DATA PROTECTION IN TRANSIT AND AT REST WITH LEAKAGE DETECTION

by

Denis Ulybyshev

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

May 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Bharat Bhargava, Chair

Department of Computer Science

Dr. Chris Clifton

Department of Computer Science

Dr. Xiangyu Zhang

Department of Computer Science

Dr. Vaneet Aggarwal

Department of Industrial Engineering

Approved by:

Voicu Popescu

Head of the Graduate Program

To my Mom and Dad

ACKNOWLEDGMENTS

I would like to thank my Academic Adviser, Professor Bharat Bhargava, for his continuous support, encouragement and inspirational guidance. Under his supervision, I learned how to deal with challenging research problems. I learned from Professor Bhargava how to write research proposals for a corporate sector, as well as for NSF and NIH. I would also like to thank my thesis committee members, Professor Chris Clifton, Professor Xiangyu Zhang and Professor Vaneet Aggarwal for their help with the research and valuable feedback. I want to express special thanks to Professor Clifton for helping me to prepare for academic job interviews. I am grateful to my initial Adviser, Professor Suresh Jagannathan, for his support during my two years at Purdue. I am thankful to Professor Jan Vitek for his guidance in 2013-2014. I am thankful to Professor Leszek Lilien for his collaboration, valuable feedback and help with writing papers.

I am thankful to all my current and former lab mates, especially Dr. Rohit Ranchal, Dr. Pelin Angin, Aala Oqab-Alsalem, KMA Solaiman, Miguel Villarreal, Servio Palacios and Ganapathy Mani for their help and collaboration on various projects.

I would also like to thank Jonathan Fulkerson (Coze Health LLC) for being a great mentor during my 2018 internship at Coze Health company and for his help with learning Amazon EC2 cloud infrastructure.

I am grateful to Purdue University for providing me with great opportunities to gain knowledge and for awarding me with Bilsland Dissertation Fellowship in Spring 2019.

I am so thankful to my family for their unconditional love, support, and many sacrifices over the years. I stayed motivated to accomplish this work so that you can be proud of me.

TABLE OF CONTENTS

LIST OF TABLES 8
LIST OF FIGURES 9
LIST OF ABBREVIATIONS 11
ABSTRACT 13
1. INTRODUCTION 14
  1.1 Problem Statement 14
  1.2 Solution Overview 14
  1.3 Research Novelty 16
2. DATA PROTECTION IN TRANSIT AND AT REST WITH ROLE-BASED AND ATTRIBUTE-BASED ACCESS CONTROL 18
  2.1 Extended Attribute-Aware Active Bundle (EA3B) 19
    2.1.1 EA3B concept 19
    2.1.2 Adversary Model 23
    2.1.3 Assumptions 23
    2.1.4 Security Analysis 23
    2.1.5 Semantic Security of EA3B 25
    2.1.6 EA3B-based Hospital Information System 29
    2.1.7 Detecting Security Level of a Web Browser as an Attribute 33
    2.1.8 Data Protection in Vehicle-to-Everything Communication System 33
    2.1.9 Isolated Execution of EA3B 36
  2.2 Related Work 37
  2.3 Evaluation 39
    2.3.1 Experiment 1. Local Performance Evaluation of EA3B 39
    2.3.2 Experiment 2. Performance Evaluation of EA3B, Hosted by Google Cloud 40
    2.3.3 Experiment 3. Evaluation of EA3B Performance Overhead Imposed by Tamper-Resistance Feature and Detection of Web Browser's Cryptographic Capabilities 42
    2.3.4 Experiment 4. Round-trip Time Evaluation for Inter-vehicular Communication 43
    2.3.5 Experiment 5. Round-trip Time Evaluation for Data Requests to Vehicle Records, Stored in Cloud 45

2.3.6 Experiment 6. Evaluation of Inter-vehicular Communication Round-trip Time vs. Baseline Decryption 46
2.3.7 Experiment 7. Performance Overhead Evaluation for EA3B with 4 Policies and for ABs Depending on Number of Included Policies 47
3. DATA LEAKAGE PREVENTION AND DETECTION 49
3.1 Data Leakage Prevention 49
3.2 Data Leakage Detection 57
3.2.1 Digital Watermarks for RGB-images 57
3.2.2 Visual Watermarks 59
3.2.3 Monitoring Network Traffic between Web Services in SOA. 61
3.2.4 Provenance Data for Leakage Investigation 61
3.2.5 Classification Level Elevation for Leaked Data 62
3.2.6 Reissuance vs. Revocation of X.509 certificates 62
3.2.7 “Fake” Leakage 63
3.3 Data Leakage Damage Assessment 64
3.4 Related Work 65
3.5 Evaluation 66
3.5.1 Experiment 1. EA3B’s Performance Overhead Evaluation with Enabled Data Leakage Detection 66
4. SEARCH AND ANALYTICS OVER ENCRYPTED DATA 68
4.1 Search through Encrypted Data 68
4.2 Fast Analytics over Encrypted Data 74
4.3 Related Work 75
4.4 Evaluation 76
4.4.1 Experiment 1. Round-trip Times for Data Requests to VRs using Plaintext and Encrypted Indexing Databases 76
5. BLOCKCHAIN-BASED DATA EXCHANGE 79
5.1 Blockchain for Data Provenance Integrity in SOA 79
5.2 Related Work 83
5.3 Challenges of Blockchain-based Technologies 83
5.3.1 Performance 83

5.3.2 Access Control 83
5.3.3 dDOS Attacks 84
5.4 Evaluation 84
5.4.1 Experiment 1. Data Request Round-trip Time in Blockhub Framework 84
6. FUTURE WORK 87
6.1 Blockchain-based Software Development Framework 87
6.2 WAXEDPRUNE on Serverless Architecture 88
6.3 Policy Aggregation Mechanism for EA3B 89
6.4 Improving Key Derivation Method for EA3B 89
6.5 Scalability Tests for EA3B 89
REFERENCES 90
VITA 97
PUBLICATIONS 106

LIST OF TABLES

Table 2-1. Access Control Policies for EHR in WAXEDPRUNE Project 29
Table 2-2. Vehicle Record Data Subsets. ©2018 IEEE.. 34
Table 2-3. Access Control Policies for Vehicle Record. ©2018 IEEE.. 35
Table 2-4. Related Work for Data Protection in Transit and at Rest 38
Table 3-1. Access Control Policies at Central Monitor in WAXEDPRUNE. ©2018 Springer ... 56
Table 4-1. Indexing Table “IndexTB” for Vehicle Records. ©2018 IEEE 71
Table 4-2. Encryption Schemes and Supported Operations. ©2018 IEEE.................................. 73
Table 4-3. SQL Query Execution Times for Plaintext and Encrypted Databases. ©2018 IEEE . 77

LIST OF FIGURES

Figure 2-1. Extended Attribute-Aware Active Bundle 20

Figure 2-2. Symmetric Key Generation in AB/ EA3B 21

Figure 2-3. Tamper-resistance of AB/ EA3B 22

Figure 2-4. IND-CPA Game 26

Figure 2-5. Cloud-based EHR Management System (use case suggested by Dr. Leon Li, Northrop Grumman) ©2017 IEEE 30

Figure 2-6. Attribute-based Access Control for EHR 31

Figure 2-7. Data Access Workflow in WAXEDPRUNE 32

Figure 2-8. Vehicle-to-Everything Communication Network ©2018 IEEE 35

Figure 2-9. EA3B Isolated Execution in Docker Container 36

Figure 2-10. EHR Round-trip Time (initial data request). ©2018 Springer 40

Figure 2-11. EHR Round-trip Time. ©2018 Springer 40

Figure 2-12. EHR Round-trip Time for Cloud (initial request). ©2018 Springer 41

Figure 2-13. EHR Round-trip Time for Cloud. ©2018 Springer 42

Figure 2-14. AB/ EA3B Performance Overhead. ©2017 IEEE 43

Figure 2-15. Inter-vehicular Communication Round-trip Time (initial request). ©2018 IEEE ... 44

Figure 2-16. Inter-vehicular Communication Round-trip Time. ©2018 IEEE 44

Figure 2-17. Round-trip Time for Vehicle Record, Hosted by Google Cloud (initial request). ©2018 IEEE 45

Figure 2-18. Round-trip Time for Vehicle Record, Hosted by Google Cloud. ©2018 IEEE....... 46

Figure 2-19. Inter-vehicular Communication Round-trip Time. ©2018 ICAEEE 47

Figure 2-20. VR Performance Overhead for Inter-vehicular Communication. Basic AB (VDRB) Performance Evaluation Depending on Number of Policies. ©2018 ICAEEE 48

Figure 3-1. Data Leakage in WAXEDPRUNE. ©2018 Springer 50

Figure 3-2. Data Leakage Detection by Central Monitor. ©2018 Springer 52

Figure 3-3. WAXEDPRUNE Architecture with Leakage Detection Support. ©2018 Springer .. 53

Figure 3-4. Embedding Digital and Visual Watermarks into RGB images, Stored in EA3B 58

Figure 3-5. Digital Watermark Verification Process 58

Figure 3-6. Visual Watermarks in WAXEDPRUNE. ©2018 Springer 60

Figure 3-7. Data Sensitivity Functions. ©2018 Springer 65

Figure 3-8. EA3B Performance with Data Leakage Detection OFF/ON. ©2018 Springer 67

Figure 4-1. Search through EA3Bs Database. ©2018 IEEE 68

Figure 4-2. Microsoft SQL Server-based Database Architecture for AE mode. ©2018 IEEE 71

Figure 4-3. Query Workflow for EA3B Database 72

Figure 4-4. Database Encryption using CryptDB. ©2018 IEEE 72

Figure 4-5. V2X Communication Network with MMCP Key Derivation. ©2018 ICAEEE 75

Figure 4-6. Data Request Round-trip Times for Plaintext and Encrypted Indexing Databases. ©2018 IEEE 78

Figure 5-1. Blockhub: Blockchain-based Framework for SOA. 81

Figure 5-2. Transaction Latencies for Local and Google Cloud-based EA3Bs in Blockhub Framework. ©2018 IEEE 85

Figure 6-1. Software Bundle. ©2018 IEEE 87

Figure 6-2. Blockchain-based Software Development Framework 88

LIST OF ABBREVIATIONS

AB	Active Bundle
ABAC	Attribute-Based Access Control
ABE	Attribute-Based Encryption
AE	Always Encrypted
CA	Certificate Authority
CM	Central Monitor
CRL	Certificate Revocation List
DB	Database
DBMS	Database Management System
DET	Deterministic Encryption
DLP	Data Leakage Prevention
EA3B	Extended Attribute-Aware Active Bundle
EHR	Electronic Health Record
FHE	Fully Homomorphic Encryption
HE	Homomorphic Encryption
ILDP	Information Leak Detection and Prevention
ITS	Intelligent Transportation System
IVC	Inter-Vehicular Communication
KDF	Key Derivation Function
NG	Northrop Grumman
NGCRC	Northrop Grumman Cybersecurity Research Consortium
OPE	Order Preserving Encryption
PHE	Partially Homomorphic Encryption
RBAC	Role-Based Access Control
RTT	Round-trip Time
SB	Software Bundle
SOA	Service-Oriented Architecture

TDE	Transparent Data Encryption
TTP	Trusted Third Party
UDF	User-Defined Function
V2V	Vehicle-to-Vehicle
V2X	V2X Vehicle-to-Everything
VR	Vehicle Record
WAXEDPRUNE	Web-based Access to Encrypted Data Processing in Untrusted Environments

ABSTRACT

Author: Ulybyshev, Denis, A. PhD
Institution: Purdue University
Degree Received: May 2019
Title: Data Protection in Transit and at Rest with Leakage Detection
Major Professor: Bharat Bhargava

In service-oriented architecture, services can communicate and share data among themselves. This thesis presents a solution that allows detecting several types of data leakages made by authorized insiders to unauthorized services. My solution provides role-based and attribute-based access control for data so that each service can access only those data subsets for which the service is authorized, considering a context and service's attributes such as security level of the web browser and trust level of service. My approach provides data protection in transit and at rest for both centralized and peer-to-peer service architectures. The methodology ensures confidentiality and integrity of data, including data stored in untrusted cloud. In addition to protecting data against malicious or curious cloud or database administrators, the capability of running a search through encrypted data, using SQL queries, and building analytics over encrypted data is supported. My solution is implemented in the "WAXEDPRUNE" (Web-based Access to Encrypted Data Processing in Untrusted Environments) project, funded by Northrop Grumman Cybersecurity Research Consortium. WAXEDPRUNE methodology is illustrated in this thesis for two use cases, including a Hospital Information System with secure storage and exchange of Electronic Health Records and a Vehicle-to-Everything communication system with secure exchange of vehicle's and drivers' data, as well as data on road events and road hazards.

To help with investigating data leakage incidents in service-oriented architecture, integrity of provenance data needs to be guaranteed. For that purpose, I integrate WAXEDPRUNE with IBM Hyperledger Fabric blockchain network, so that every data access, transfer or update is recorded in a public blockchain ledger, is non-repudiatable and can be verified at any time in the future. The work on this project, called "Blockhub," is in progress.

1. INTRODUCTION

In service-oriented architecture (SOA), services can communicate and share data among themselves in both centralized and decentralized ways. It is necessary to protect data associated with each service both in transit and at rest.

1.1 Problem Statement

1. Provide data confidentiality and integrity in SOA, including untrusted cloud environments.
2. Support role-based and attribute-based access control.
3. Detect data leakages that can be made by authorized insiders to unauthorized entities.
4. Support search through encrypted data.
5. Collect provenance data for transactions and provide integrity of provenance data.
6. Provide capabilities of audit and control in data communication networks.

1.2 Solution Overview

In this thesis, the methodology that addresses the six aforementioned requirements is presented. In chapter 2, I describe an approach to provide role-based and attribute-based access control for data so that each service can access only those data subsets for which the service is authorized, considering context and service's attributes, such as the security level of the web browser, authentication method, type of the device service is running on and trust level of service. The solution is based on using an Extended Attribute-Aware Active Bundle (EA3B), which is an extension of an Active Bundle (AB). The AB concept was developed by Dr. Bhargava (Purdue University) and by Dr. Lilien (University of Western Michigan), and published in 2006 [18]. Significant contributions for developing the AB concept were made by Dr. Othmane [17] and Dr. Ranchal [1], [4], [33]. One of my main motivations to extend the AB concept was to support more attributes, such as the security level of client's browser, in an attribute-based access control model and to add more options for authentication of services requesting data from an AB. I used the EA3B concept to implement a prototype Hospital Information System. EA3B represents an Electronic Health Record (EHR) of a patient and provides EHR protection in transit and at rest (see chapter 2). I also applied the EA3B concept to secure an exchange of video data, captured by vehicle’s dash camera [31]. EA3B carries captured video data used in Vehicle-to-Everything (V2X) communication system and provides video data protection in transit and at rest. A face detection algorithm is applied to every frame of captured video and the result of face detection is used in access control policies, embedded in EA3B.

My other main motivation to extend the AB methodology was the necessity to detect data leakages that can be made by authorized insiders to unauthorized parties. In chapter 3, I present my solution for Information Leak Detection and Prevention (ILDP) for several types of data leakage scenarios. Leakage detection is based on embedding digital watermarks in an EA3B and in data, which are stored in an EA3B in encrypted form. Visual watermarks are embedded in the data as well. A web crawler verifies watermarks and checks whether discovered data is supposed to be where the data were found by the web crawler. Furthermore, to ensure data leakage prevention, a Central Monitor is employed to validate data transactions and to approve data extraction and decryption from an EA3B. I implemented the solution as a “WAXEDPRUNE” (Web-based Access to Encrypted Data Processing in Untrusted Environments) project [5], funded by Northrop Grumman Cybersecurity Research Consortium (NGCRC). As a use case, I applied my solution to build a Hospital Information System, which protects Electronic Health Records (EHRs) in transit and at rest.

My other accomplishment is developing a mechanism for search through a database of EA3Bs, i.e., search through encrypted data. EA3B stores data in encrypted form, and it is highly desirable to support capabilities of search and analytics over encrypted data. My solution for that is presented in chapter 4. The solution supports SQL queries over encrypted data, search through a database of EA3Bs and fast analytics over EA3Bs. As a use case, I applied my solution to build a Vehicle-to-Everything (V2X) communication system, which provides protection of Vehicle Records (VRs) in transit and at rest. It also enables capabilities of search through encrypted VRs, supporting SQL queries over encrypted data. In 2018, I presented a paper on my solution in IEEE International Conference on Internet-of-Things (IEEE ICIOT) [6]. It illustrates the application of my concept for Road Assistance and Intelligent Transportation Systems (ITS). Furthermore, I added the functionality of building fast analytics over EA3Bs and applied the approach to build fast analytics over encrypted VRs. In 2018, my solution was presented in 4-th International Conference on Advanced Computer, Electric and Electronic Engineering [24].

To help with investigating data leakages, provenance data can be used and their integrity needs to be guaranteed. Blockchain-based technology fits well to provide integrity of provenance data in SOA with multiple untrusted writers. That is why, I decided to integrate WAXEDPRUNE framework, which supports fine-grained role-based and attribute-based access control and provides data protection in transit and at rest, with IBM Hyperledger Fabric blockchain network. I selected IBM Hyperledger Fabric blockchain network for two reasons: (1) it is open-source; (2) it is a mature enterprise-level solution. The idea is that every data access, transfer or update in WAXEDPRUNE is register in blockchain public ledger through IBM Hyperledger Fabric network. It guarantees that every data transaction is non-repudiatable and can be verified at any time in the future. Chapter 5 presents the details of the solution. The great feature of AB and its successor, EA3B, is that it can carry data, as well as binary large objects, software source code and binary executables, in encrypted form, providing their confidentiality and integrity. It inspired the idea of using EA3B for software exchange in a collaborative software development environment, called "Blockhub". This work is in progress, and the "Blockhub" idea was presented and published in IEEE CLOUD conference in 2018 [30]. As a future work, I plan to integrate Blockhub with existing software repository system, such as GitHub or a corporate internal software development system, which involves multiple collaborators from different domains. The core idea is that software module update, before being checked-in to GitHub, is recorded in a blockchain public ledger. It will allow to audit and control software updates. Furthermore, it will protect against malicious repository administrators by preventing audit logs from corruption. Future work is discussed in Chapter 6.

1.3 Research Novelty

Novelty of my work is the comprehensive solution that provides the following:

1. Data protection in transit and at rest for SOA, supporting peer-to-peer and centralized service architectures, including untrusted cloud environments.
2. Fine-grained role-based and attribute-based access control. Service attributes considered by data access model in SOA include security level of service's web browser and type of the device service is running on (mobile vs. desktop).
3. Detection of several types of data leakages that can be made by authorized insiders to unauthorized entities.

4. Search and analytics over encrypted data.
5. Provenance data recording for data transactions. Integrity of provenance data is guaranteed by a blockchain-based technology.
6. Capabilities of audit and control in data communication network in SOA.

2. DATA PROTECTION IN TRANSIT AND AT REST WITH ROLE-BASED AND ATTRIBUTE-BASED ACCESS CONTROL

Portions of this chapter have been presented and published at 4-th International Conference on Advanced in Computer, Electrical & Electronic Engineering Conference, ICACEEE 2018 Ulybyshev, Denis, Servio Palacios, Ganapathy Mani, Aala Oqab Alsalem, Bharat Bhargava, and Puneet Goyal. "On-the-fly Analytics over Encrypted Records in Untrusted V2X Environments," ICACEEE, 2018, Zurich.

2.1 Extended Attribute-Aware Active Bundle (EA3B)

2.1.1 EA3B concept

My solution for protecting data in transit and at rest in SOA with providing leakage detection, as well as role-based and attribute-based access control, relies on an Extended Attribute-Aware Active Bundle (EA3B), which is built on top of an Active Bundle (AB) [1]. The AB is a self-protecting structure that incorporates sensitive data in encrypted form, access control policies and policy enforcement engine (Virtual Machine). The AB concept was designed and developed by Prof. Bharat Bhargava (Purdue University), and by Prof. Leszek Lilien (University of Western Michigan) [18]. Dr. Lotfi ben Othmane [17] and Dr. Rohit Ranchal [1], [4], [33] made significant contributions to further develop and improve the AB concept. I selected the AB-based concept since it provides data protection in transit and at rest and has the following advantages:

1. It does not rely on Trusted Third Party (TTP) [32] to issue keys for the recipient services.
2. Data owner's availability is not required to enforce access control policies.
3. Symmetric encryption / decryption keys are generated on-the-fly, based on the AB execution flow [1]. Keys are not stored neither inside AB nor on TTP.
4. Role-based and attribute-based access control are supported. Attributes include trust level of services, which is constantly recalculated, and context, e.g., normal vs. emergency.
5. AB is tamper-resistant: integrity of data, metadata and access control policies is provided.
6. AB is agnostic to policy specification language and policy evaluation (enforcement) engine.
7. Complex access control policies can be written in Java language (language of AB implementation).
8. Services can update data on-the-fly and store updated data in encrypted form.
9. Solution supports X.509 certificates for authentication and RESTful API: it is compatible with industry-standard SOA/cloud frameworks.
10. AB works in both centralized and decentralized peer-to-peer service architectures

My contributions to develop EA3B on top of AB concept are as follows:

1. Data Leakages Detection (DLD) mechanism for leakages, made by authorized insiders [7]
2. Mechanism for indexing, search and analytics over the database of EA3Bs [6]
3. Extended support of attribute-based access control [3], [5], [8]
    3.1. Security level of client's browser [12], [13]

3.2. Type of client's device: desktop vs. mobile [5]

3.3. Authentication method (password-based vs. hardware-based vs. fingerprint) [5]

4. Password-based authentication, in addition to X.509 certificate-based authentication [5]
5. Provenance data captured for EA3B transfers, data accesses [7]

I implemented the aforementioned features in the "WAXEDPRUNE" project, funded by Northrop Grumman Cybersecurity Research Consortium.

Data are a non-relational database stored in an EA3B in the form of key-value pairs with encrypted values. Separate symmetric 128-bit AES key is used per separate data subset, such as medical information or contact information of a patient. Here is the example of key-value pair stored in the EA3B:

*{ "ab.patientID" : "***Enc***(0123456)" }*

Patient ID is 0123456 and it is stored in EA3B in encrypted form.

