloud computing services against bursty ddos attacks. *Advances in Information Sciences and Service Sciences*, 7(1):39, 2015.

[oC] US Department of Commerce. 2000/520/ec: Commission decision of 26 july 2000 pursuant to directive 95/46/ec of the european parliament and of the council (safe harbor principle). URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000D0520:EN:HTML [cited 20.05.2015].

[oJ01] Department of Justice. Highlights of the usa patriot act, 2001. [Online; accessed 11-April-2015]. URL: http://www.justice.gov/archive/ll/highlights.html.

[oST93] National Institute of Standards and Technology. Data encryption standard. Technical Report NIST FIPS PUB 46-2, âĂİ U.S. Department of Commerce, dec 1993.

[oT11] Massachusetts Institute of Technology. Cryptdb, 2011. URL: https://css.csail.mit.edu/cryptdb/#Software [cited 14.04.2015].

[oT15] Massachusetts Institute of Technology. Cryptdb, 2015. URL: http://css.csail.mit.edu/cryptdb/ [cited 10.08.2015].

[PAB+05] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *Progress in Cryptology-INDOCRYPT 2004*, pages 61–72. Springer, 2005.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology-EUROCRYPT'99*, pages 223–238. Springer, 1999.

[Par] The European Parliament. European parliament legislative resolution of 12 march 2014 on the proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation) (com(2012)0011 c7-0025/2012 2012/0011(cod)). URL: http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+TA+P7-TA-2014-0212+0+DOC+XML+V0//EN [cited 10.05.2015].

[Par10] European Parliament. Charter of fundamental rights of the european union (2000/c 364/01). Official Journal of the European Communities, 2010.

[Per] Wet Bescherming Persoonsgegevens. Retrieved from overheid website: http://wetten. overheid. nl. $BWBR0011468/geldigheidsdatum_23-04-2015$.

[Per00] Wet Bescherming Persoonsgegevens. Wet van 6 juli 2000, houdende regels inzake de bescherming van persoonsgegevens (wet bescherming persoonsgegevens), 2000. URL: http://wetten.overheid.nl/BWBR0011468/geldigheidsdatum_24-05-2015.

[PLZ13] Raluca A Popa, Frank H Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 463–477. IEEE, 2013.

[PP05] Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Selected Areas in Cryptography*, pages 182–197. Springer, 2005.

[PRZB11a] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.

[PRZB11b] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.

[PZ12] Raluca Ada Popa and Nickolai Zeldovich. Cryptographic treatment of cryptdb's adjustable join. 2012.

[PZB11] Raluca Ada Popa, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: A practical encrypted relational dbms. 2011.

[Rad] Chris Radburn. Google privacy changes break dutch data protection law, says regulator. URL: http://www.theguardian.com/technology/2013/nov/29/dutch-data-privacy-google-breaks-accused [cited 19.05.2015].

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Ram] Thanos Rammos. Passing clouds: The ec's data protection reform plans and their relevance for cloud computing. URL: http://www.taylorwessing.com/globaldatahub/article_passing_clouds.html [cited 20.05.2015].

[RDWa] RDW. Uw gegevens in het kentekenregister inzien. URL: https://www.rdw.nl/Particulier/Paginas/Uw-gegevens-in-het-kentekenregister-inzien-.aspx [cited 27.04.2015].

[RDWb] RDW. Voertuig open data. URL: https://datamarket.azure.com/dataset/opendata.rdw/vrtg.open.data [cited 27.04.2015].

[RE10] Wolfgang Rankl and Wolfgang Effing. *Smart card handbook*. John Wiley & Sons, 2010.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM. URL: http://doi.acm.org/10.1145/1060590.1060603, doi:10.1145/1060590.1060603.

[Rel] Google Investor Relations. 2015 financial tables. URL: https://investor.google.com/financial/tables.html [cited 19.05.2015].

[Res13] 451 Research. Enterprise cloud computing poised for explosive growth during next two years, September 2013. URL: https://451research.com/images/stories/Marketing/press_releases/cloud_wave_5_press_release_final.pdf [cited 12.07.2015].

[Rij14] Informatie Rijksoverheid. Beleidsregels gevoelige gegevens kentekenregister, December 2014. URL: https://zoek.officielebekendmakingen.nl/stcrt-2009-1572.html [cited 10.08.2015].

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RV05] Alexandre Ruiz and Jorge Luis Villar. Publicly verfiable secret sharing from paillier's cryptosystem. *WEWoRC*, 74:98–108, 2005.

[Rya11] Mark D. Ryan. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, 54(1), 2011.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 543–553. IEEE, 1999.

[Sav] Eric Savitz. Can european firms legally use u.s. clouds to store data? URL: `http://www.zdnet.com/article/google-admits-patriot-act-requests` [cited 11.04.2015].

[Sch94a] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption*, pages 191–204. Springer, 1994.

[Sch94b] Paul M Schwartz. European data protection law and restrictions on international data flows. *Iowa L. Rev.*, 80:471, 1994.

[Sen13] Jaydip Sen. Homomorphic encryption: Theory & applications. *arXiv preprint arXiv:1305.5886*, 2013.

[Sha49] Claude E Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949.

[Sha00] Gregory Shaffer. Globalization and social protection: the impact of eu and international rules in the ratcheting up of us data privacy standards. *Yale Journal of International Law*, 25:1–88, 2000.