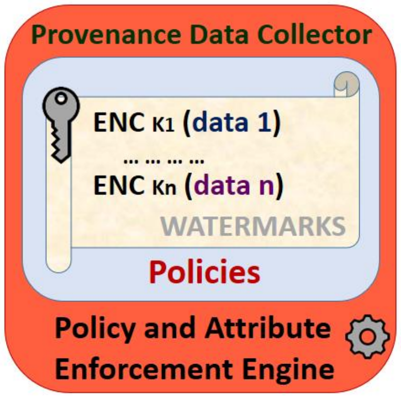

Provenance Data Collector
ENC K1 (data 1)
… … … …
ENC Kn (data n)
WATERMARKS
Policies
Policy and Attribute Enforcement Engine

Figure 2-1. Extended Attribute-Aware Active Bundle

Each data item is encrypted with a separate symmetric AES key, which is generated on-the-fly based on the execution flow. An EA3B uses the same key generation scheme as an AB. Symmetric key generation is based on the unique information generated in the execution control flow path of an EA3B [1]. This information depends on the EA3B modules and their following resources:

1) the result of authentication phase, i.e., whether authentication passed or failed;
2) the result of authorization phases. It depends on whether the access to data subset, which is being encrypted, is granted or denied by EA3B kernel, based on subject's role (e.g. doctor, insurance agent, researcher), extracted from the X.509 certificate of the subject (service);
3) the authentication code;

4) the authorization code; which does policy evaluation;
5) the data subset to be encrypted;
6) the applicable access control policies, which are stored in an encrypted form.

To ensure proper entropy in the key, hash of the aforementioned resources is transformed into a secret. This secret is then used to derive the symmetric key, using *SecretKeyFactory*, *PBEKeySpec* and *SecretKeySpec* methods from javax.crypto library. During EA3B creation, the data owner's policies are first embedded into the EA3B template, which is then executed to obtain the information and to derive the corresponding symmetric keys for each separate data subset [1]. Decryption key derivation procedure is similar to encryption key derivation. EA3B execution control steps generate the information, based on six aforementioned resources above. Hash of this information is then used to derive the symmetric key, employing *SecretKeyFactory*, *PBEKeySpec* and *SecretKeySpec* methods. This symmetric key is able to decrypt the corresponding data subset.

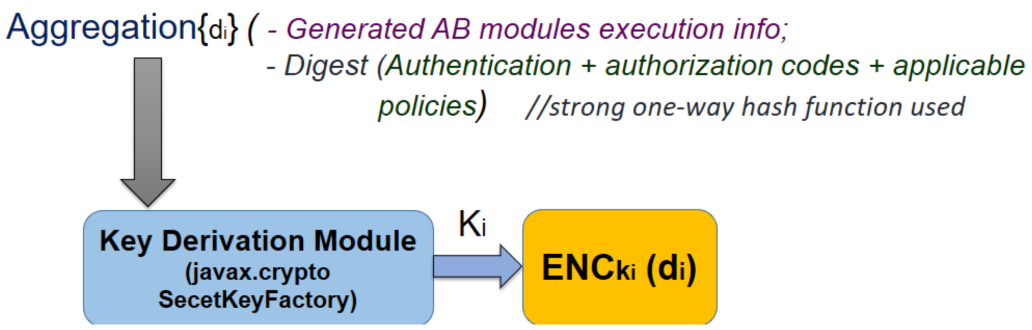

Aggregation{d_i} (- Generated AB modules execution info;
- Digest (Authentication + authorization codes + applicable policies) //strong one-way hash function used
Key Derivation Module (javax.crypto SecetKeyFactory)
K_i
ENC_{ki} (d_i)

Figure 2-2. Symmetric Key Generation in AB/ EA3B

When service requests data from an EA3B, as a first interaction step the identity of the requesting service requesting is verified. An EA3B supports password-based and signed digital certificate-based authentication schemes. The former redirects the service to the Authentication Server (AS), where credentials of the service are validated and the authentication ticket is generated. Authentication Ticket is encrypted and contains information on service's role and set of attributes, such as security level of the used web browser, authentication type and type of the device service is running on. Encrypted Authentication Ticket is forwarded to EA3B and is handled by EA3B's attribute enforcement engine. Details are illustrated in Fig. 2-7. In certificate-based authentication scheme, service presents its X.509 certificate signed by a trusted Certificate Authority (CA) to EA3B to verify its authenticity. After service authenticates itself, its attributes (trust level, cryptographic capabilities of a browser, authentication method, type of the device, etc.) and the context (e.g., normal vs. emergency) are evaluated and enforced by the policy enforcement engine embedded into EA3B. Then evaluation of applicable access control policies determines what data can be disclosed to the requesting service. Symmetric decryption keys will be generated to decrypt those data items for which the authenticated service is authorized, based on access control policies, stored in EA3B. Service requests keys from key-value pairs and corresponding values are decrypted, based on derived decryption keys. Decrypted values will be sent to the service using https protocol, provided service's trust level is sufficient, its browser's cryptographic capability level is sufficient and client's authentication method is secure.

Another very important feature an EA3B inherited from an AB is tamper-resistance [1]. Modification of any items from the list below:

1. Authentication code (when an attacker tries to bypass the authentication phase);
2. Service's X.509 certificate (when an attacker tries to impersonate his/her identity or use wrong certificate);
3. Authorization code (when an attacker tries to bypass evaluation of access control policies);
4. Applicable access control policies (when an attacker tries to modify policies to gain access to unauthorized data);

will result in EA3B's digest change and lead to the incorrect decryption key derivation. Tamper-resistance mechanism provides integrity of data and embedded access control policies. Storing data and access control policies in encrypted form provides data confidentiality.

EA3B, same as AB, is written in Java language and implemented as a Java Executable Archive (JAR). Access control policies are specified using Javascript Object Notation (JSON) [10]. WSO2 Balana [15] is used for policy evaluation.

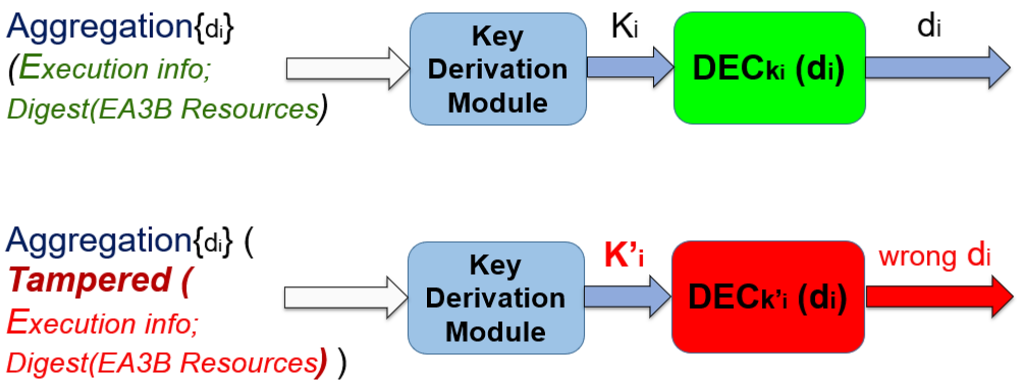


Figure 2-3. Tamper-resistance of AB/ EA3B

2.1.2 Adversary Model

An EA3B can be hosted and executed by the recipient or server/ cloud administrator. The adversary model in this work protects from:

1. Malicious actor who hosts and executes EA3B (client or server/ cloud administrator) and tries to do the following:
   a) Modify EA3B code to bypass authentication phase and/or access control policy check
   b) Add new policies allowing data access
   c) Gain unauthorized access to encrypted data, stored in EA3B by reverse-engineering EA3B from jar-file
   d) Misrepresent identity to gain unauthorized access to data
2. Authorized insider who leaks data to unauthorized parties

2.1.3 Assumptions

1. Hardware, OS and Java Virtual Machine (JVM) are trusted on a side hosting (executing) an EA3B
   - Intel SGX or ARM TrustZone platforms can be used to relax these assumptions
2. Https protocol is used for:
   - EA3B transfer between web services
   - Communications between an EA3B and web services
3. Unauthorized party attempts to use leaked data and/ or leaked data is available to investigator/ web crawler

2.1.4 Security Analysis

Let us discuss how an EA3B-based approach provides protection against a malicious actor who hosts and executes EA3B (client or server/ cloud administrator) and tries to gain unauthorized access to data, stored in EA3B in encrypted form, using the following:

1a) Modify EA3B code to bypass authentication phase and/or access control policy check;

1b) Add new policies allowing data access.

Solution: symmetric key derivation process, discussed in Section 2.1.1 and illustrated on Fig. 2-2, will not allow an adversary to decrypt data for which the adversary is not authorized. Once EA3B source code modules, which are responsible for service authentication and for evaluation of access control policies evaluation, are modified by an attacker, the input for key derivation module will also change and become different from the original input, used to derive encryption key during EA3B generation process. Modified input will lead to deriving a decryption key, different from the original one, so that the derived key will not be able to decrypt the requested data subset.

1c) Gain unauthorized access to encrypted data, stored in EA3B by reverse-engineering EA3B from jar-file

Solution: it might be possible for an attacker to reverse-engineer the original source code of EA3B, but the data and access control policies will still be in encrypted form. Confidentiality of data and access control policies relies on AES encryption scheme with 128-bit symmetric key. It is computationally infeasible to brute force 2^128 keys. Based on reverse-engineered code, attacker might be able to re-create a new EA3B with modified access control policies, but this EA3B will not be able to pass 2-way authentication process with services, requesting data, since attacker does not have a required X.509 certificate to present his benign identity to the requesting service. I assume that CA will not issue a certificate corresponding to the benign role, such as “Hospital Administrator”, to an attacker. Semantic security aspect of AES encryption scheme is discussed in the Section 2.1.5.

1d) Misrepresent identity to gain unauthorized access to data

Solution: identity management in WAXEDPRUNE is based on two mechanisms:

(1) X.509 certificates, issued by a trusted CA;

(2) password-based authentication.

For (1), misrepresenting X.509 certificate is infeasible since attacker does not have the required private key. I assume that CA will not issue an X.509 certificate, corresponding to a benign role, for an attacker.

For (2), it is necessary for clients to keep their credentials secure, select strong passwords and periodically change them.

2. Authorized insider who leaks data to unauthorized parties

Solution: will be covered in chapter 3

2.1.5 Semantic Security of EA3B

“A semantically secure cryptosystem is one where only negligible information about the plaintext can be feasibly extracted from the ciphertext“ [65]. Goldwasser and Micali introduced a probabilistic model for data encryption, according to which “extracting any information about the cleartext from the cyphertext is hard on the average for an adversary with polynomially bounded computational resources.” [66]. It means that for any probabilistic polynomial time algorithm it is computationally infeasible for an adversary A to tell which of plaintexts M0 and M1 is encrypted in a ciphertext C. In other words, an adversary cannot distinguish between the ciphertexts of two plaintexts M0 and M1.

As described in Section 2.1.1, the decryption keys for the data subsets stored in EA3B (one separate symmetric key per data subset) are generated on-the-fly, based on the execution flow. EA3B does not reveal any ciphertext for the declined data request. By sending unauthorized data requests to EA3B, an attacker will not be able to see ciphertexts of data he or she asked for, since data decryption is made by EA3B kernel on-the-fly only if authentication phase passes. I assume an attacker is not able to authenticate itself to EA3B since an attacker does not have an X.509 certificate corresponding to the benign role which would allow to pass authentication with EA3B. Thus, an attacker cannot decrypt any ciphertext, as well as a target ciphertext stored in EA3B, using EA3B encryption scheme. As discussed in Section 2.1.4 (1c), an attacker might be able to reverse-engineer EA3B from its source jar-file and gain an access to encrypted data sets. For an attacker that has AES-encrypted data from a reverse-engineered EA3B, it is computationally infeasible to break AES encryption scheme with 128 bit key, as it will be explained below in this section. But an attacker might try a Chosen Plaintext Attack (CPA) [69] by generating his or her own EA3B, then again reverse-engineering it and get the ciphertexts for any plaintexts of attacker’s choice. An attacker might recover the key derivation scheme but still it will not help him/ her to recover plaintexts from the original EA3B with sensitive data since an attacker does not have an X.509 certificate for a benign role, which is needed to pass an authentication with EA3B, which, in turn, is needed to derive the right symmetric key.

It is shown in [67] that notions of Semantic Security and Indistinguishability Security are equivalent. Indistinguishability under chosen plaintext attack (IND-CPA) is illustrated on Fig. 2-4 and uses the following IND-CPA experiment (game):

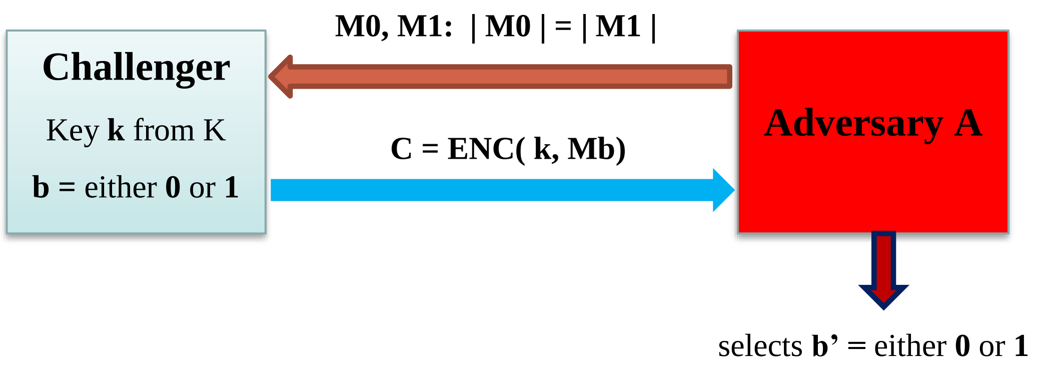


Figure 2-4. IND-CPA Game

1. A key k is generated on a Challenger's side
2. An Adversary is given an oracle access to ENC(k, M), i.e., an adversary does not know the key k, but can send the plaintext messages M to a Challenger and get them back in an encrypted form
3. An Adversary sends a pair of equal-length messages M0, M1 to a Challenger
4. A random bit b from {0,1} is selected by a Challenger
5. A Challenger sends the challenge ciphertext ENC(k, Mb) to an Adversary. ENC(k, Mb) denotes the ciphertext, which is the result of the plaintext Mb encryption with the key k
6. An Adversary chooses as many plaintexts as he/ she wants, and receives the corresponding ciphertexts via oracle access ENC(k, M)
7. An Adversary outputs b': either 0 or 1.
8. An Adversary wins if b'=b.

Here is the formula for Advantage of an Adversary A in encryption scheme E:

Adv (A, E) = Pr[Adversary wins] – ½

Encryption scheme is IND-CPA secure if for an adversary A if:

Adv (A, E) is "negligible", or Pr[b=b'] - ½ is "negligible"

It means that in IND-CPA secure encryption scheme, the probability to win the IND-CPA game for an Adversary can only be negligibly greater than ½ . Having limited computational resources, an Adversary can not distinguish whether M0 or M1 is encrypted in a challenge ciphertext ENC(k, Mb).

There are two approaches to define the negligibility:

1. Concrete approach

   "A scheme is (t,ε)-secure if every adversary running for time at most t succeeds in breaking the scheme with probability at most ε" [72]

   Example: AES with 128-bit key may be expected to be (t, t/2^n)-secure.

   N=128, t=2^58, then ε= 2^(-70). (2^58 is the approximate number of seconds since the big bang). Probability ε= 2^(-70) for an adversary to succeed in breaking the encryption scheme is negligible

2. Asymptotic approach

   "For every constant C the Adversary's success probability is smaller than n^(-c) for large enough values of n" [72].

EA3B uses AES symmetric encryption algorithm. Any deterministic encryption scheme is not IND-CPA secure since the same plaintext is always encrypted into the same ciphertext. AES in Electronic Codebook (ECB) mode is not IND-CPA secure since two identical plaintext blocks are always encrypted into two identical plaintext blocks. An attacker can distinguish what ciphertexts correspond to M0 and M1 with probability 1. Then the probability for an Adversary to win the IND-CPA game is:

$$\Pr[\, b=b' \,] - \tfrac{1}{2} = \tfrac{1}{2} \,,$$

and it is non-negligible. An Adversary's advantage is non-negligible, too:

$$\text{Adv}\,(A, E) = \Pr[\text{Adversary wins}] - \tfrac{1}{2} = 1 - \tfrac{1}{2} = \tfrac{1}{2}$$

An adversary should not have a non-negligible advantage to satisfy IND-CPA security property.

Counter (CTR) and Cipher Block Chaining (CBC) modes for block ciphers provide semantic security under a CPA attack "assuming a random IV" [75], where IV is an Initialization Vector. Thus, AES encryption scheme in CBC and CTR modes provides IND-CPA security if an IV is random. A random IV is used as a random input in CBC mode to encrypt the first plaintext block. For the consecutive plaintext blocks, exclusive OR (XOR) is applied to XOR a plaintext block with the result of the previous block's encryption. "Each ciphertext block depends on all plaintext blocks processed up to that point" [68]. The last cipher block depends on the entire plaintext. CBC "is secure as a probabilistic encryption scheme, achieving indistinguishability from random bits, assuming a random IV" [75]. A plaintext message is padded to a multiple of 128 bit, which is the AES block size. Thus, for the lengths of the plaintext less than 128 bits, the length of the ciphertext does not reveal any information about the length of the plaintext. However, for messages greater than 128 bits, the length of a ciphertext reveals information about the length of a plaintext with a multiple of 128 bit. For instance, in a Hospital Information System, discussed in section 2.1.6 and illustrated on Fig. 2-5, network traffic for the response on doctor's request for one EHR will be less than for the response on doctor's request for several EHRs, belonging to different patients. If an attacker intercepts encrypted network traffic between doctor's service and Hospital Server (see Fig. 2-5), then an attacker would be able to differentiate the responses for requests of one EHR vs. multiple EHRs.

To mitigate this side channel attack, data fields stored in EA3B can all have the same length. For multiple-field data requests, an *oblivious transfer* protocol can be used, as a future work. "An oblivious transfer (OT) protocol is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred" [70]. Shimon Even, Oded Goldreich, and Abraham Lempel introduced "1 out of 2 oblivious transfer" [71]. "The l-out-of-2 oblivious transfer allows one party to transfer exactly one secret, out of two recognizable secrets, to his counterpart. The first (second) secret is received with probability one half while the sender is ignorant of which secret has been received" [71]. A technique to prevent a plaintext's property leakage from the ciphertext length, includes two parts:

(1) Adding randomness (noise) to the data, sent by EA3B's host to the client in encrypted form.
    M dummy blocks (128 bits each) are added to N blocks (128 bits each) of real data. A "man-in-the-middle" attacker cannot determine N blocks of real data among (N+M) blocks

(2) Splitting a transferred message into chunks
    Encrypted message Enc(M) is sent by EA3B's host to the client not all at once, but in multiple chunks with a time delay t, so that a "man-in-the-middle" attacker who intercepts the network traffic, sent to the client, is not able to differentiate whether the

chunk belongs to message Enc(M) or to some other encrypted message Enc(M2), which was sent to the client after Enc(M).

2.1.6 EA3B-based Hospital Information System

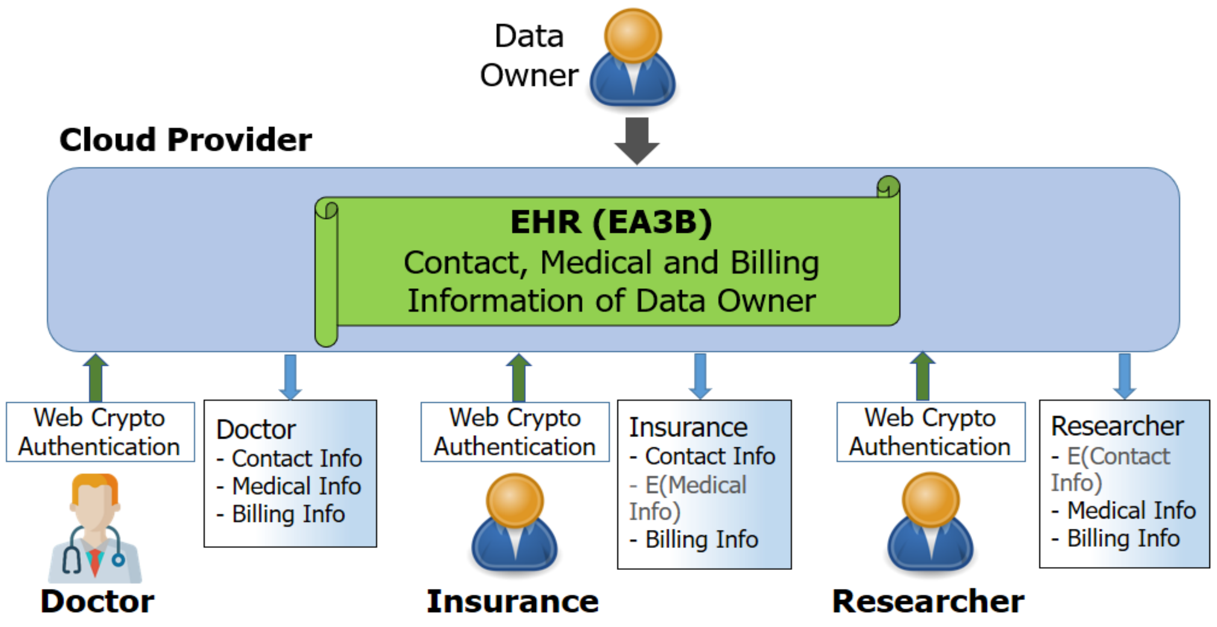
As a use case illustrating rich functionality of EA3Bs, I implemented a Hospital Information System. Based on NGCRC's vision, it was called "WAXEDPRUNE". It provides protection of Electronic Health Records (EHRs), stored in the form of EA3Bs in cloud, one EA3B per one EHR. Cloud can be untrusted and EA3Bs guarantee EHR protection from curious or malicious cloud administrators. In implemented WAXEDPRUNE prototype there are three client services: doctor, insurance and researcher. Access control policies specify that doctor can access medical data (test results, diagnosis, prescriptions, etc.) of a patient, as well as contact and billing information. Table 2-1 shows access control policies for the Medical, Billing and Contact Information of a Patient, correspondingly. Insurance can access contact and billing information only, access to medical information is not allowed. Researcher can only access anonymized records of patients, i.e., medical and billing information.

Three services (Doctor, Researcher, Insurance), as well as a Hospital service, which serves data requests, are running as NodeJS servers (daemons) on a cloud provider (see Fig. 2-5) and are listening to the corresponding opened ports [5]. The services communicate with an EA3B, which represents the given patient, on a server (or cloud) side. WAXEDPRUNE architecture is illustrated on Fig. 2-7.

Table 2-1. Access Control Policies for EHR in WAXEDPRUNE Project

ALLOW			
Resource	Medical Info	Contact Info	Billing Info
Subject's Role	Doctor, Researcher	Doctor, Insurance	Doctor, Researcher, Insurance
Action	Read	Read	Read

Initial data request from unauthenticated client is redirected from a Cloud Provider to an Authentication Server (AS) where client needs to authenticate itself in order to receive a valid authentication ticket. Along with the authentication procedure, the level of cryptographic capabilities of client's browser and client's authentication method are determined and are included to the authentication ticket, which is signed by AS. If the client authentication procedure is successful then AS redirects client's data item request to the proper service, corresponding to client's role, with the new valid authentication ticket. Once a corresponding service running in cloud receives data request and the authentication ticket from the client, signature, client ID and ticket expiration time, extracted from the ticket, as well as access control policies are evaluated, based on client's role [5]. Based on evaluation of access control policies, the client's browser cryptographic capabilities and client's authentication method, the EA3B responds to the requesting service with the authorized data. Then service transfers authorized data to the client. The workflow is illustrated on Fig. 2-7.



**Doctor's icon made by Freepik from www.flaticon.com https://www.flaticon.com/free-icon/doctor_1021606*

Figure 2-5. Cloud-based EHR Management System (use case suggested by Dr. Leon Li, Northrop Grumman) ©2017 IEEE

For a doctor, who logs in and requests for data from an insecure browser with WebCrypto not enabled (see Fig.2-6), the set of retrieved data is smaller than for the same doctor who logs in

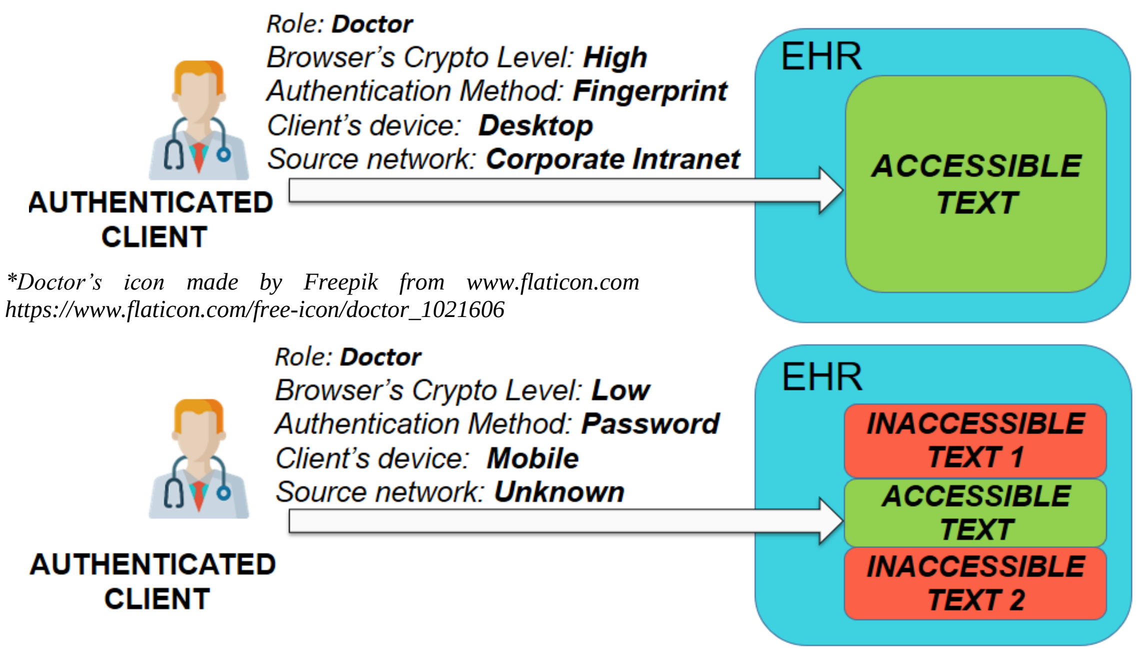

Role: Doctor
Browser's Crypto Level: High
Authentication Method: Fingerprint
Client's device: Desktop
Source network: Corporate Intranet
AUTHENTICATED CLIENT
EHR
ACCESSIBLE TEXT
*Doctor's icon made by Freepik from www.flaticon.com https://www.flaticon.com/free-icon/doctor_1021606
Role: Doctor
Browser's Crypto Level: Low
Authentication Method: Password
Client's device: Mobile
Source network: Unknown
AUTHENTICATED CLIENT
EHR
INACCESSIBLE TEXT 1
ACCESSIBLE TEXT
INACCESSIBLE TEXT 2

Figure 2-6. Attribute-based Access Control for EHR

from a secure browser with WebCrypto enabled [37]. Cryptographic capabilities, i.e., "WebCrypto enabled", assume existence and support of certain cryptographic libraries in the client's browser [5]. The threshold of the sufficient amount of cryptographic libraries supported by the browser, can be tuned, depending on the context. Demo video for Hospital Information System prototype is available [8].

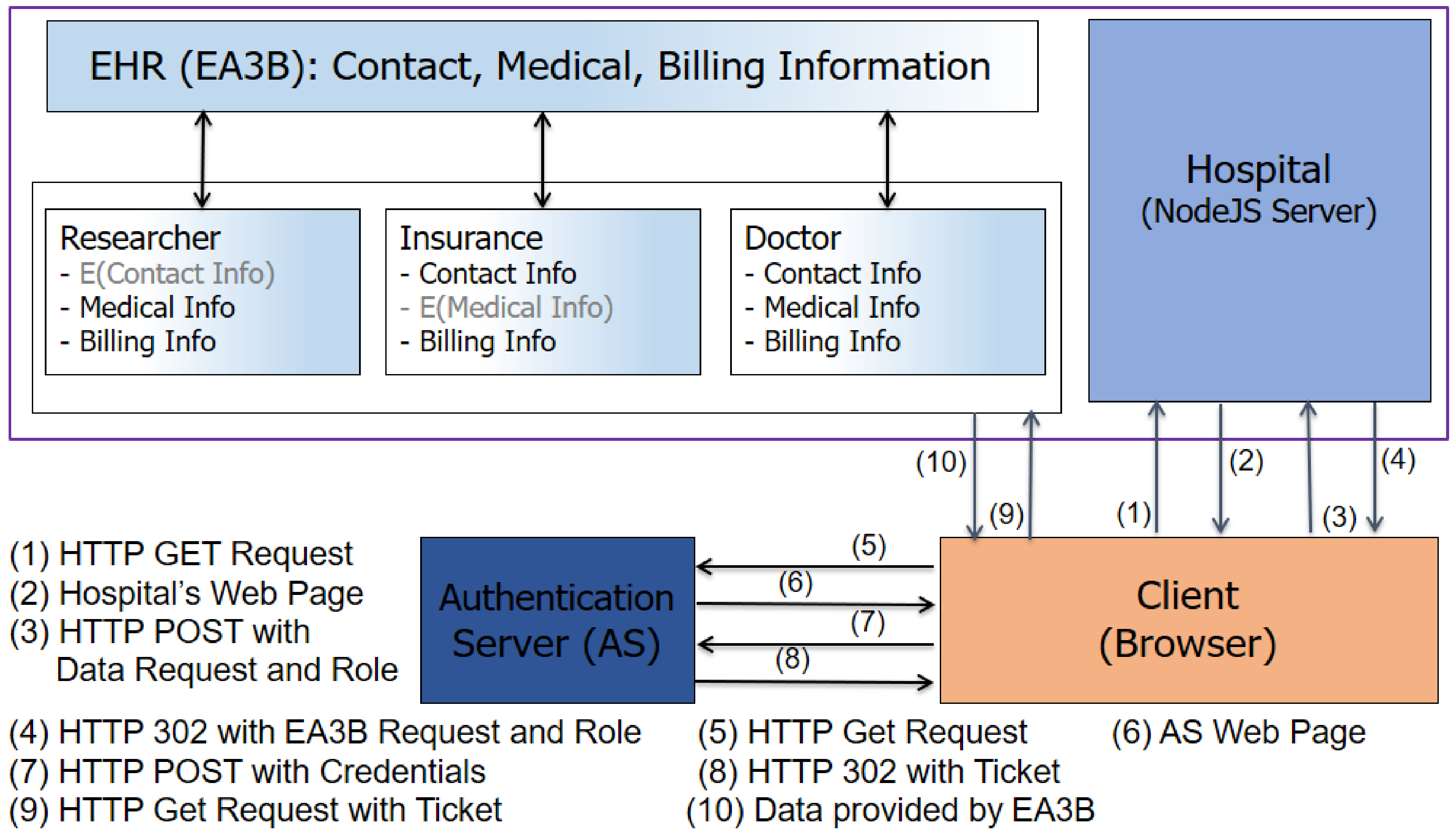


Figure 2-7. Data Access Workflow in WAXEDPRUNE

2.1.7 Detecting Security Level of a Web Browser as an Attribute

EA3B supports an extended attribute-based access control. Set of supported attributes includes cryptographic capabilities of client’s web browser, i.e., how secure the browser is. If a client requests data from an EA3B using an old browser with low level of cryptographic capabilities, then either limited or no data can be retrieved from EA3B even if the client’s role allows data access and client’s trust level is sufficient. “The W3C Web Cryptography API [12] provides generic cross-browser access to cryptographic primitives such as AES and ECDSA in browsers” [5]. Once a client opens the web browser, the cryptographic capabilities of this browser are evaluated. The existence of cryptographic libraries is verified and then the corresponding trust level is assigned to the browser used by a client, which sent data request to EA3B. This trust level of the browser is included in the authentication ticket, which is generated by an Authentication Server in case of password-based service authentication is used. Then the authentication ticket is encrypted and forwarded to a site, which hosts EA3B. It could be other service, or a server or a cloud provider. EA3B kernel at a host site decrypts an authentication ticket and extracts the level of client’s browser cryptographic capabilities. Data owner can adjust the thresholds in EA3B’s configuration metadata for assigning data authorization levels to corresponding levels of browser’s cryptographic capabilities. Detection of browser’s cryptographic capabilities is currently supported for password-based authentication only, when the service requesting for data from EA3B is redirected to an Authentication Server to enter credentials. This work was done to implement “WAXEDPRUNE” project in 2016 [5], in collaboration with Harry Halpin from MIT.

“Hardware token-based authentication can be enabled as soon as the W3C Web Authentication API [13] is deployed in modern browsers” [5].

2.1.8 Data Protection in Vehicle-to-Everything Communication System

Vehicles and road infrastructure objects can exchange different types of useful information among themselves in Vehicle-to-Everything (V2X) communication systems, aiming to help preventing traffic accidents, improve road safety and assist drivers. This information may include vehicle’s and driver’s data, as well as data on road events such as traffic jams and road hazards. The research report "Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application" [34] made by National Highway Traffic Safety Administration, assesses the readiness of Vehicle-to-Vehicle (V2V) communication systems in terms of protecting data privacy. Based on European

"ETSI" standards, "anonymity, pseudonymity, unlinkability and unobservability" are all required to establish privacy [35]. EA3B provides data protection in transit and at rest for V2X communication systems. It can carry different types of data, including vehicle's health reports and video, captured by a dash camera [31].

Vehicles and road infrastructure objects exchange data among themselves by means of Vehicle Records (VRs) [6]. VR contains owner and vehicle data, as well as information on captured road events, in encrypted form and is implemented as EA3B, built on top of AB [1]. Table 2-2 shows the structure of VR. Examples of access control policies for VR are given in Table 2-3.

Table 2-2. Vehicle Record Data Subsets. ©2018 IEEE

VR			
ID	**Owner's Info**	**Vehicle's Info**	**Road Events**
	• Name • Address • Phone • Driver's license number	• VIN • License plate • Health Check ➢Engine temperature ➢Fluids Level ➢Tires pressure	• Traffic jam • Accident • Road work • Obstacle

Fig. 2-8 illustrates the workflow in a given V2X communication network. Once a vehicle reaches the proximity of the base station, an EA3B is created on a vehicle's side with data subsets, specified in Table 2-2. Then the created EA3B is sent to a cloud provider via a base station at step 2. EA3B is created on a vehicle as opposed to cloud provider in order to protect VR data from curious and/or malicious cloud database administrators. At step 1, in the proximity of a base station, encrypted ID, speed and model are sent every 5 seconds to a cloud provider as well. They will be stored in indexing database in cloud to provide search capabilities through encrypted VRs. This will be explained in details in chapter 4. Inter-vehicular data exchange might take place at step 3. A vehicle can send data request to an EA3B hosted by another vehicle. EA3B's kernel will evaluate the request based on the role and attributes of a requesting service [6]. At step 4, communication between vehicle and VR, stored in cloud, may take place. For instance, Law Enforcement entity might query the database of VRs, stored in cloud, in order to identify vehicles, which exceeded the speed limit. Or, regular vehicle might query the database of VRs, stored in cloud, in order to find out whether there are traffic jams, which is determined based on recorded speeds of vehicles, who recently passed the same base station.

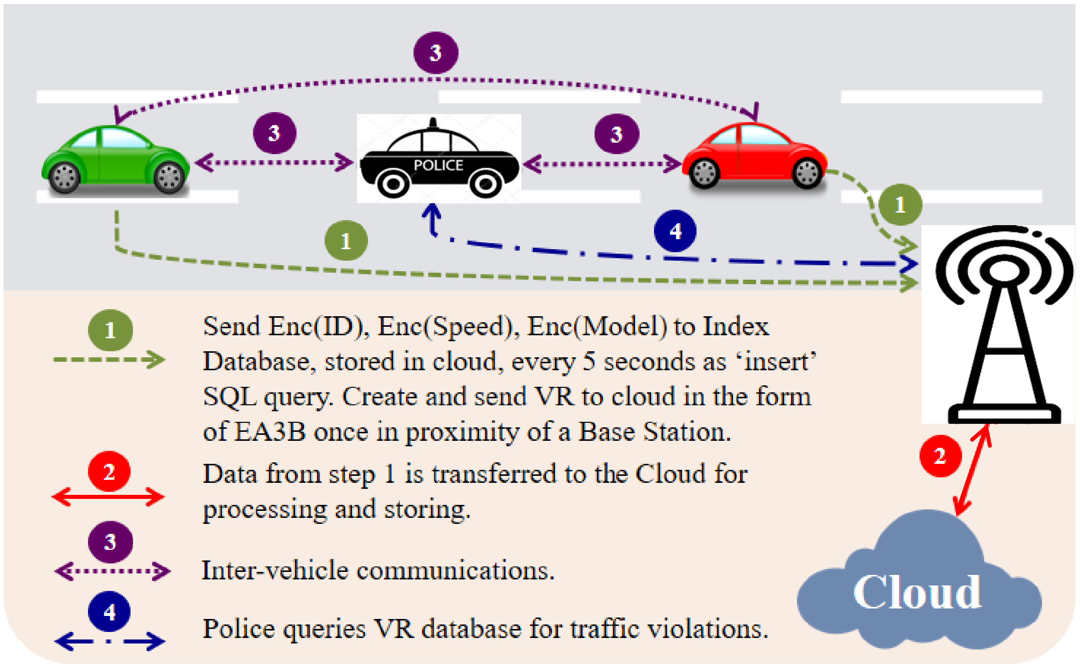

3
3
POLICE
3
1
4
1
1
Send Enc(ID), Enc(Speed), Enc(Model) to Index Database, stored in cloud, every 5 seconds as 'insert' SQL query. Create and send VR to cloud in the form of EA3B once in proximity of a Base Station.
2
Data from step 1 is transferred to the Cloud for processing and storing.
3
Inter-vehicle communications.
4
Police queries VR database for traffic violations.
2
Cloud

Figure 2-8. Vehicle-to-Everything Communication Network ©2018 IEEE

*Clipart taken from http://www.clker.com/cliparts/a/0/d/e/11971194531912449946Chrisdesign_Beetle_(car).svg.hi.png; http://www.clker.com/cliparts/R/7/e/b/5/X/red-car-hi.png; https://cdn1.iconfinder.com/data/icons/vehicles-roads-road-symbols/32/street-medium-512.png; http://www.clker.com/cliparts/s/E/z/g/z/V/cloud-hi.png; http://www.clker.com/clipart-777462.html . Antenna (base station) icon made by Freepik from www.flaticon.com https://www.flaticon.com/free-icon/antenna_145093

Table 2-3. Access Control Policies for Vehicle Record. ©2018 IEEE

ALLOW				
Resource	Driver's License Number	VIN	Owner's Address	Traffic Events
Subject's Role	Law Enforcement	Law Enforcement, Car Repair	Law Enforcement, Car Repair, Insurance	Law Enforcement, Insurance, Other Drivers
Action	Read	Read	Read	Read

In 2015, we applied AB concept for secure exchange of video data, where the result of face detection algorithm is used in access control policies [31]. Providing privacy of humans appearing in a video, captured by vehicle's dash camera, is essential and AB ensures the data privacy. For instance, access control policy, specified by vehicle or dash camera manufacturer or by a vehicle owner, may allow viewing captured video fragments with human faces only for vehicle owner and Law Enforcement entities. Face detection algorithm employs OpenCV open-source library [73] and is applied to every frame of a captured video. For incoming data requests, video is recreated from the processed frames, based on specified access control policies, such that for entities other than vehicle owner or Law Enforcement, frames with human faces will be cut out [31].

Automotive industry imposes strict requirements on vehicle safety. Cybersecurity mechanisms, involving data encryption/ decryption and digital signatures creation/ verification, must not compromise vehicle's safety. "V2X systems need to have a balanced approach among safety, security, and Quality of Services (QoS)" [6]. Based on EA3B performance evaluations, presented in chapter 2.3, EA3B should not be applied for time-critical vehicular systems, such as braking system or engine control, where time to react and make decisions is less than 10ms. However, it can be applied in Intelligent Transportation Systems and V2X communication networks for tasks without strict real-time requirements, such as traffic jam notification or notification about vehicles exceeding a speed limit.

2.1.9 Isolated Execution of EA3B

EA3B can be hosted and executed at:

1. Recipient side
2. Server side
3. In the cloud (including untrusted clouds)

To ensure that there is no damage from executing EA3B's jar file for the recipient service, the EA3B can be executed inside Linux Docker container to serve data requests (see Fig. 2-9).

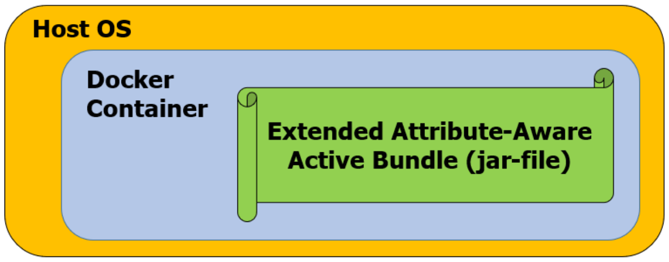


Figure 2-9. EA3B Isolated Execution in Docker Container

Another option is to execute EA3B in cloud so that services send data request to the cloud instance, which hosts and executes EA3B. To evaluate EA3B performance, I deployed EA3B on Amazon EC2 and Google clouds. Results are presented in chapter 2.3.

2.2 Related Work

To my knowledge, there is no comprehensive solution that provides all the features, offered by EA3B, including detection of leakages, made by authorized insiders to unauthorized entities; fine-grained role-based and attribute-based access control; data protection in transit and at rest for both centralized and peer-to-peer service architectures with support of RESTful API; capabilities of analytics and search through encrypted data, supporting a subset of SQL queries; collecting provenance data and providing their integrity.

A mechanism of micro-policies [2] enforced at a browser's side was proposed to provide confidentiality and integrity of web sessions. The mechanism, implemented as a Google Chrome extension *Michrome* [11], can be used to ensure secure access to web data by means of http(s) protocol. Micro-policies are specified in terms of tags, used to label URLs, network connections, cookies, etc; and a transfer function, which monitors security-relevant operations based on these tags and defines which operations are permitted by the browser. In my approach, in contrast, the access control policies are enforced in the EA3B, by its policy and attribute-enforcement engine. My solution provides the following advantages: (1) data leakage detection for multiple scenarios of leakages that can be made by authorized insiders [7]; (2) trust level of clients can be constantly monitored and recalculated and it can be considered by attribute-based access control model [5]; (3) modification of web browser's source code is not required; (4) data storage mechanism allows to store requested data records locally in encrypted form and provides capabilities of search through encrypted data with support of SQL queries subset [6].

A privacy-preserving information brokering (PPIB) system was proposed “to preserve privacy of multiple stakeholders involved in the information brokering process” [23]. This approach relies on centralized TTP to manage keys, metadata, joining and leaving brokers. Centralized TTP creates a single point of failure. Suggested methodology does not allow to detect leakages that can be made by authorized insiders to unauthorized entities.

Pearson et al. proposed “*EnCoRe”* solution [16] for secure data dissemination when the recipients are not known in advance. To manage data privacy “*EnCoRe”* uses sticky policies –

"conditions and constraints attached to data that describe how it should be treated" [16]. Data and policies are made inseparable. Sticky policies are enforced by a TTP. However, integrity of sticky policies is not guaranteed and they are prone to attacks from malicious recipients. The approach itself is prone to TTP – related issues, such as single point of failure.