[Sli15] Vincent Slieker. Vmware image of preconfigured cryptdb and made tutorial, 2015. URL: `https://www.dropbox.com/sh/gahn6ijhbcy0k9f/AABGsDn5jcOUXMkTCKs7zfBAa?dl=0` [cited 20.08.2015].

[soFRotEC12] The secretariat of Fundamental Rights and Union Citizenship of the European Commission. Opinion 05/2012 on cloud computing, 2012. URL: `http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2012/wp196_en.pdf` [cited 13.05.2015].

[SR01] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *Advances in Spatial and Temporal Databases*, pages 79–96. Springer, 2001.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *Advances in Cryptology-ASIACRYPT 2010*, pages 377–394. Springer, 2010.

[Sul] Bob Sullivan. 'la difference' is stark in eu, u.s. privacy laws. URL: `http://www.nbcnews.com/id/15221111/ns/technology_and_science-privacy_lost/t/la-difference-stark-eu-us-privacy-laws/#.VV3oyk-qqko` [cited 19.05.2015].

[SV10] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography–PKC 2010*, pages 420–443. Springer, 2010.

[SWP00] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[Szy04] Michael Szydlo. Merkle tree traversal in log space and time. In *Advances in Cryptology-EUROCRYPT 2004*, pages 541–554. Springer, 2004.

[TEHEG12] Maha Tebaa, Saïd El Hajji, and Abdellatif El Ghazi. Homomorphic encryption applied to the cloud computing security. In *Proceedings of the World Congress on Engineering*, volume 1, pages 4–6, 2012.

[tH] Elze 't Hart. Wetsvoorstel: Hoge boetes bij schending privacy. URL: `http://www.vbk.nl/kennis-delen/actualiteiten/wetsvoorstel-hoge-boetes-bij-schending-privacy/` [cited 28.05.2015].

[TOM08] B T OGRAPH and Y RICHARD MORGENS. Cloud computing. *Communications of the ACM*, 51(7), 2008.

[Tri] Loic Triger. Overview of the eu initiative to simplify data protection in 2015. URL: `http://www.techradar.com/news/internet/policies-protocols/changes-in-european-data-protection-regulation-a-look-at-the-gdpr-1278235` [cited 02.06.2015].

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[VHAvEK12] JVJ Van Hoboken, Mr AM Arnbak, NANM van Eijk, and NPH Kruijsen. Cloud diensten in hoger onderwijs en onderzoek en de usa patriot act. *Rapport in opdracht van SURF, published on www. ivir. nl*, 2012.

[vV] PJ van Vlaanderen. Architectuurdocumentatie evaluatie.

[Weg13] Rijksdienst Wegverkeer. Elektronische informatieverstrekking, 2013. URL: `http://jaarverslag2013.rdw.nl/Paginas/%28Elektronische%29-informatieverstrekking.aspx` [cited 10.08.2015].

[Weg14] Rijksdienst Wegverkeer. Uw gegevens in het kentekenregister inzien, 2014. URL: `https://www.rdw.nl/Particulier/Paginas/Uw-gegevens-in-het-kentekenregister-inzien-.aspx` [cited 10.08.2015].

[whia] Allias whitehatty. cryptdb : Howto compile on ubuntu linux [update 2]. URL: `http://whitehatty.com/2012/09/30/cryptdb-howto-compile-on-ubuntu-linux-12-04/` [cited 02.06.2015].

[Whib] Zack Whittaker. Google admits patriot act requests; handed over european data to u.s. authorities. URL: http://www.zdnet.com/article/google-admits-patriot-act-requests-handed-over-european-data-to-u-s-authorities/ [cited 11.04.2015].

[Whic] Zack Whittaker. Patriot act can 'obtain' data in europe, researchers say. URL: http://www.cnet.com/news/patriot-act-can-obtain-data-in-europe-researchers-say/ [cited 11.04.2015].

[Whi11] Zack Whittaker. Google admits patriot act requests; handed over european data to u.s. authorities, October 2011. URL: http://www.zdnet.com/article/google-admits-patriot-act-requests-handed-over-european-data-to-u-s-authorities/ [cited 26.07.2015].

[Wid] Brandon Widder. Consider this your one-stop shop for your next webmail client. URL: http://www.digitaltrends.com/web/best-web-based-email-clients/ [cited 04.06.2015].

[Wor] Justin Worland. How that massive celebrity hack might have happened. URL: http://time.com/3247717/jennifer-lawrence-hacked-icloud-leaked/ [cited 04.06.2015].

[WY05] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Advances in Cryptology–EUROCRYPT 2005*, pages 19–35. Springer, 2005.

[XYH12] Liangliang Xiao, I-Ling Yen, and DT Huynh. A note for the ideal order-preserving encryption object and generalized order-preserving encryption. *IACR Cryptology ePrint Archive*, 2012:350, 2012.

[Yun13] J.H. Cheon J.-S. Coron M.S. Lee J. Kim T. Lepoint M. Tibouchi A. Yun. Batch fully homomorphic encryption over the integers. EUROCRYPT 2013, 2013. URL: https://www.cryptoexperts.com/tlepoint/pub/slides-CCK+13.pdf.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[zv15] Rijksoverheid zelfrijdende voertuigen. Nederland wordt testland voor zelfrijdende voertuigen, 2015. URL: http://www.rijksoverheid.nl/nieuws/2015/01/23/nederland-wordt-testland-voor-zelfrijdende-voertuigen.html [cited 10.08.2015].