Ranchal et al [33] proposed a framework for Enforcing Security Policies in Composite Web Services (EPICS), which protects data privacy throughout the service interaction lifecycle. The framework instantiates and extends the AB concept [1] for SOA. The solution ensures that the data are distributed along with the client policies that dictate data access, and with an execution monitor that controls data disclosure. The framework empowers data owners with control of data disclosure decisions outside their trust domains and reduces the risk of unauthorized access. My solution extends this approach with the following capabilities:

a) Detection of multiple scenarios of data leakages that can be made by authorized insiders to unauthorized parties. Leaked data might include data which were already extracted from an AB by an authorized party (insider).
b) Search through encrypted data records.
c) Set of attributes used in attribute-based access control includes the level of client's web browser cryptographic capabilities and type of the device (mobile vs. desktop).
d) Data provenance is collected and recorded in a blockchain public ledger, such that every transaction is non-repudiatable and can be verified at any time in the future (implementation is in progress).

Table 2-4. Related Work for Data Protection in Transit and at Rest

	MiChrome [2], [11]	EnCoRe [16]	PPIB [23]	EPICS [33]
RBAC	YES	YES	YES	YES
ABAC	YES	YES	YES	YES
DLP	YES	NO	YES	YES
DLD	NO	NO	NO	NO
Place for policy enforcement	Client's browser	TTP	TTP	Recipient, TTP, Server
ABAC re-computes trust level	NO	NO	NO	YES
Data transfers in P2P network	YES	YES	YES	YES

2.3 Evaluation

I measured performance for services sending data requests to an EA3B. Data request Round-trip Time (RTT) is measured between times of sending a data request and data retrieval from an EA3B. RTT is a sum of times spent for authentication, evaluation of access control policies and client's attributes, data leakage checks (if the data leakage detection feature is enabled), decryption key derivation and data retrieval. *ApacheBench* utility (version 2.3), as well as browser developer consoles (for Firefox and Chrome web browsers) were used for RTT evaluations.

2.3.1 Experiment 1. Local Performance Evaluation of EA3B

In this experiment, I aim to measure the latency of a data request sent to EHR, represented as EA3B, which is hosted by the Hospital Server, located on the same host as the requesting service (client). The Hospital Server that hosts EHR has the following characteristics:

Hardware: MacBook Pro, Intel Core i7 CPU @ 2.2 GHz, 16GB memory

OS: macOS Sierra 10.12.6.

The client sends a request for Patient ID (16 bytes of data) to the EHR, represented as EA3B, and to the basic AB, which supports neither extended attribute-based access control nor tamper-resistance. In contrast, EA3B supports tamper-resistance and extended attribute-based access control with checking cryptographic capabilities of the client's browser [5]. The Basic AB, as well as EHR, contains four access control policies. The access control matrix for EHR is shown in Table 2-1. Basic AB and EHR (EA3B) are running on a Hospital Server, located on the same host. I measure RTT for data request processing at the server side, and do not consider network delays between client and server in this experiment. Results in Fig. 2-10 represent latency when a first initial data request is sent to EHR. I consider it a special case, since the very first request to the basic AB and to EHR takes significantly longer to be executed due to the initial authentication phase and initial evaluation of attributes. The EHR with embedded attribute-based access control and tamper resistance imposes a 12.9% performance overhead as compared to a Basic AB. Tamper-resistance imposes performance overhead because the digest of an AB is validated by its kernel whenever the data request arrives. Detection of the cryptographic capabilities of client's browser and checking whether it is sufficient imposes extra overhead.

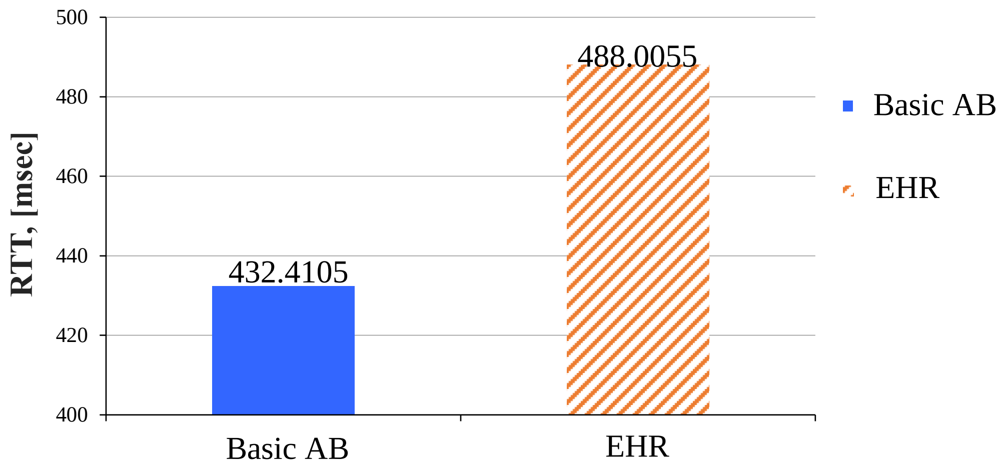


Figure 2-10. EHR Round-trip Time (initial data request). ©2018 Springer

In the next experiment, I run 50 similar data requests for Patient ID (16 bytes of data) in a row. As shown in Fig. 2-11, mean RTT has been decreased 33.5 times for a Basic AB and 35.9 times for EHR. Having embedded attribute-based access control and tamper resistance in EHR imposes a 5.2% performance overhead as compared to a Basic AB.

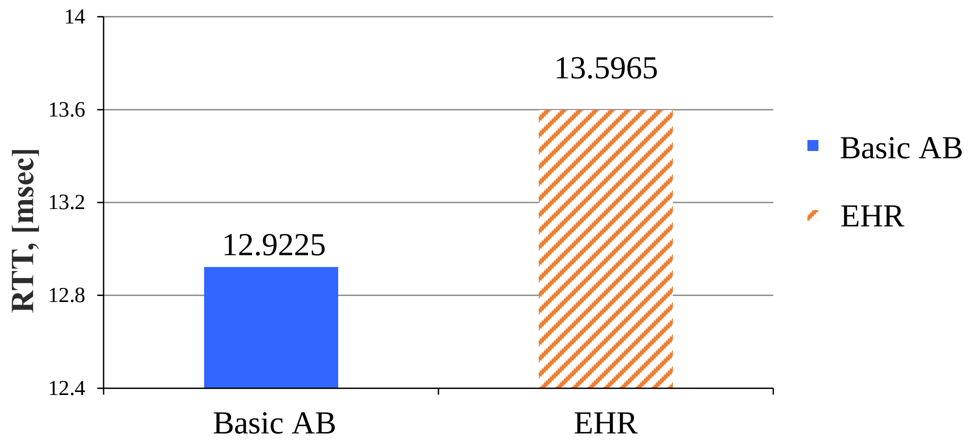


Figure 2-11. EHR Round-trip Time. ©2018 Springer

2.3.2 Experiment 2. Performance Evaluation of EA3B, Hosted by Google Cloud

In this experiment, I aim to measure the latency of a data request sent to EHR which is hosted by Google Cloud Provider and has the following characteristics:

Hardware: Intel(R) Xeon(R) CPU 2.30GHz

OS: Linux Debian 4.9.65-3+deb9u2 (2018-01-04) x86_64, kernel 4.9.0-5-amd64

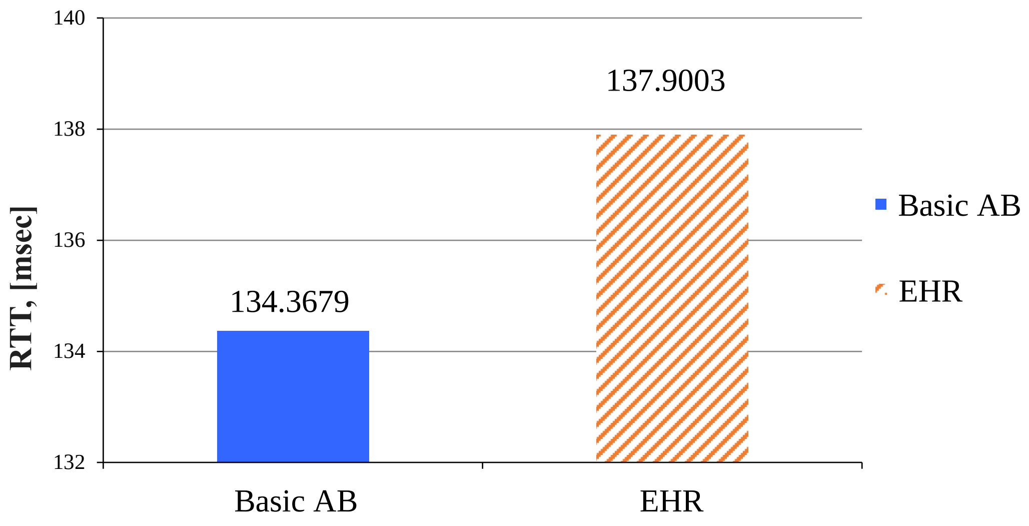


Figure 2-12. EHR Round-trip Time for Cloud (initial request). ©2018 Springer

The procedure is same as in experiment 1, but now the client queries a Basic AB and EHR, running on a Google cloud instance. For the data request, I measure the overall RTT that includes network delays between client and cloud server in this experiment. Results in Fig. 2-12 represent latency when the first initial data request is sent to an EHR. As pointed in the previous experiment 1, I consider it a special case. EHR with embedded attribute-based access control and tamper resistance imposes a 2.6% performance overhead as compared to a Basic AB.

In the next experiment, I run 50 similar data requests for Patient ID (16 bytes of data) in a row. As shown in Fig. 2-13, mean RTT has been decreased 5.85 times for basic AB and 5.59 times for EHR. Having embedded attribute-based access control and tamper resistance in EHR imposes a 7.4% performance overhead as compared to a Basic AB.

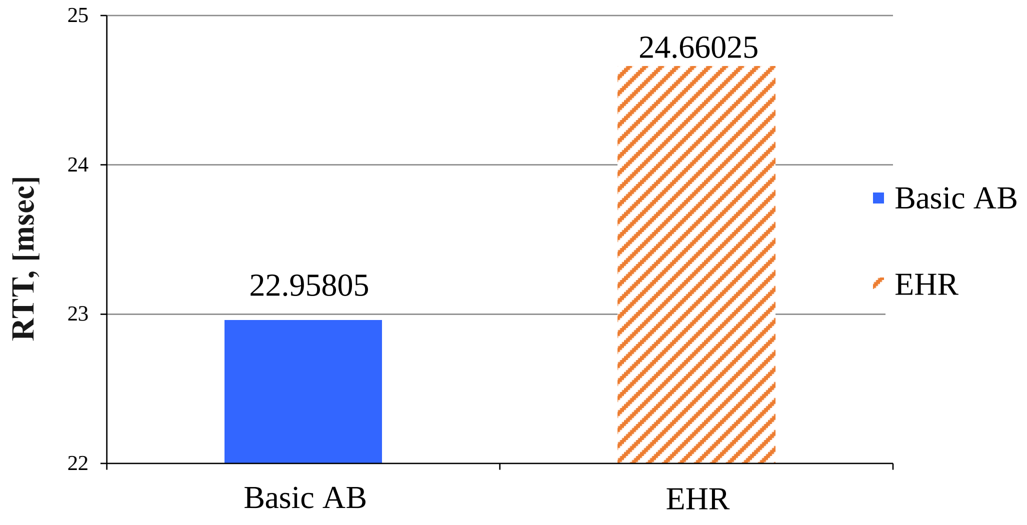


Figure 2-13. EHR Round-trip Time for Cloud. ©2018 Springer

2.3.3 Experiment 3. Evaluation of EA3B Performance Overhead Imposed by Tamper-Resistance Feature and Detection of Web Browser's Cryptographic Capabilities

Hardware: Intel Core i7, CPU 860 @2.8GHz x8, 8GB DRAM

OS: Linux Ubuntu 14.04.5, kernel 3.13.0-107-generic, 64 bit

Browser: Mozilla Firefox for Ubuntu, ver. 50.1.0

In the following experiment (see Fig. 2-14), I use EA3B which, in addition to tamper-resistance, supported by AB, supports client's browser cryptographic capabilities and authentication method detection. Local request for a patient's Contact Information (197 Bytes of data) is sent to the EA3B, which represents EHR and runs on a Purdue University Server. I use AB and EA3B with 8 access control policies, similar in terms of complexity. Examples of access control policies are given in Table 2-1. Data request is issued from the service running on the same host with AB and EA3B. Thus, I measure RTT for a local data request and network delays are excluded. Instead of ApacheBench tool, in this experiment I measured RTT through web browser's developer console.

Tamper-resistance support adds 12.3% performance overhead since the hash value of AB and its modules (code, access control policies) is verified by AB's kernel when the data request comes. The same applies to EA3B. Support of client's browser cryptographic capabilities and authentication method detection imposes additional performance overhead of 82.8%. RTT for a data request increases because now, before responding to the client's request, EA3B needs to check the cryptographic capabilities of the browser and authentication method, which are sent to an EA3B by means of https message.

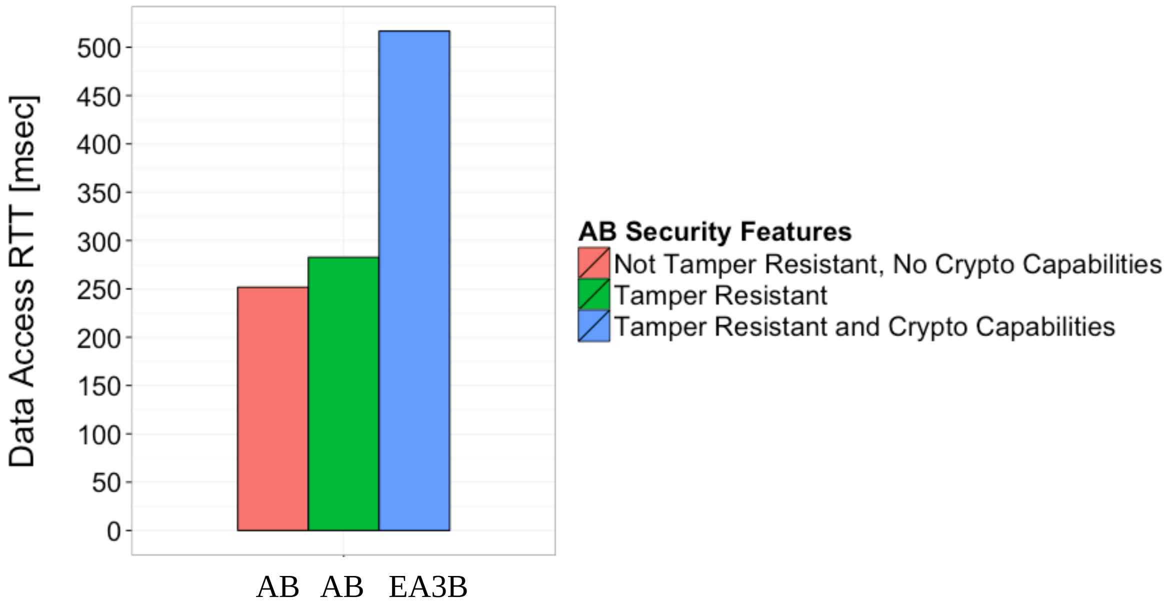


Figure 2-14. AB/ EA3B Performance Overhead. ©2017 IEEE

2.3.4 Experiment 4. Round-trip Time Evaluation for Inter-vehicular Communication

In this experiment, I simulate inter-vehicular communication by having two Raspberry Pi’s communicating with each other. The first vehicle (Raspberry Pi) hosts a VR in the form of an EA3B and listens to the opened port 5555. The second vehicle (also represented by a Raspberry Pi) sends data request to that VR over TCP/IP network, port 5555. Vehicle ID field (16 bytes of data) is requested. I measured data request latency that also includes network delays.

Vehicles 1, 2 (Raspberry PI 3 Model B)

Hardware: ARMv7 Processor rev 4 @1.2GHz, RAM 1GB

OS: Raspbian GNU/Linux 9.1 (stretch)

Results on Fig. 2-15 represent the inter-vehicle communication round trip time when the first initial data request is sent to by the first vehicle to the second one. I consider it as a special case since the very first request to VR takes significantly longer to be executed due to initial authentication phase and initial evaluation of attributes. Using a VR with embedded attribute-based access control and tamper resistance, adds 6.4% performance overhead.

In the next experiment, I run 50 data requests for Vehicle ID (16 bytes) in a row. As it can be seen from Fig. 2-16, mean RTT value has been decreased 4.64 times for the Basic AB and 4.37 times for the VR. Using a VR with embedded attribute-based access control and tamper resistance, adds 13% performance overhead.

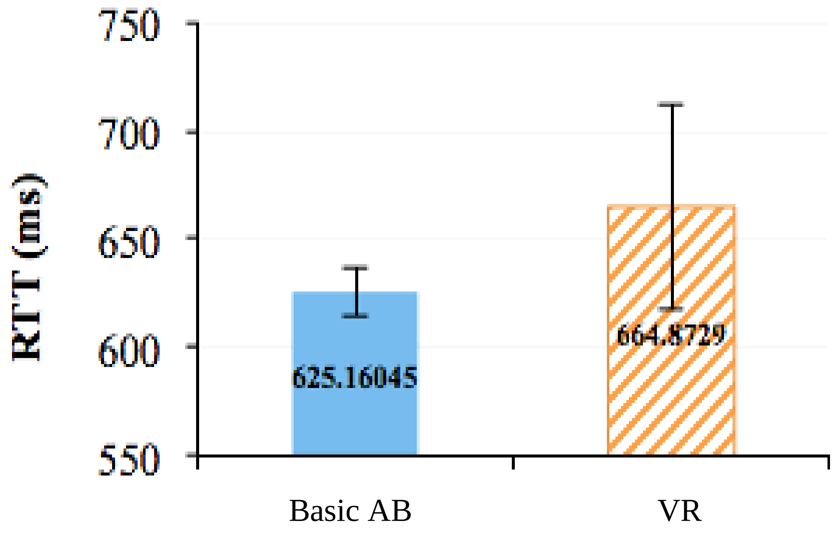


Figure 2-15. Inter-vehicular Communication Round-trip Time (initial request). ©2018 IEEE

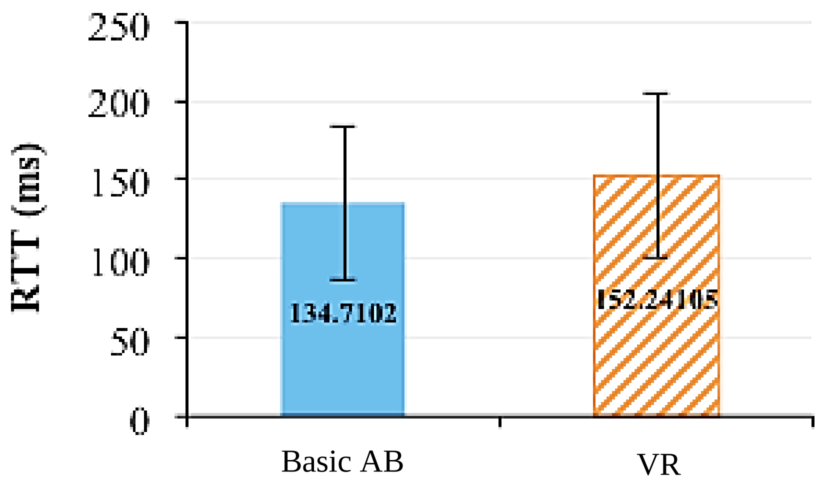


Figure 2-16. Inter-vehicular Communication Round-trip Time. ©2018 IEEE

2.3.5 Experiment 5. Round-trip Time Evaluation for Data Requests to Vehicle Records, Stored in Cloud

In this experiment, I aim to measure the latency of a data request sent to VR that is hosted by a Google Cloud instance with the following characteristics:

Hardware: Intel(R) Xeon(R) CPU 2.30GHz

OS: Linux Debian 4.9.65-3+deb9u2 (2018-01-04) x86_64, kernel 4.9.0-5-amd64, 64 bit

Request for a Vehicle ID (16 bytes) is sent to VR, represented as an EA3B, and to a Basic AB that, in contrast with VR, supports neither extended attribute-based access control nor tamper-resistance nor data leakage detection. Both VR and Basic AB run on a Google cloud instance. Basic AB, as well as VR, has 4 access control policies. Examples of access control policies are given in Table 2-3. Data request is issued from the service to the Basic AB and VR, running on a remote cloud service, which has an external IP address. Thus, I measure RTT for a cloud data request and consider network delays between vehicle and cloud provider.

Results on Fig. 2-17 represent the latency when the first initial data request is sent to cloud. I consider it a special case since the very first request to AB and VR takes significantly longer to be executed due to initial authentication phase and initial evaluation of attributes. Using a VR with embedded attribute-based access control and tamper resistance (EA3B), adds 5.1% performance overhead.

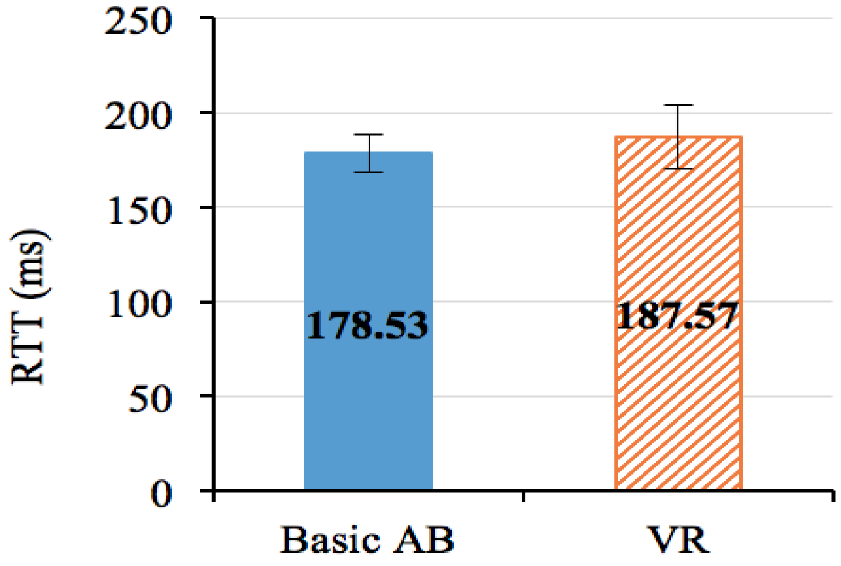


Figure 2-17. Round-trip Time for Vehicle Record, Hosted by Google Cloud (initial request). ©2018 IEEE

In the next experiment, I run 50 similar data requests in a row. As it can be seen from Fig. 2-18, the mean value of RTT has been decreased 2.57 times for the Basic AB and 2.62 times for the VR. Tamper-resistance support adds overhead since the hash value of AB modules (code, access control policies) is verified by an AB kernel when a data request arrives. Detection of client's browser cryptographic capabilities and authentication method impose extra overhead since before responding to the client's request, an EA3B needs to know the level of the browsers cryptographic capabilities and to check whether it is sufficient. Authentication method detection is also processed by an EA3B kernel. Using a VR with embedded attribute-based access control and tamper resistance, adds 3.1% performance overhead.

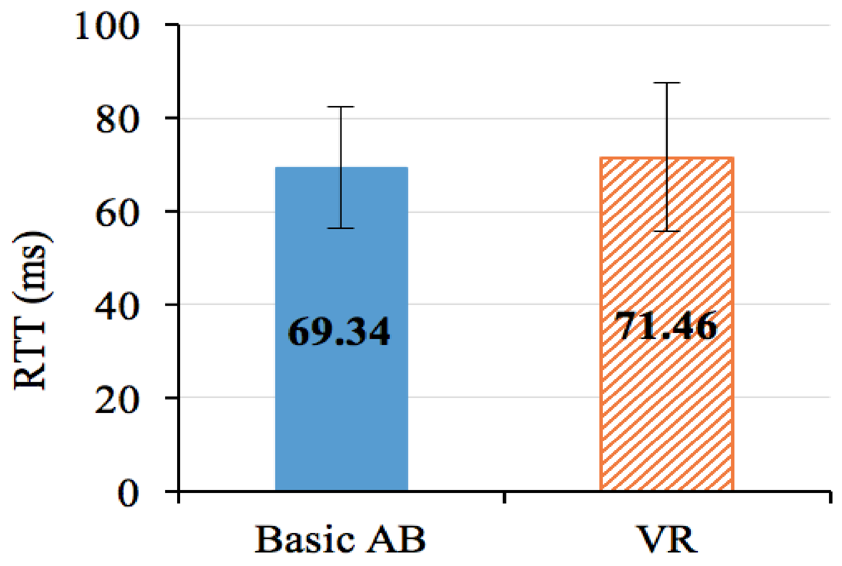


Figure 2-18. Round-trip Time for Vehicle Record, Hosted by Google Cloud.
©2018 IEEE

2.3.6 Experiment 6. Evaluation of Inter-vehicular Communication Round-trip Time vs. Baseline Decryption

In this experiment, I compare a Round-trip Time for a data request sent to VR with just decryption of the same (encrypted) dataset without involving evaluation of access control policies and client's attributes. In both cases, client requests the same dataset of 617 bytes, stored in encrypted form. In the first case, encrypted dataset is just decrypted, using an AES algorithm. In the second case, request for the same dataset needs to go through evaluation of access control policies and client's attributes, as well as leakage detection check. Our VR, represented as an EA3B, incorporates four access control policies and uses an AES algorithm to encrypt and decrypt the stored data. As it can be seen from Fig. 2-19, VR imposes 102% performance overhead compared to just decryption of the encrypted dataset. I assume that Raspberry Pi board, which has a small credit-card size and moderate power consumption, represents the vehicle's communication hardware.

Hardware: Raspberry PI 3 Model B, ARMv7 Processor rev 4 @1.2GHz, RAM 1GB

OS: Raspbian GNU/Linux 9.1 (stretch)

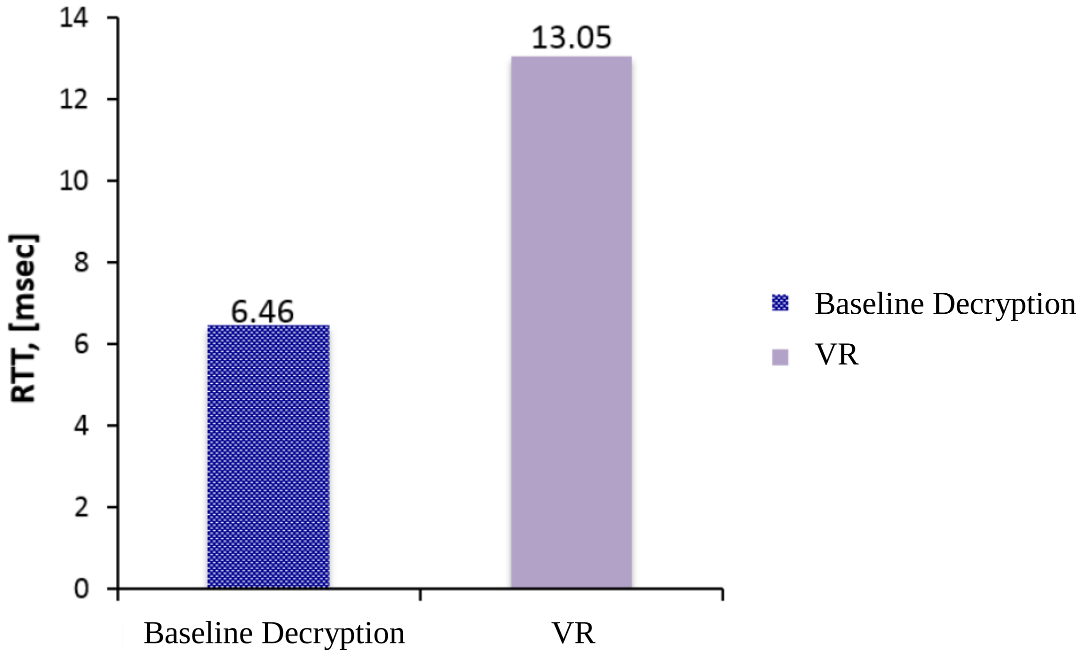


Figure 2-19. Inter-vehicular Communication Round-trip Time. ©2018 ICAEEE

2.3.7 Experiment 7. Performance Overhead Evaluation for EA3B with 4 Policies and for ABs Depending on Number of Included Policies

In this experiment, I evaluate a Round-trip Time of an inter-vehicular data request sent to VR, represented as an EA3B. Then I compare the performance of VR with a Vehicle Data Record Baseline (VDRB). VDRB is represented as a Basic AB and, in contrast with VR, does not support data leakage detection, fast on-the-fly data analytics and detection of cryptographic capabilities of client's web browser. Furthermore, VDRB in this experiment is not tamper-resistant. We varied the number of access control policies, included in VDRB. Experiment covers 1, 2, 4, 8 and 16 access control policies. VR contains four access control policies. Thus, conclusion about VR performance overhead is made based on comparing the RTT for VDRB(4) and VR.

I assume that Raspberry Pi 3 board Model B represents the vehicle's communication hardware.
Hardware: Raspberry PI 3 Model B, ARMv7 Processor rev 4 @1.2GHz, RAM 1GB
OS: Raspbian GNU/Linux 9.1 (stretch)

The first vehicle hosts a VR and listens to the opened port 5555. The service, representing the second vehicle, sends a request for 617 bytes of data to the first vehicle over wireless TCP/IP network. Measured RTT includes network delays. As it can be seen from Fig. 2-20, VR adds 127% performance overhead, compared to the baseline VDRB(4) [24].

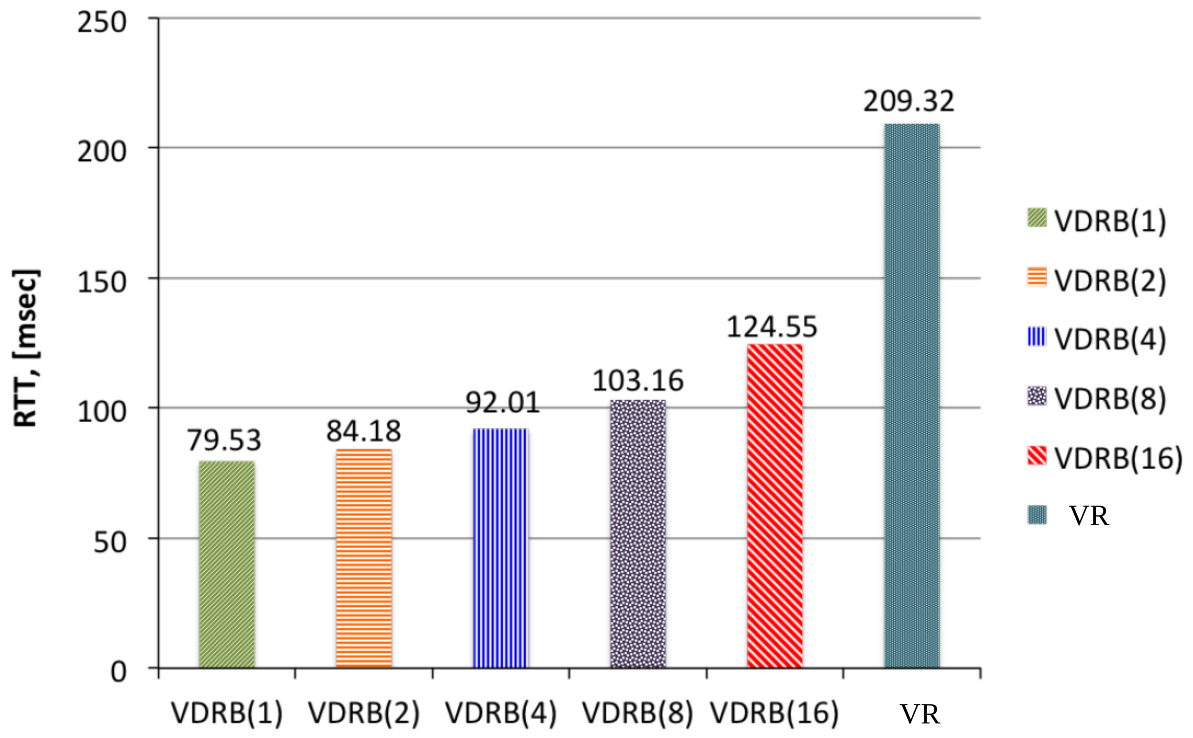


Figure 2-20. VR Performance Overhead for Inter-vehicular Communication. Basic AB (VDRB) Performance Evaluation Depending on Number of Policies. ©2018 ICAEEE

3. DATA LEAKAGE PREVENTION AND DETECTION

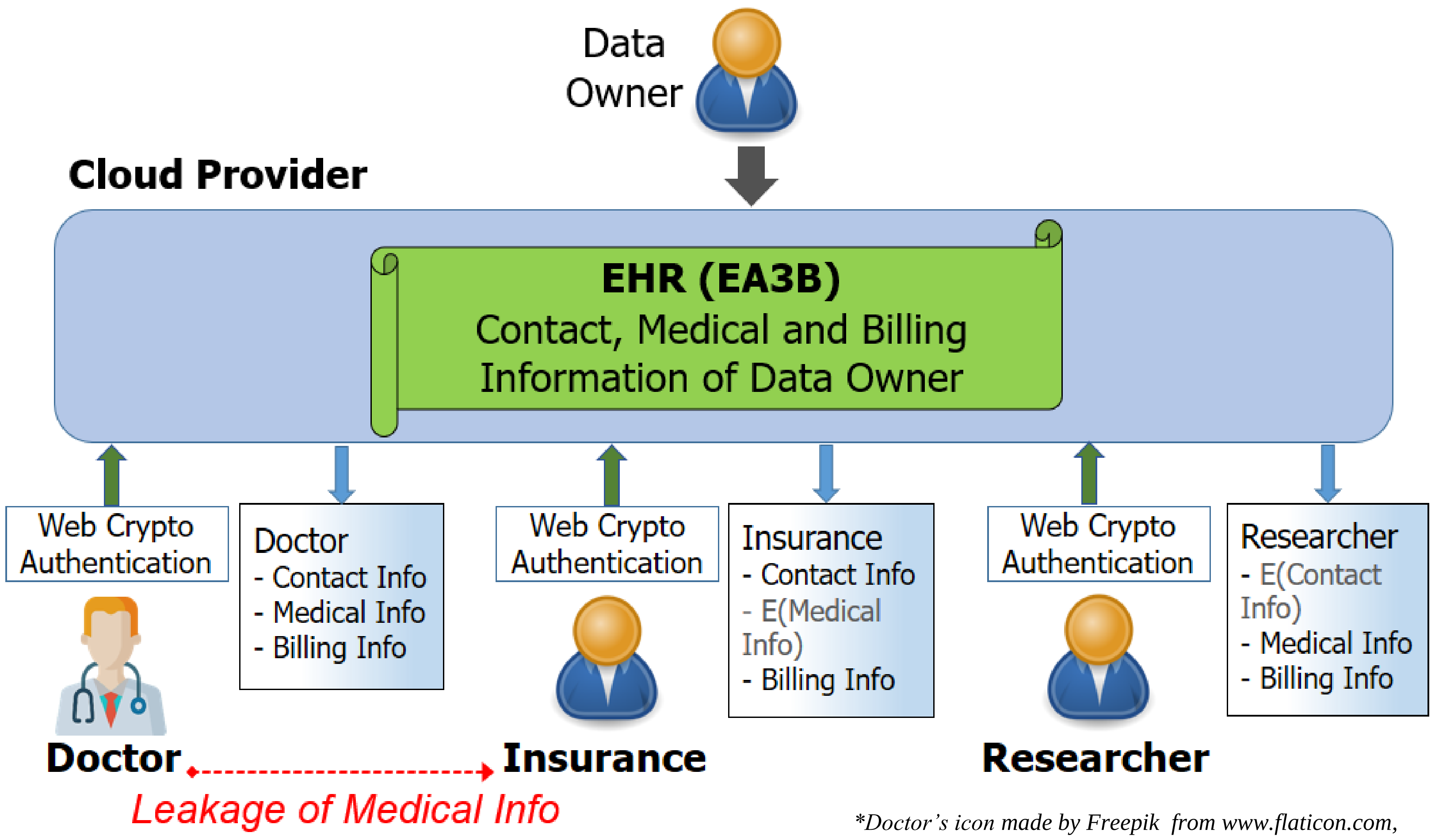
Portions of this chapter have been reprinted with permission from the following:

Springer Nature Customer Service Centre GmbH : Springer Singapore, In: Deka G., Kaiwartya O., Vashisth P., Rathee P. (eds), Applications of Computing and Communication Technologies ICACCT 2018, Communications in Computer and Information Science, vol 899, pp. 99-113. Ulybyshev, Denis, Bharat Bhargava, Aala Oqab-Alsalem. "Secure Data Exchange and Data Leakage Detection in an Untrusted Cloud." ©2018
https://doi.org/10.1007/978-981-13-2035-4_10

In SOA, authorized services (insiders) may leak data behind the scene to unauthorized entities. In this chapter, I present a solution that allows preventing data leakages and detecting multiple types of data leakages. I also propose a damage assessment model for data leakages. The implemented prototype, which is an extended version of a WAXEDPRUNE project, described in chapter 2, provides protection of EHRs that can be hosted by untrusted cloud providers, in transit and at rest. Furthermore, WAXEDPRUNE allows detecting several types of data leakages made by insiders.

3.1 Data Leakage Prevention

AB, which is a self-protecting structure, discussed in chapter 2, provides data confidentiality and integrity. It supports fine-grained role-based and attribute-based access control. But it does not address data leakages that can be made by malicious insiders. A service, e.g., "Doctor" in a WAXEDPRUNE project, is authorized to access medical information of a patient from patient's EHR, including diagnosis; test results; prescriptions; etc. (see Fig. 3-1). Doctor may leak, inadvertently or intentionally, medical information to an Insurance Agent, who is not authorized to access it. EA3B allows preventing and detecting several types of such data leakages. In this work, I address two scenarios of leakages: the whole EA3B is leaked; and the plaintext, extracted from an EA3B by authorized insider, is leaked. The former leakage type is addressed in this section 3.1, the latter one is addressed in section 3.2.



**Doctor's icon made by Freepik from www.flaticon.com, https://www.flaticon.com/free-icon/doctor_1021606*

Figure 3-1. Data Leakage in WAXEDPRUNE. ©2018 Springer

Leakage prevention is guaranteed by the following:

1) EA3B that stores data and access control policies in encrypted form. If the whole EA3B is leaked to unauthorized service, data leakage is prevented by an EA3B kernel, which prohibits unauthorized data accesses. For instance, according to the access control policies, embedded into EA3B, shown in Table 2-1, Insurance service is not allowed to access medical information of a patient and is only allowed to access contact and billing information.
2) Digital watermark, embedded into EA3B. Digital watermark is verified by a web crawler, provided an EA3B is available for a crawler, which checks whether EA3B is supposed to be where it is discovered.

If a service requests from an AB the data for which the service is not authorized, then the AB kernel will deny this request. An EA3B, furthermore, notifies a trusted Central Monitor (CM) each time the service requests data from EA3B. The CM checks against its database of access control policies whether the requested data is accessible by the service who is requesting it. An attempt to decrypt data made by any service is recorded by the CM and stored as a provenance record [36]. Every EA3B transaction in the data exchange network is monitored by the CM, who is notified each time a client tries to decrypt a data subset from an EA3B. The notification message contains information on what service attempts to decrypt what type of data, when, who is the origin / sender of EA3B. CM queries its local database of access control policies (see Table 3-1), in order to check whether the service that tries to decrypt data is authorized for that class of data. Detecting leaking service is similar to traitor tracing problem [29]. Without obtaining a permission from the CM, data decryption process will not continue. Fig. 3-2 illustrates the process. EA3B contains encrypted data

$$\mathbf{Enc\ [Data(D)] = \{Enc_{k1}\ (d_1), \ldots , Enc_{kn}\ (\ d_n) \}}$$

and access control policies

$$\mathbf{(P) = \{p_1,.., p_k\}}.$$

Service M is authorized to read d_1 and it may leak decrypted d_1 (addressed in the Section 3.2) or the entire EA3B to service N, who is not authorized to access d_1. If service N attempts to decrypt d_1, the EA3B kernel sends a message to the CM to verify whether d_1 is supposed to be at N. In addition, service N might be asked to get an activation code from the authentication server, which is under our control, and which will again notify the CM that data of type d_1 has arrived from service M to service N. If d_1 is not supposed to be at N then:

1) trust level of services M and N is decreased;
2) data d_1 is marked as compromised and all other services are notified about that;
3) EA3B is re-created with stricter access control policies to make it stronger against similar leakages:
    3.1) separate compromised role (of service M) into *Role* and *Trustworthy_Role*;
    3.2) send new certificates with *Trustworthy_Role* to all trustworthy entities;
    3.3) create a new EA3B with modified policies to prohibit data access for *Role*;
    3.4) raise the sensitivity level for leaked data types to prevent leakage repetition
    3.5) disable the "Save As" functionality to prohibit storing sensitive data locally. By default, authorized services in WAXEDPRUNE are allowed to store EA3B locally. It might help e.g., Doctor, who needs to work with a local copy of an EHR when the central EHR storage, hosted by a cloud provider or by a Hospital server, is unavailable.

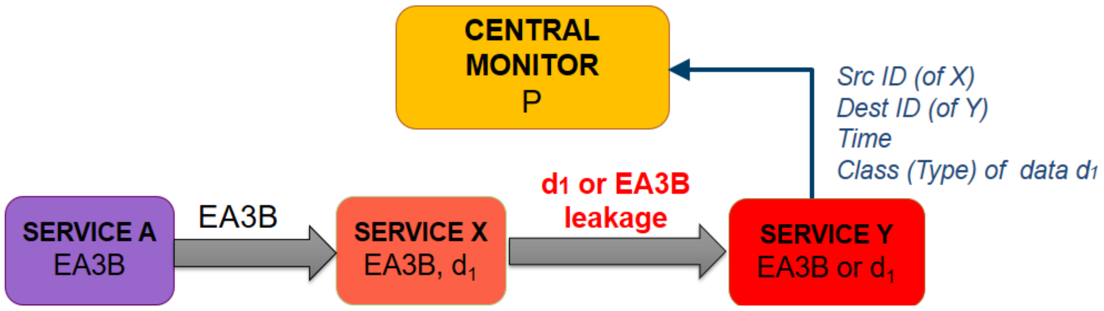
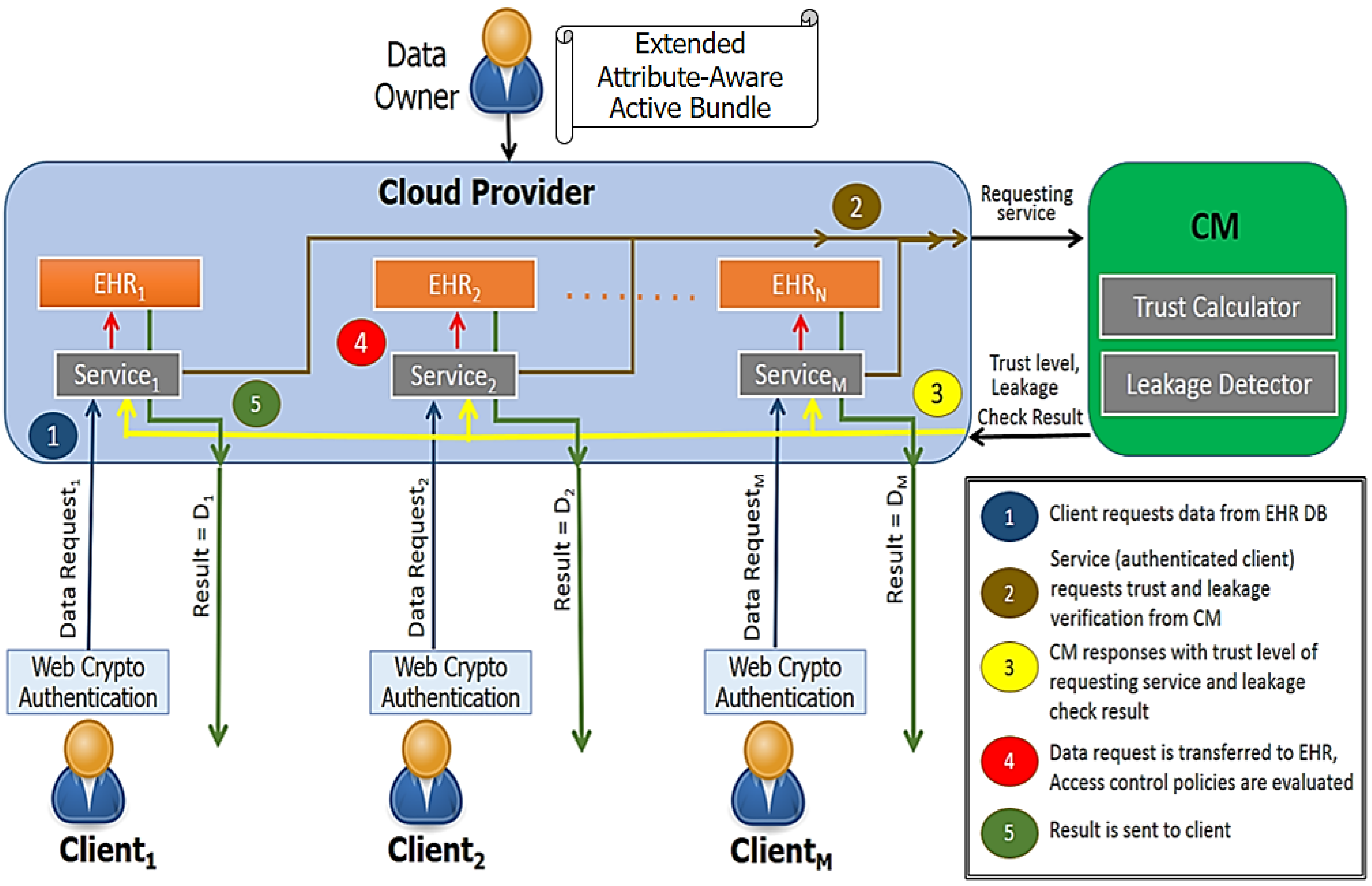


Figure 3-2. Data Leakage Detection by Central Monitor. ©2018 Springer

WAXEDPRUNE architecture with the data leakage detection support is illustrated on Fig. 3-3. There is a cloud provider that stores EHRs of patients in the form of EA3Bs, one EA3B per patient. The client service, which wants to request data from an EHR first needs to authenticate itself. WebCrypto Authentication module is responsible for that. Details of the authentication process are explained in Fig. 2-7. Initial data request from unauthenticated client, which is http get request, is redirected from a Cloud Provider to an Authentication Server (AS) web page where client needs to enter valid credentials in order to receive a valid authentication ticket. The level of cryptographic capabilities of client's browser and client's authentication method are determined and are included to the authentication ticket, which is signed by AS.



Data Owner
Extended Attribute-Aware Active Bundle
Cloud Provider
EHR$_1$
EHR$_2$
EHR$_N$
Service$_1$
Service$_2$
Service$_M$
Requesting service
CM
Trust Calculator
Leakage Detector
Trust level, Leakage Check Result
Data Request$_1$
Result = D$_1$
Data Request$_2$
Result = D$_2$
Data Request$_M$
Result = D$_M$
Web Crypto Authentication
Client$_1$
Client$_2$
Client$_M$
1 Client requests data from EHR DB
2 Service (authenticated client) requests trust and leakage verification from CM
3 CM responses with trust level of requesting service and leakage check result
4 Data request is transferred to EHR, Access control policies are evaluated
5 Result is sent to client

Figure 3-3. WAXEDPRUNE Architecture with Leakage Detection Support. ©2018 Springer

If entered credentials are valid then the AS redirects client's data request to the service, corresponding to the client's role (Doctor, Insurance or Researcher), with the authentication ticket attached. Once a corresponding service running in cloud receives the data request and authentication ticket, the signature, client ID and ticket expiration time, extracted from the ticket, as well as access control policies are evaluated, based on the client's role [5]. The cloud-side service extracts client’s role and requested data type from the authentication ticket and then queries CM to check for a trust level of client’s service and for data leakage. Trust calculator checks the current trust level of client’s service. Trust level is constantly re-calculated by CM based on the following metrics:

(a) number of sent/received data requests,

(b) number of rejected data requests,

(c) number of communication errors,

(d) CPU/Memory usage.

If the trust level of client service goes below the specified threshold, data requests coming from that service to EA3B will be denied, even if access control policies allow that service to access a certain data subset. If the trust level of service is sufficient for the data request, then a Leakage Detector issues a SQL query to its local relational database of access control policies (see Table 3-1) to check whether the client’s role is allowed to access the requested data subset. If not then data request is denied and leakage alert is raised. Database of access control policies stored at the CM contains same policies as EA3B. The reason for such a redundant storage is as follows. If a service requests data it is not authorized for then the EA3B kernel will just deny the request but we need a leakage alert, in addition to that, since someone tried to access and decrypt unauthorized data. For raising a leakage alert, I cannot really rely on a local EA3B kernel, which rejected unauthorized data access, since in decentralized architecture services can host EA3Bs. If a hosting service is malicious, then it can shut down network connectivity such that an EA3B kernel, which detected unauthorized data access attempt, will not be able to send an alert notification to anybody. That is why I rely on a trusted Central Monitor, which monitors every data transaction and queries its local trusted database of access control policies.

If a leakage is detected or the trust level of a client service is not sufficient, then the data request is denied.

If a Leakage Detector approves the data request, then the corresponding cloud-side service forwards the data request to the EHR of a patient, stored as an EA3B. EA3B extracts client's role and attributes from the attached authentication ticket. Then EA3B evaluates client's role against access control policies, client's browser cryptographic capabilities and client's authentication method. Based on these evaluations, EA3B responds to the requesting cloud-service with the data, for which the client is authorized. Then the cloud-side service transfers authorized data to the client by means of https protocol.

In addition to the access control policies enforced by the trusted CM to detect data leakages, I implemented a web crawler to verify a digital watermark, which is embedded into EA3B. If EA3B is stored in a publicly available directory in the network, the web crawler verifies the digital watermark to check whether an EA3B is supposed to be at that network node. I assume that it is possible to determine the identity of the network node that hosts a public directory (e.g., in the Hospital Intranet). Network nodes, participating in data exchanges in WAXEDPRUNE, are required to have X.509 certificates that identify their roles (e.g., Doctor, Insurance agent or Researcher). X.509 certificate of the service, which is part of WAXEDPRUNE network, is stored in a directory, known for a web crawler. If an X.509 certificate is missing or directory is not available, the crawler tries to check the IP address of the host, where EA3B is discovered. Then the crawler checks the detected IP address against its databases of whitelisted and blacklisted IP addresses. The web crawler's configuration can be adjusted based on the use case needs. For instance, a conservative strict configuration would raise a data leakage alert if X.509 certificate of the host, where EA3B is discovered, is missing or corrupted or IP address of the host is not in the whitelist of allowed IP addresses.

Table 3-1. Access Control Policies at Central Monitor in WAXEDPRUNE. ©2018 Springer

Recipient's Role	**Sender's Role**	**Data Type**	**Access Result**
Doctor	Doctor	All	ALLOW
Doctor	Insurance	All	ALLOW
Doctor	Researcher	All	ALLOW
Insurance	Doctor	All	DENY
Insurance	Insurance	All	ALLOW
Insurance	Researcher	All	DENY
Researcher	Doctor	All	DENY
Researcher	Insurance	All	DENY
Researcher	Researcher	All	ALLOW
Doctor	Doctor	Medical	ALLOW
Insurance	Doctor	Medical	DENY
Researcher	Doctor	Medical	ALLOW
Doctor	Insurance	Medical	DENY
Insurance	Insurance	Medical	DENY
Researcher	Insurance	Medical	DENY
Insurance	Researcher	Medical	DENY
…	…	Billing	ALLOW
Researcher	Researcher	Billing	ALLOW
Doctor	Doctor	Contact	ALLOW
Insurance	Doctor	Contact	ALLOW
Researcher	Doctor	Contact	DENY
Doctor	Insurance	Contact	ALLOW
Insurance	Insurance	Contact	ALLOW
Researcher	Insurance	Contact	DENY
Doctor	Researcher	Contact	DENY
Insurance	Researcher	Contact	DENY
Researcher	Researcher	Contact	DENY

3.2 Data Leakage Detection

The scenario when the plaintext, extracted from an EA3B by an authorized insider, is leaked to unauthorized entity, is challenging since the data protection provided by EA3B kernel is gone. Insider, e.g., Doctor, who is authorized to extract medical information of a patient from an EHR, represented as an EA3B, may leak extracted plaintext medical information in different forms. For instance, Doctor may take a picture of a screen, displaying X-Ray of a patient, on his/her smartphone and send it to unauthorized entity via email. To detect and mitigate data leakage, I employ the following methods:

a) Embedding *digital watermarks* into data, stored in an EA3B
b) Embedding *visual watermarks* on a web page, where data, retrieved from an EA3B, are displayed. Both visual and small nearly invisible watermarks are used
c) *Monitoring network traffic between web services in SOA*. For some data types, e.g., credit card number, it is possible detect that this type of data was sent from service A to service B
d) *Collecting provenance data* that contains information on who is trying to access what class of data from an EA3B, when, and who is the origin/sender of that EA3B.
e) *Classification level elevation for leaked data*. It will prevent leakage repetitions of the same data type
f) Honeypot “fake leakage” to lower the value of leaked real data and to create uncertainty

3.2.1 Digital Watermarks for RGB-images

When creating an EA3B, I embed digital watermarks into RGB (red-green-blue) images, before they are encrypted and stored in EA3B. Procedure is illustrated on Fig. 3-4. RGB values can vary between 0 and 255. The conversion function F (r, g, b) is applied to every pixel of an image. It changes the RGB image in such a way that it is indistinguishable by a human eye from the original RGB image. The simple way to modify the RGB image is to change RGB values for every pixel by adding or subtracting 1 in such a way that the sum of RGB values is always odd for every pixel. Initial values should be less than 255 for adding and greater than zero for subtracting. If we add or subtract value 1 to/from RGB values of every pixel, it will not be distinguishable by a human eye. However, my web crawler, which has a built-in classifier, is able to determine whether the RGB image has the embedded watermark or not. If all the pixels of the RGB image follow the rule of odd sum of RGB values, my classifier considers this image to be watermarked. Once watermark is detected, the CM is notified, and it checks whether the given RGB image is supposed to be at that network node. Fig. 3-5 illustrates digital watermark verification process. Same as described in section 3.1, trusted CM issues a SQL query to a relational database of access control policies, stored on a trusted site.

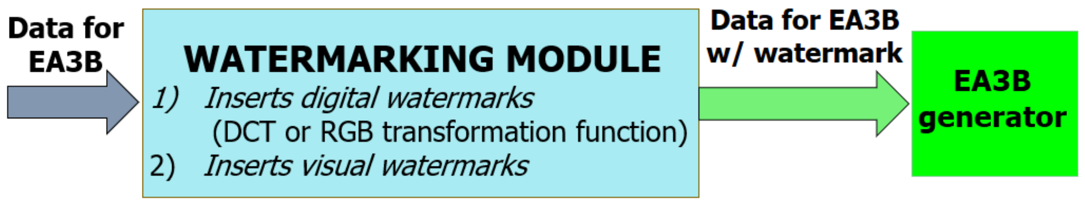

Data for EA3B
WATERMARKING MODULE
1) Inserts digital watermarks
(DCT or RGB transformation function)
2) Inserts visual watermarks
Data for EA3B w/ watermark
EA3B generator

Figure 3-4. Embedding Digital and Visual Watermarks into RGB images, Stored in EA3B

I assume that it is possible to determine the identity of the node that hosts a public directory. As discussed in section 3.1, network nodes, participating in WAXEDPRUNE data network, are required to store X.509 certificates that identify their roles in a directory, known for a web crawler. If an X.509 certificate is missing or directory is not available, crawler tries to check the IP address of the host, where watermarked image is discovered. Then the crawler checks the detected IP address against its databases of whitelisted and blacklisted IP addresses. The complications arise if IP addresses are allocated dynamically and whitelists/ blacklists need to be updated periodically.

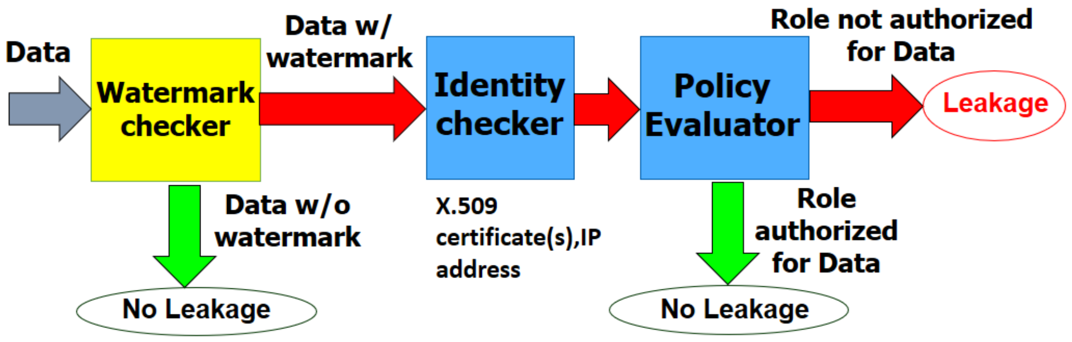


Figure 3-5. Digital Watermark Verification Process

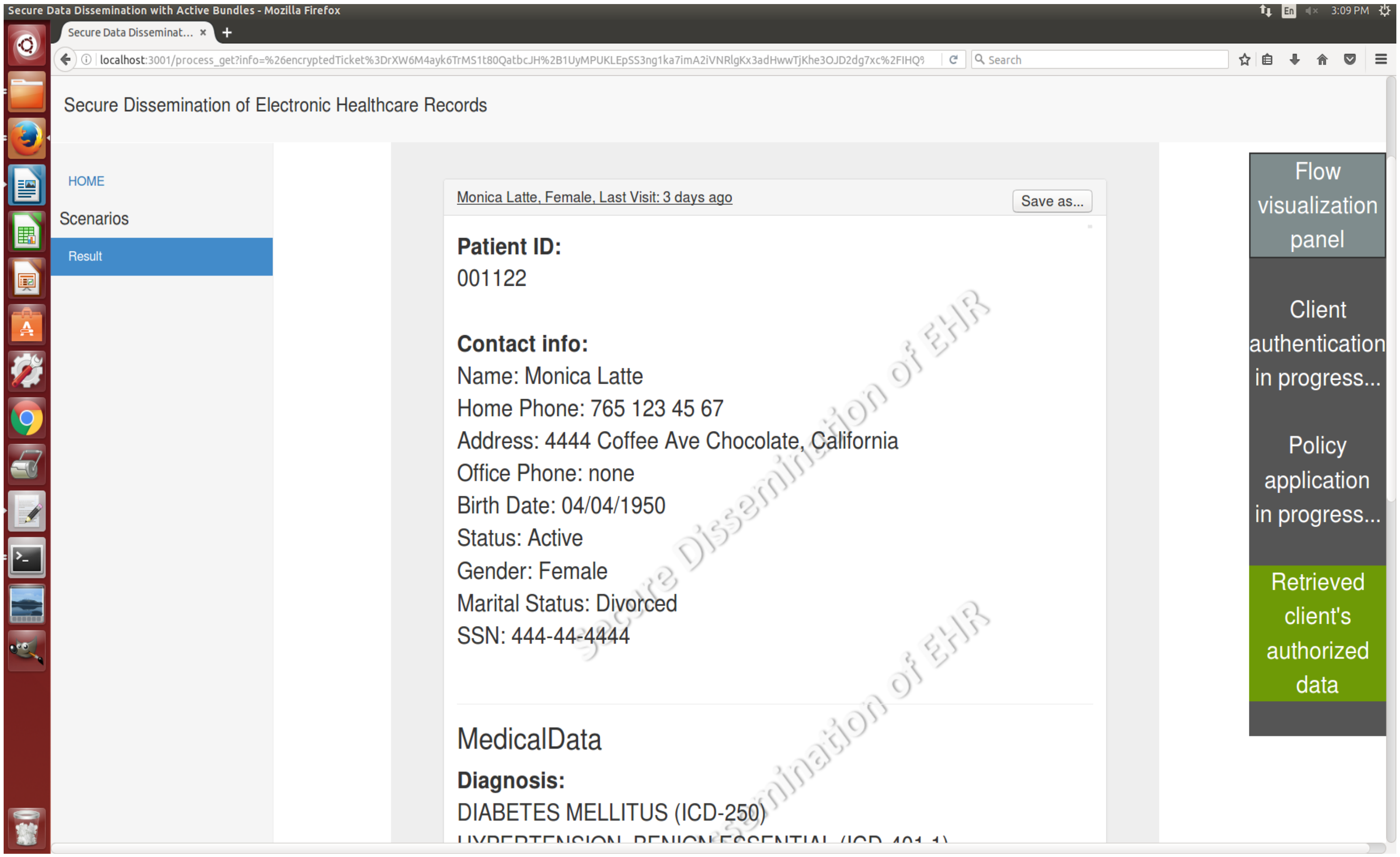
More secure conversion function F (r, g, b) than discussed above, should be applied to every pixel. For RGB-images, Digital Cosine Transformation (DCT) functions [20] can be used.

To provide more robust watermark verification, a pixel conversion function for images periodically changes. Watermark checker is notified about the conversion function change and knows which function to use for a watermark verification. Whereas an attacker might still be trying to break the old conversion function in order to remove a watermark from data and mislead a watermark checker.

3.2.2 Visual Watermarks

To help detect data leakages, a web page, which displays authorized data, retrieved from an EA3B, for the authorized service, contains visual watermarks. Furthermore, some data types, such as images, pdf and word documents, videos, etc., also contain an embedded visual watermark. This will help to detect leakage of e.g., X-Ray image. There are two types of visual watermarks embedded into data and/or a web page which displays authorized data on a client side in a web browser: clearly visible (see text “Secure Dissemination of EHR” on Fig. 3-6) and nearly invisible ones that are only visible if zoomed (upper right corner of displayed EHR on Fig. 3-6). These watermarks will remain on the image if picture of a screen is taken by a malicious client. For some types of data, e.g., diagnosis on Fig. 3-6, it is easy to reproduce the screen’s content and write it down e.g., on a piece of paper. It removes the visual watermark. However, for some types of medical data, e.g., X-Ray images, it is hard to reproduce them on a piece of paper and easier to take picture of a screen, which will contain both our visual watermarks: large visible and small nearly invisible.

Visual watermarks can be used as an evidence of data ownership, e.g., in a court.


Secure Data Dissemination with Active Bundles - Mozilla Firefox
Secure Data Disseminat...
localhost:3001/process_get?info=%26encryptedTicket%3DrXW6M4ayk6TrMS1t80QatbcJH%2B1UyMPUKLEpSS3ng1ka7imA2iVNRlgKx3adHwwTjKhe3OJD2dg7xc%2FIHQ
Search
3:09 PM
Secure Dissemination of Electronic Healthcare Records
HOME
Scenarios
Result
Monica Latte, Female, Last Visit: 3 days ago
Save as...
Flow visualization panel
Client authentication in progress...
Policy application in progress...
Retrieved client's authorized data
Secure Dissemination of EHR
Secure Dissemination of EHR

Patient ID:
001122

Contact info:
Name: Monica Latte
Home Phone: 765 123 45 67
Address: 4444 Coffee Ave Chocolate, California
Office Phone: none
Birth Date: 04/04/1950
Status: Active
Gender: Female
Marital Status: Divorced
SSN: 444-44-4444

MedicalData

Diagnosis:
DIABETES MELLITUS (ICD-250)

Figure 3-6. Visual Watermarks in WAXEDPRUNE. ©2018 Springer

3.2.3 Monitoring Network Traffic between Web Services in SOA.

CM monitors every network message in a WAXEDPRUNE framework. Message monitoring imposes overhead, but it allows to detect certain types of messages that a web service sends. Validating data packets with a specific pattern allows detecting data leakage for cases when data pattern can be validated. For instance, a credit card number always follows the specific pattern that can be validated using regular expressions [14]. If the CM detects that service A sent a message, containing a credit card number, to some network destination (IP address), CM verifies against its trusted database of access control policies whether a credit card number is allowed to be at that destination IP address. As discussed in sections 3.1 and 3.2.1, network nodes, participating in WAXEDPRUNE data network, are required to store X.509 certificates that identify their roles in a known directory. However, complication arise if several web services with different roles are hosted on the same host with the same IP address, since TCP/IP message only specifies destination IP address. Another difficulty comes from the fact that IP addresses can be allocated dynamically and whitelists/ blacklists need to be updated by CM periodically.

3.2.4 Provenance Data for Leakage Investigation

Provenance data is recorded locally and at the CM each time when a data request is served by an EA3B. Provenance records contain information regarding who is trying to access what class of data subset, when, and the origin of EA3B. Recorded provenance records help to investigate data leakages and do forensics. If leaked data is used by an attacker, e.g., stolen credit card number is used for online shopping, provenance data helps to identify the services involved in working with that credit card number and narrow down the list of suspects. However, there are two problems with recording provenance data:

a) For provenance data stored locally at a client side, there is no integrity guarantee since a malicious client can corrupt provenance record. To address that, I propose to apply blockchain-based technology, so that every data transaction is recorded in a public ledger and can not be corrupted without being noticed by other network nodes. Details are discussed in chapter 5.
b) Storing provenance records on CM imposes significant overhead and makes the system centralized with a single point of failure. One of the main selling points of AB/EA3B methodology is that it provides data protection in decentralized peer-to-peer networks

3.2.5 Classification Level Elevation for Leaked Data

The idea is to raise the classification level for leaked data class to prevent leakage repetition. For instance, if a doctor who is authorized to see medical data of a patient leaks medical data to an insurance agent who is not authorized to see it (see Fig. 3-1), and this leakage is detected by any of the methods discussed above, then EHRs of patients need to be re-created to prevent future leakages of medical data for other patients. The following steps are done:

a) separate compromised role into *Role* and *Trustworthy_Role,* e.g., *Doctor and Trustworhy Doctor*;
b) issue new X.509 certificates with *Trustworthy_Role* and send them to all trustworthy entities; e.g., all trustworthy doctors. Malicious doctor is identified by credentials, he/she used to login to WAXEDPRUNE system. Even though the role "Doctor" is the same for all doctors, WAXEDPRUNE collects provenance data for a particular doctor who logged in
c) create EA3Bs for all the patients whose EHRs and medical data are accessible by malicious doctor. New EA3B contains modified access control policies which prohibit access to medical information for *Role*, i.e., for Doctor. New EA3Bs allow access to medical information for *Trustworthy_Role*, and malicious doctor will not receive the renewed X.509 certificate with *Trustworthy_Doctor* role

Optionally, some other functionality for malicious doctor can be also disabled. In WAXEDPRUNE, the functionality of storing EHRs locally is disabled for a malicious doctor, involved in data leakage incident.

3.2.6 Reissuance vs. Revocation of X.509 certificates

As discussed in Section 3.2.5, if a role is compromised then this role is divided into *Role* and *Trustworthy_Role.* For a new *Trustworthy_Role*, a new X.509 certificate is issued and sent to all services that belonged to *Role* and that were not involved in leakage incident(s). The services that belong to *Role* and that were involved in leakage incident do not receive the updated X.509 certificate for *Trustworthy_Role*. Thus, their role remains to be *Role* and they still can access the old data, e.g., in the old EHRs, for which the *Role* was authorized. But those services are not authorized to access new data, e.g., in new EHRs. For instance, if Service A with the role "*Doctor*" leaked diagnosis to an Insurance agent, who is not authorized to access diagnosis, and this leakage is detected, then the role *Doctor* is divided into “*Doctor*” and “*Trustworthy_Doctor*”. All services but A receive a new X.509 certificate with a role “*Trustworthy_Doctor*”. In new EHRs, the access to a “diagnosis” data field is granted only to a role “*Trustworthy_Doctor*”, but not to “*Doctor*”. Such mechanism requires to re-issue (n-1) X.509 certificates and to send (n-1) messages to services, one per service, with a new X.509 certificate, where n is a total number of services with the role *Doctor* in a network. It should be said that in the very beginning of a data request process, Doctor has to authenticate itself in a system by entering credentials on the Authentication Server login page (see Section 2.1.6 and Fig. 2-7). Thus, if a leakage is detected then the system (Central Monitor) knows what username, e.g., which Doctor exactly, is involved in a leakage incident within a role *Doctor*.

The alternative solution is to use a Certificate Revocation List (CRL), which is a list of certificates revoked by CA before their expiration date. In this case, every time when a service authenticates itself to EA3B, the CA checks the status of X.509 certificate of a service, before trusting it. It makes the system more centralized since for every data request to EA3B, the CA needs to be queried to check the status of X.509 certificate of a requesting service. In my approach I tried to make the framework as decentralized as possible, so that it is not needed to send a message to the CA to check X.509 certificate status every time when a data request arrives to EA3B. The downside of my approach is that, instead of revoking just one X.509 certificate, (n-1) new certificates need to be re-issued and sent to (n-1) services, where n is number of services with the compromised role *Role*.

3.2.7 “Fake” Leakage

The goal of a fake leakage is to lower the value of leaked real data and to create uncertainty. For instance, if diagnosis of a patient was leaked by Doctor to an Insurance agent, who is not authorized to see it, and this data leakage is detected, then CM may send several fake diagnoses of the same patient to an Insurance agent to lower the value of real leaked diagnosis and to confuse an Insurance agent on which diagnosis is correct. The confusing fake data might be sent to the suspicious service in advance.

3.3 Data Leakage Damage Assessment

Leakages of different types of data may have different consequences in terms of damage caused by leakage. Data leakage damage in WAXEDPRUNE is evaluated using the following information [36]:

1. How malicious is the recipient of unauthorized data;
2. How sensitive is the leaked data
3. Leakage timing: when was the data leaked

Here is the suggested formula to calculate data leakage damage in WAXEDPRUNE:

$$\textbf{DMG} = \mathbf{K_{ds}}\ \textbf{(Data Sensitivity)} * \mathbf{K_{rm}}\ \textbf{(Recipient Maliciousness)} * \mathbf{F(t)} \qquad (1)$$

DMG is the damage coefficient from the interval [0, 1]. DMG = 1 means the most severe damage from data leakage, DMG = 0 means no damage. K_{ds} denotes data sensitivity coefficient, it varies from 0 to 1. $K_{ds} = 1$ for the most sensitive data. K_{rm} is the coefficient of recipient maliciousness coefficient. It is greater for services with lower trust value or services previously involved in leakage incidents. F(t) is the data sensitivity function, which can take values from 0 to 1. Fig. 3-7 illustrates different data sensitivity functions. The data event (e.g., course final exam) happens at time t_0. Damage from data being leaked before t_0 is high. Damage from data being leaked after t_0 can be as follows:

(1) immediately drops to zero: e.g., final exam is leaked after the exam is over – see type 1 on Fig. 3-7;

(2) decreases linearly: e.g., for leaked data protection technique, which is examined by users and attackers and with time it might be broken so that leaking it makes no damage since it is easy to find information in Internet on how to bypass the given data protection technique – see type 2 on Fig. 3-7;

(3) remains high: e.g., for sensitive technology – see type 3 on Fig. 3-7.

Coefficients K_{ds}, K_{rm} and data sensitivity function are assigned by WAXEDPRUNE administrator or cybersecurity expert(s).

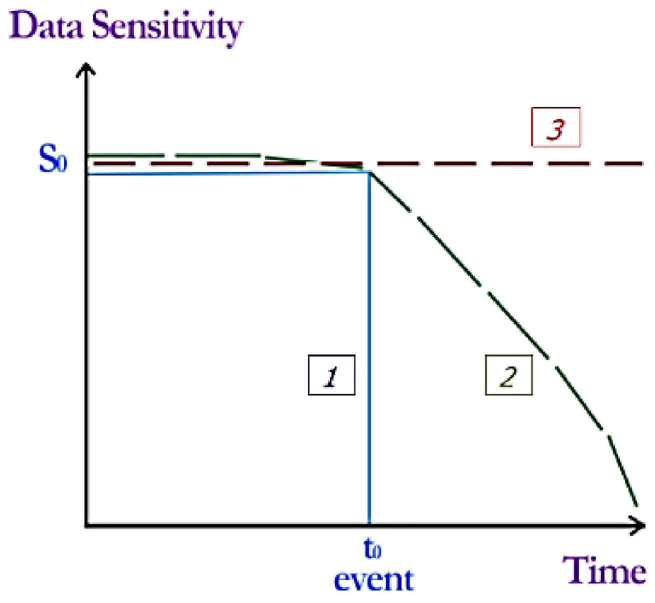


Figure 3-7. Data Sensitivity Functions. ©2018 Springer

3.4 Related Work

There are variety of Digital Rights Management (DRM) protection tools [38], [74] that provide role-based access control. These tools are supposed to protect against data leakages. Microsoft ® has a DRM-service, called Windows Media DRM [22]. It is designed to provide dissemination of audio and video content over an IP network. The "MediaSnap ®" DRM solution [39] was proposed to protect PDF documents. Most of its principles are applicable to other digital media content. The core component of the "MediaSnap ®" system is a pdf-plugin. Our data exchange model considers the context and client's attributes, such as the trust level, which is constantly recalculated, cryptographic capabilities of a browser and authentication method.

"SandMark" [25] tool supports several code obfuscation and software watermarking algorithms. This tool aims "to study the effectiveness of software-based methods for protecting software from piracy, tampering, and reverse engineering" [25].

The hardware-based DRM approach provides a trusted hardware space for executing only permitted applications. "DRM services such as content decryption, authentication and rights rendering take place only in this trusted space" [38]. Advantages of hardware-based DRM are that it is resistant to security breaches in operating systems, it is infeasible to bypass security features and it provides memory space protection. The main disadvantages are higher costs, limited flexibility and less interoperability [40].

Nevase et al [41] proposed a steganography-based approach to detect leakages of images, text, video and audio content. Steganography provides a covert communication channel between data entity and data owner by hiding the message in the sensitive content, e.g., in the image or text. The existence of sensitive message in the content is hidden for everyone except the data owner, who is able to decipher it. Steganography-based "forensic readiness model" [42] identifies and prevents emails, which attempt to leak data.

Kaur et al [43] also addressed prevention of data leakages that can be made by malicious insiders via emails. Email is protected via gateway during data transfer. The algorithm matches the email pattern with the stored keywords in order to detect leakage and take the action to prevent it.

Gupta and Singh [44] presented an approach for detecting intentional and inadvertent data leakages using a probabilistic model. Detecting a data leaking malicious entity is based on the allocation and distribution of data items among the agents using Bigraph.

3.5 Evaluation

3.5.1 Experiment 1. EA3B's Performance Overhead Evaluation with Enabled Data Leakage Detection

In this experiment, I measure performance overhead imposed by a *data leakage detection* feature, added to EA3B, used in WAXEDPRUNE framework. There are four web services in the framework: Hospital, Doctor, Researcher and Insurance. These services are running as NodeJS servers. Local data requests for 617 bytes of patient's data is sent from a client service with the role 'Doctor', from the web browser, to the EHR, hosted by the Hospital Information System at localhost in the form of EA3B. Network delays do not affect RTT measurements. The hospital server has the following specification:

Hardware: Intel Core i7, CPU 860 @2.8GHz x8, 8GB memory

OS: Linux Ubuntu 14.04.5, kernel 3.13.0-107-generic, 64 bit

Web Browser: Mozilla Firefox, version 50.1.0

Doctor requests all available information of a patient (617 bytes). EA3Bs, used in Data Leakage OFF/ON scenarios, contain the same data and access control policies. The tamper-resistance mechanism and the client's browser cryptographic capabilities detection are the same and are used in both scenarios. The only difference is the data leakage detection feature. If it is enabled, it requires the CM to examine every data access before accessible data can be retrieved from the EHR, if a leakage is not detected. As shown in Fig. 3-8, data leakage detection support adds 60.8% performance overhead. Before approving or denying a data request, CM issues a SQL query to the trusted relational database access control policies (see Table 3-1) in order to check whether the requested data is accessible by the requesting client service. Then either data request is denied or approved. If it is approved, the decryption of accessible data starts. EA3B also contains metadata of its origin and who currently hosts it.

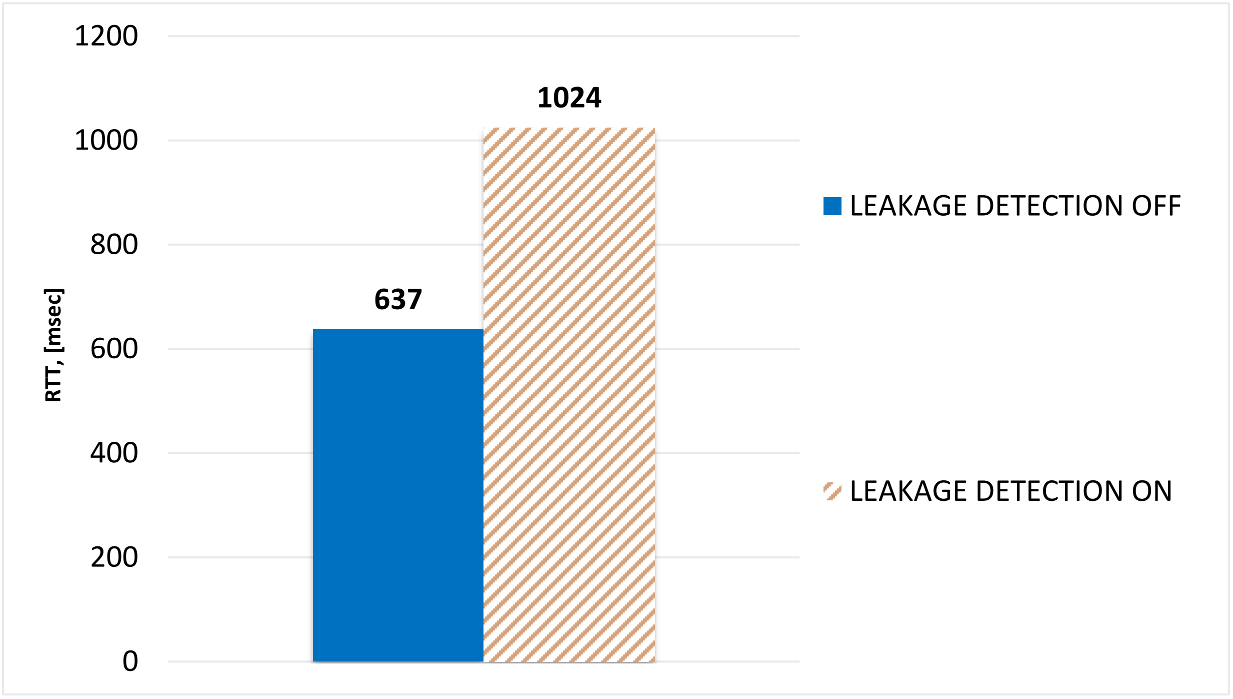


Figure 3-8. EA3B Performance with Data Leakage Detection OFF/ON. ©2018 Springer

4. SEARCH AND ANALYTICS OVER ENCRYPTED DATA

4.1 Search through Encrypted Data

EA3B-based solution provides data protection in transit and at rest. An EA3B stores data in encrypted form. It is highly desirable to have capabilities of running a search through encrypted

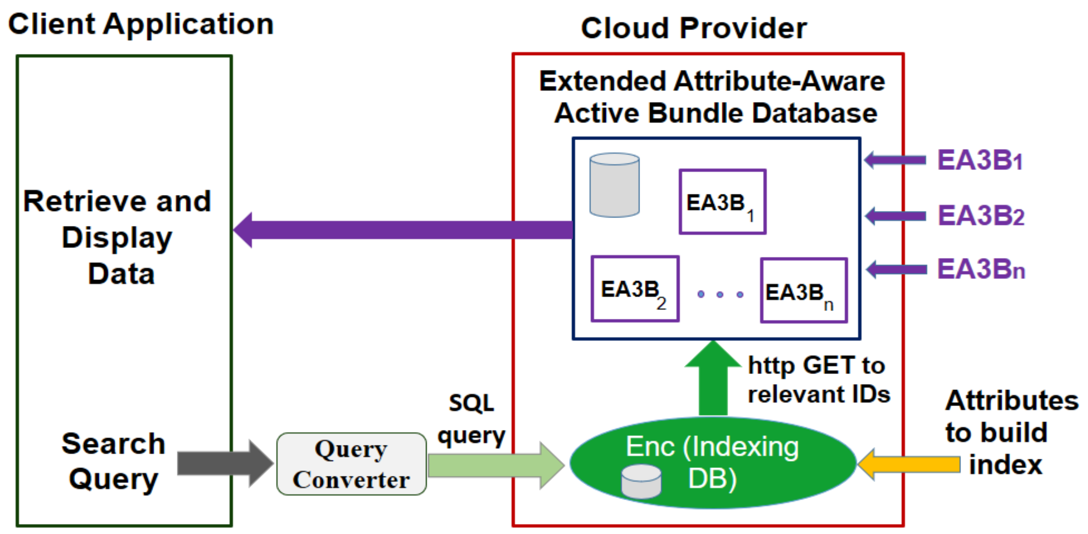


Figure 4-1. Search through EA3Bs Database. ©2018 IEEE

data, stored in the form of EA3Bs. My solution is to create an extra-database, let us call it Indexing Database, with attribute(s), which can be used for building index and for search through a database of EA3Bs. Attributes of Indexing Table in Indexing Database are stored in an encrypted form and may include EA3B's keywords or other attributes used to build an index (see Fig. 4-1). The value of a unique "ID" attribute maps each tuple from an Indexing Table to its corresponding EA3B with the same value of key = "ID". For instance, ID can be a patient ID used in EHR of a given patient.

As a use case, I developed a solution to provide protection of Vehicle Records (VRs) in transit and at rest in Vehicle-to-Everything (V2X) communication system. Its architecture is illustrated on Fig. 2-8. VRs are stored in the form of EA3Bs and contain vehicle's and vehicle owner's information as well as information on road events (see Table 2-2). As a motivating example, let us discuss how to find in a database of VRs the driver's license numbers of those drivers who exceeded speed limit and drove over 90 mph. Since recorded speed and driver's license number are fields (keys), stored in VR with corresponding encrypted values, the straightforward solution to search through encrypted data would be to send data requests for speed and driver's license numbers to all EA3Bs, decrypt all accessible data, store them on trusted site and then perform search. Search through decrypted pairs (speed, driver's license number) will then find records with speed > 90 mph and then retrieve corresponding driver's license numbers for those records. This will impose significant overhead. Data request to an EA3B needs to go through authentication phase, policy and attributes evaluation and data decryption phase. My solution is to create Indexing Table in Indexing Database, add attributes, e.g., "Speed", used for search, to it and then split search in 2 phases.

Phase 1

Query Indexing Table to retrieve relevant EA3Bs by retrieving "ID"s.

Here is the corresponding SQL query for our example with finding drivers violating speed limits:

```
select ID from IndexTB where speed > speed_limit
```

Phase 2

Send http GET request for driver's license number to relevant VRs, i.e., to VRs with IDs returned by the SQL query in the first phase.

As illustrated in Fig. 2-8 and explained in section 2.1.8, once vehicle reaches the proximity of a base station, VR is created on a vehicle's side as an EA3B. VR includes owner's information, such as driver's license number and name; vehicle's information, such as VIN and license plate numbers; and information on road events, such as traffic jams and road hazards (see Table 2-2). Then created VR is sent to a cloud provider via a base station as a relay. In the proximity of a base station, vehicle sends ID, speed and model every 5 seconds (see Fig. 4-2), these attributes are stored in indexing database in encrypted form on a cloud provider (see Table 4-1).

In my implementation, I tested two Database Management Systems (DBMS) with support of data encryption and SQL queries over encrypted data: CryptDB [9] and Microsoft SQL Server 2016 in "Transparent Data Encryption" (TDE) [45] and "Always Encrypted" (AE) [46] modes.

For a CryptDB-based architecture, the attributes including ID, speed and model are sent from a vehicle in plaintext as an "INSERT" SQL query to cloud provider via trusted proxy server, where data are encrypted. Alternatively, road camera can capture these data once the vehicle passes by and then send it to cloud provider instead of a vehicle. Modern highway cameras can capture vehicle's speed, as well as license plate number and model and then send it to trusted proxy, where data are encrypted and then sent to cloud provider in encrypted form.

For Microsoft SQL Server-based architecture in AE mode, which is supported starting from Microsoft SQL Server 2016, the attributes, including ID, speed and model, are encrypted on a client side, i.e., on a vehicle. Then vehicle sends encrypted data to a cloud provider in encrypted form using base station as a relay (see Fig. 4-2). Microsoft SQL Server in AE mode, as well as CryptDB, provide protection against malicious database administrator and data protection both in transit and at rest. In contrast, Microsoft SQL Server in TDE mode provides protection at rest only [47]. It is useful for HIPAA-compliancy, but in my scenario, I aim to provide data protection both in transit and at rest, for which only AE mode of Microsoft SQL Server fits. The problem with AE mode in Microsoft SQL Server with versions earlier than 2019 is that it does not support range or inequality SQL queries over encrypted data [47]. Support of inequality and range queries is announced for version 2019 with secure enclaves. As of February 2019, there only exists a Community Technology Preview (CTP) release of Microsoft SQL Server 2019 [48] and I did not test it.

Table 4-1. Indexing Table "IndexTB" for Vehicle Records. ©2018 IEEE

ID	**Speed**	**Model**	**Timestamp**
Enc(001)	Enc(65)	Enc(Toyota)	02/18/2018 15:28
Enc(002)	Enc(66)	Enc(Ford)	02/18/2018 15:29
Enc(003)	Enc(67)	Enc(Mercedes)	02/18/2018 15:31
Enc(004)	Enc(68)	Enc(Mitsubishi)	02/18/2018 15:44
⋮	⋮	⋮	⋮
Enc(1000)	Enc(94)	Enc(Ferrari)	02/18/2018 23:59

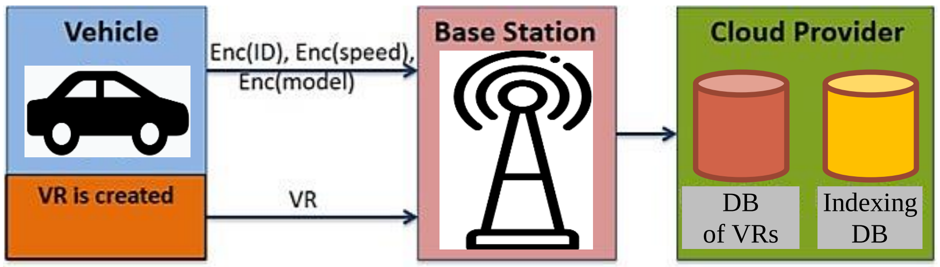


Figure 4-2. Microsoft SQL Server-based Database Architecture for AE mode. ©2018 IEEE

*Vehicle icon made by Freepik from www.flaticon.com, https://www.flaticon.com/free-icon/car-trip_72833

Antenna (base station) icon made by Freepik from www.flaticon.com, https://www.flaticon.com/free-icon/antenna_145093

On a cloud instance, each received record is timestamped. Fragment of indexing table, *IndexTB*, is given in Table 4-1. ID is a primary key and it maps every record to VR, which is also stored on the same cloud in a DB of VRs.

As another use case scenario, Law Enforcement may need to know license plate numbers of drivers who drove with a speed between 80 and 100 mph on a highway with a speed limit 70 mph. In the first phase, SQL query is sent to encrypted indexing table, IndexTB, to get IDs of all the vehicles which drove with the speed between 80 and 100 mph. In the second phase, http GET request for license plate number is sent to relevant VRs with IDs returned by SQL query in the first phase. Fig. 4-3 illustrates the workflow.

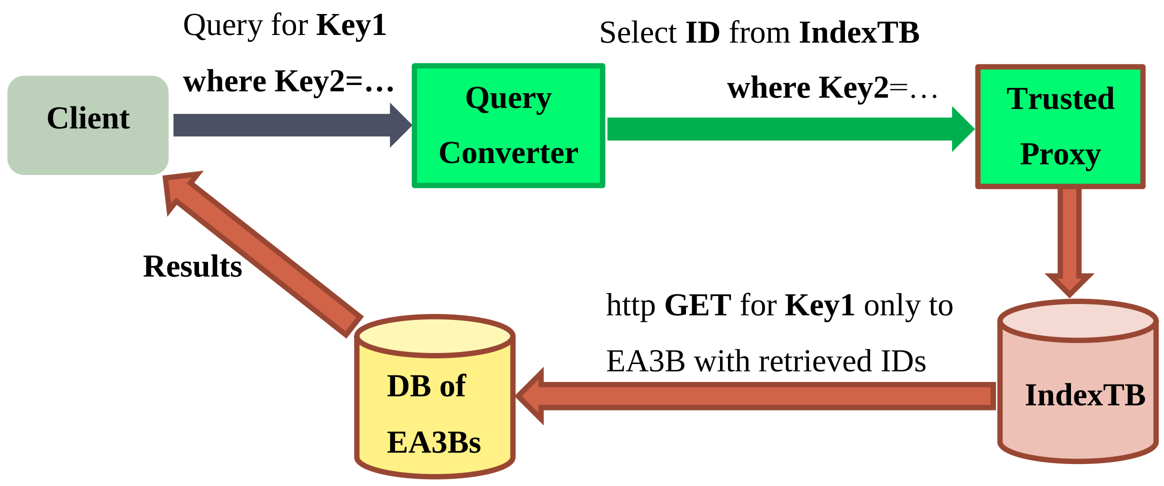


Figure 4-3. Query Workflow for EA3B Database

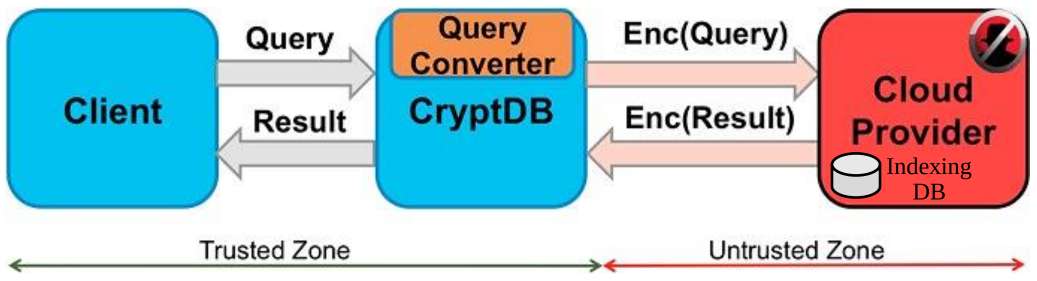


Figure 4-4. Database Encryption using CryptDB. ©2018 IEEE

Fig. 4-4 illustrates a CryptDB-based approach. When a user submits a SQL query, a proxy at the client side intercepts a query and converts it into a semantically equivalent query that operates over encrypted data. The converted query is then forwarded to the encrypted database, which might be located in untrusted environment. The functions shown in Table 4-2 are implemented as User-defined functions (UDFs). CryptDB supports a set of SQL queries, operating on encrypted data. It never releases decryption key to a database and provides data protection in transit and at rest. When compromised, only ciphertext is revealed and data leakage is limited to data for users who are currently logged in. CryptDB provides protection against malicious or curious database administrators. Query results are only decrypted at trusted proxy server and then sent to client.

Partially Homomorphic Encryption (PHE) schemes are used to encrypt data items in an indexing database. This allows executing queries over encrypted data according to the homomorphic operations that these PHE schemes provide. If $E(x)$ and $D(x)$ denote the encryption and decryption functions for input data x respectively, an encryption scheme is said to be partially homomorphic with respect to operation $\odot$ iff $\exists\otimes\ s.t. D(E(x1)\otimes E(x2)) = x1 \odot x2$. For instance, D (E(5) * E(6)) = 30. Table 4-2 shows the encryption schemes along with the supported operations of each scheme and an example query.

Table 4-2. Encryption Schemes and Supported Operations. ©2018 IEEE

Encryption Scheme	**Homomorphic Property**	**Supported Operations**	**Example**
Paillier	AHE	+, SUM	Count sum of tolls paid by vehicles on a highway
ElGamal	MHE	*	Count covered distance which is multiplication: time * average speed
Boldyreva et al.	OPE	<, >, MIN, MAX	select ID, Speed, Model from IndexTB where Speed between 80 and 100
SWP	SRCH	Tokenized search	select Model from IndexTB where issue LIKE %battery%
AES	DET	Exact search	select ID from IndexTB where Model = 'Ferrari'

Naveed et al. published a paper[1] in 2015 [19] where they demonstrated successful attacks to recover the plaintext from database columns, encrypted using Order Preserving Encryption (OPE) and Deterministic (DET) encryption, used in CryptDB. However, Raluca Popa in [27] explained the guidelines for using the CryptDB system securely. DET scheme provides strong encryption guarantees if there are no data repetitions in table columns encrypted with DET and every value is unique. OPE should not be used for columns with sensitive data, but still can be used for non-sensitive data columns with high entropy values from a sparse domain, e.g., timestamps. In my use case, OPE is used for vehicle's speed. OPE is known to be prone to frequency analysis attack which may allow to recover plaintext [19]. To mitigate this, fully homomorphic encryption (FHE) can be used, but it imposes prohibitive overhead.

Overall, my framework is agnostic to the encrypted database engine.

[1] Naveed, Muhammad, Seny Kamara, and Charles V. Wright. "Inference attacks on property-preserving encrypted databases," In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 644-655. ACM, 2015.

4.2 Fast Analytics over Encrypted Data

My EA3B-based approach provides data protection in transit and at rest. EA3B stores data in encrypted form. It is highly desirable to have capabilities of building analytics over encrypted data, stored in the form of EA3Bs.

My solution for this problem is to add a "Summary" field to an EA3B. The value of "Summary" key contains the abstract of an EA3B and is used to build analytics over a database of EA3Bs. To enable fast analytics building, "Summary" field of an EA3B can be accessed bypassing authentication, policy and attribute evaluation phases, which occur for regular EA3B data request workflow. To bypass these phases, "Summary" is encrypted using Attribute-Based Encryption (ABE), when an EA3B is generated. Encryption key is derived using MMCP technique. Key-Policy Attribute-Based Encryption (KP-ABE) is used to derive encryption key from Make, Model, Color and license Plate (MMCP) attributes [24]. The service who has an MMCP-key, for instance ITS administrator or Law Enforcement representative, can decrypt "Summary" without going through policy evaluation. "Summary" field contains vehicle ID, license plate number, vehicle's health and performance data, including average speed, trip duration, fuel consumption, engine temperature, etc. Other data, necessary for ITS or Law Enforcement, might be included in the "Summary" as well. More sensitive data such as driver's license number of vehicle's owner or home address are not included in the "Summary" and requesting services are required to go through full policy evaluation check to access these sensitive data, as described in 2.1.1.

The use case is illustrated on Fig. 4-5. V2X communication objects are represented as web services, e.g., Driver, Law Enforcement, Insurance Service, Intelligent Transportation System, Corporate Car Service, etc. At step 1, speed camera captures the vehicle speed and license plate and sends them together with the speed limit at step 2 to the cloud provider. Speed camera, in contrast with the toll gate high-resolution camera, might not be able to detect vehicle's make and model. At step 3, once vehicle reaches the toll gate, the high-resolution camera captures vehicles make, model, color, license plate number and sends these attributes at step 4 to the cloud provider, where they are used to derive a unique encryption MMCP key at step 5. At step 6, this encryption key is sent to the vehicle, along with previously captured at steps 1, 2 pairs of (speed, speed limit). At step 7, all the vehicles attributes are bonded together and Vehicle Record (VR) is created, having them in encrypted form [6]. VR is created locally at the vehicle, not at the cloud provider, to guarantee protection against malicious or curious cloud administrators. VR is used for secure decentralized inter-vehicular data communications (IVC) in wireless V2X networks.

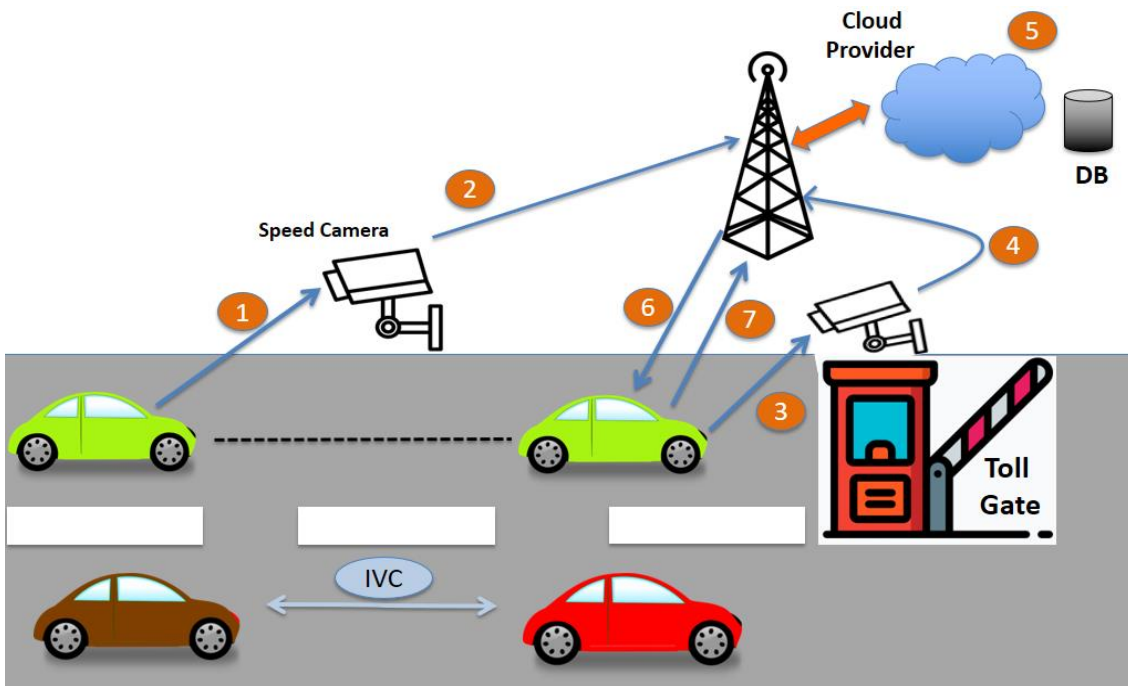


Figure 4-5. V2X Communication Network with MMCP Key Derivation. ©2018 ICAEEE

*Clipart taken from: http://www.clker.com/clipart-soft-green-car-1.html, http://www.clker.com/clipart-red-car-11.html, http://www.clker.com/clipart-brown-car.html (car icons shared by Nur Aisyah),
Video camera icon made by Kiranshastry from www.flaticon.com, https://www.flaticon.com/free-icon/camera_1613985; https://www.clker.com/clipart-basestation.html,
Toll road (toll gate) icon made by Smashicons from www.flaticon.com, https://www.flaticon.com/free-icon/toll-road_829376

Authorized vehicles with MMCP key, e.g., insurance service car, can query nearby vehicles, insured in a given company, for summary statistics and build local analytics, e.g., on vehicles performance and health. Health report can be used to help insured cars on a highway. ITS, which is supposed to have an MMCP key, can build statistics on traffic jams by retrieving an average speed from the VR's "Summary" field. Capability of on-the-fly fast data analytics can be useful for ITS in order to provide better road assistance for the drivers.

4.3 Related Work

In 2013 Microsoft came up with a CipherBase [26] product, which is a SQL database system for storing and processing strongly encrypted (i.e., using FHE) data. Cipherbase is based on a combination of customized trusted hardware and Microsoft SQL Server. It simulates FHE on top of non-homomorphic encryption schemes (e.g., AES in CBC-mode) by integrating trusted hardware. Application logic is decoupled from encryption. FHE can lead to slowdowns in the order of 10^9 times [49]. CryptDB [9], [27] is a seminal work that demonstrates how PHE can be used to enable the execution of secure SQL queries over encrypted data with practical performance. Crypsis [50] and Cuttlefish [51] demonstrate that PHE can be used in cloud environments with multiple computing nodes to perform data analytics on large datasets in a distributed manner. My solution, in addition to support of search through encrypted data and data protection in transit and at rest, provides extended attribute-based access control, data leakage detection capabilities and supports decentralized service architectures.

4.4 Evaluation

4.4.1 Experiment 1. Round-trip Times for Data Requests to VRs using Plaintext and Encrypted Indexing Databases

In this experiment, I deploy two databases, a regular plaintext one where no encryption is used and therefore all sensitive information might get leaked, and one that employs homomorphic encryption to keep data confidential while allowing search through encrypted data. Data request processing has two phases:

(1) filter relevant VRs from the plaintext/encrypted databases;

(2) extract accessible requested data items from relevant VRs, stored as EA3Bs. It takes 71.46 [msec], based on experiment 2.3.5 and Fig. 2-18.

Hardware: Intel 1.9GHz CPU, 1GB RAM

OS: Linux Ubuntu 12.04.5 LTS (kernel 3.13.0-32-generic, 64-bit)

Plaintext DBMS: MySQL v. 5.6

Encrypted DBMS: CryptDB, ver. March 2014

The database contains a single table named IndexTB and it was populated with 1000 tuples. Fragment of IndexTB table is given in Table 4-1. In this experiment, the client issues the following five SQL queries:

Equality query (Q1):

```
SELECT ID FROM IndexDB WHERE model = "Ford"
```

Inequality query (Q2):

```
SELECT ID, speed, model FROM IndexDB WHERE speed > 80
```

Inequality query, shortened (Q3):

```
SELECT ID FROM IndexDB WHERE speed > 80
```

Range query (Q4):

```
SELECT ID, speed, model FROM IndexDB WHERE speed BETWEEN 71 AND 80
```

Range query, shortened (Q5):

```
SELECT ID FROM IndexDB WHERE speed BETWEEN 71 AND 80
```

Q1 retrieves all records that match an exact value for the model of the vehicle and therefore requires an exact match. Q2 retrieves all records with speed greater than a certain value and therefore it requires order comparisons. Q3 is the same as Q2 except that the query returns only the ID of the records. In the case the query is executed in an encrypted database, this query demonstrates the time spent in decrypting the results, in comparison to Q2. Similarly, Q4 and Q5 filter records whose speeds are within a certain range of values. I measure query execution time for a plaintext database and compare it with the query execution time when data are encrypted, using CryptDB. I encrypted the "Model" column using a DET scheme and the "Speed" column using an OPE scheme.

Table 4-3. SQL Query Execution Times for Plaintext and Encrypted Databases. ©2018 IEEE

	SQL Query Execution Time (ms)	
	Plaintext Database	**Encrypted Database**
Q1	1.91	50
Q2	3.22	360
Q3	2.76	150
Q4	4.90	770
Q5	4.58	220

The results of SQL queries execution times are shown in Table 4-3. All reported times are the average of 50 executions. Query execution time grows 26 times for Q1 over encrypted database. This time includes the decryption time for the results. For Q2, the overhead is 112 times when all three columns need to be decrypted or 54 times when only the ID needs to be decrypted as in Q3. For Q4, the overhead 157 times and for Q5 the overhead is 48 times.

Fig. 4-6 contains overall data request round-trip times, which are the sums of times for phase 1 and phase 2. Precisely speaking, overall data request round-trip time is the sum of SQL query execution time (phase 1) and time to process data request by VR kernel, which is 71.46 [msec].

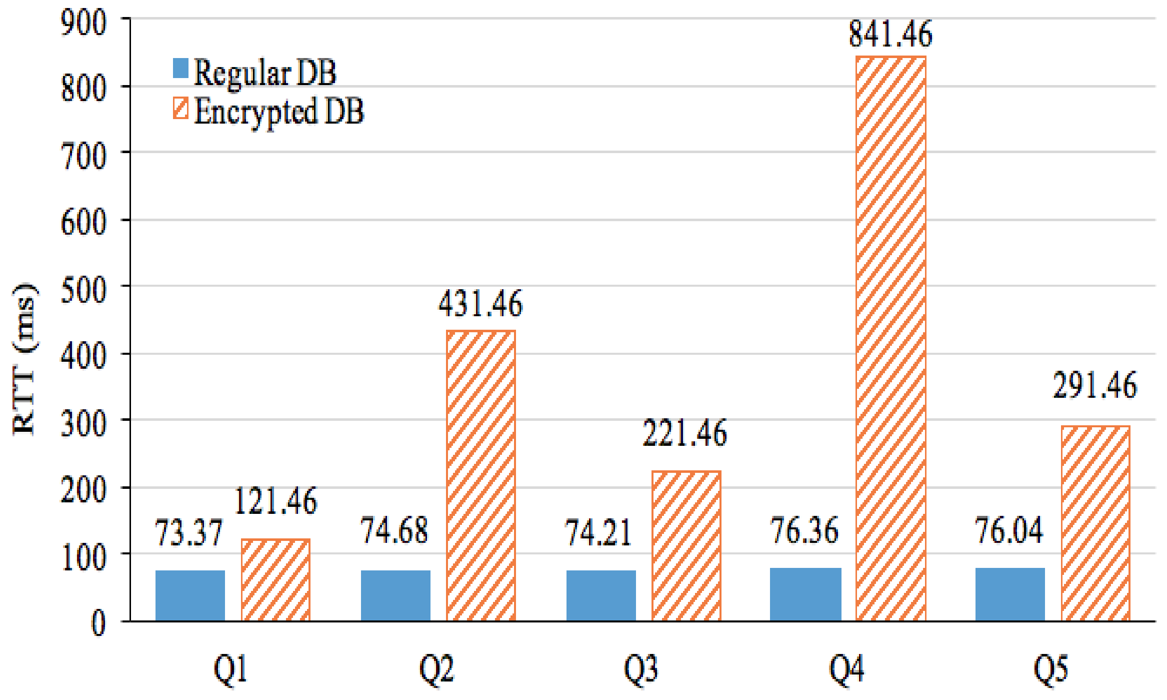


Figure 4-6. Data Request Round-trip Times for Plaintext and Encrypted Indexing Databases. ©2018 IEEE

5. BLOCKCHAIN-BASED DATA EXCHANGE

Portions of this chapter have been reprinted with permission from the following:

Ulybyshev, Denis, Miguel Villarreal-Vasquez, Bharat Bhargava, Ganapathy Mani, Steve Seaberg, Paul Conoval, Robert Pike, and Jason Kobes. "'Blockhub': Blockchain-based Software Development System for Untrusted Environments." In *IEEE International Conference on Cloud Computing (CLOUD)*. 2018. pp. 582-585.
©2018 IEEE
https://doi.org/10.1109/CLOUD.2018.00081

5.1 Blockchain for Data Provenance Integrity in SOA

It is essential to collect provenance data about data transactions in SOA. Provenance data is necessary to troubleshoot issues, such as communication failures, for system's audit and for investigating data leakages. In WAXEDPRUNE [3], [5] framework, provenance data is recorded locally and at a CM each time when a data request is served by an EA3B, which can be hosted by a recipient service, central server or a cloud provider. Provenance records contain information on what service is trying to access what class of data subset from an EA3B, when, and the origin of EA3B. As discussed in section 3.2.4, there are two problems with recording provenance data in WAXEDPRUNE:

a) For provenance data stored locally at a client side, there is no integrity guarantee since a malicious client can corrupt provenance record. This is resolved by applying blockchain-based technology, discussed below.
b) Storing provenance records on CM imposes significant overhead and makes the system centralized with a single point of failure.

In WAXEDPRUNE implementation with a trusted CM (see Fig. 3-3) for leakage detection there is no need to apply blockchain-based technology since the CM, as a trusted intermediary, can store provenance records and guarantee their integrity [58]. However, one of the main selling points of the AB/ EA3B methodology is that it ensures data protection in transit and at rest in decentralized peer-to-peer networks. Applying blockchain-based technology in decentralized peer-to-peer network without trusted intermediary is a good fit since there are multiple untrusted writers (see Fig. 5-1). If there is trusted intermediary and the writers are trusted in SOA, applying blockchain-based technology becomes pointless [58].

Compared to regular centralized databases, blockchain provides more trust, robustness and fault tolerance due to its immutability [52]. Instead of relying on a central administrator, the transactions can be verified by the blockchain network nodes that can access the digital public ledger and provide consensus-based transaction validation.

Blockchain technologies can be categorized in two groups based on their architecture and controls applied to the participants. Permissionless blockchains are the networks in which any node can act as a verifier of the network without previous authorization (e.g., Bitcoin, Ethereum). Permissioned blockchains are networks that require authorization from a centralized authority or consortium, which imposes identity management and role-based access control (e.g., Hyperledger). Blockchain technologies can also be categrozied as public or private blockchains depending on whether access control is applied in the network.

The core idea of my "Blockhub" project is to integrate EA3B, which provides role-based and attribute-based access control and data protection in transit and at rest, with blockchain-based network. "Blockhub" utilizes WAXEDPRUNE functionality for RESTful API, used for data communications between web services in NodeJS framework. The strong point of integrating WAXEDPRUNE with the blockchain-based platform, IBM Hyperledger Fabric [56], in Blockhub framework is that the integrated solution inherits advantages of both WAXEDPRUNE and blockchain-based technology. WAXEDPRUNE itself, if used separately, does not provide integrity of provenance data and does not allow verifying any data transaction at any time in the future. Integration with WAXEDPRUNE enriches blockchain-based platform, in my case IBM Hyperledger Fabric, with the following features:

1) Capability of detecting several types of data leakages that can be made by malicious insiders behind the scene outside of blockchain network.
2) Attribute-based access control, supporting wide set of attributes, such as:
    a. security level of service's web browser;
    b. trust level of service, which is constantly re-calculated;
    c. authentication method;

d. type of the device, service is running on;

e. context (e.g., normal vs. emergency).

As mentioned in section 1.2, I selected IBM Hyperledger Fabric as a blockchain platform for two reasons:

(1) It is open-source;

(2) It is a mature enterprise-level solution.

As illustrated on Fig. 5-1, every data access, transfer or update in WAXEDPRUNE is registered in a blockchain public ledger through IBM Hyperledger Fabric network. It guarantees that every data transaction is non-repudiatable by its invoker and can be verified at any time in the future. Blockchain technology allows to track and control data accesses, transfers and updates across multiple security domains.

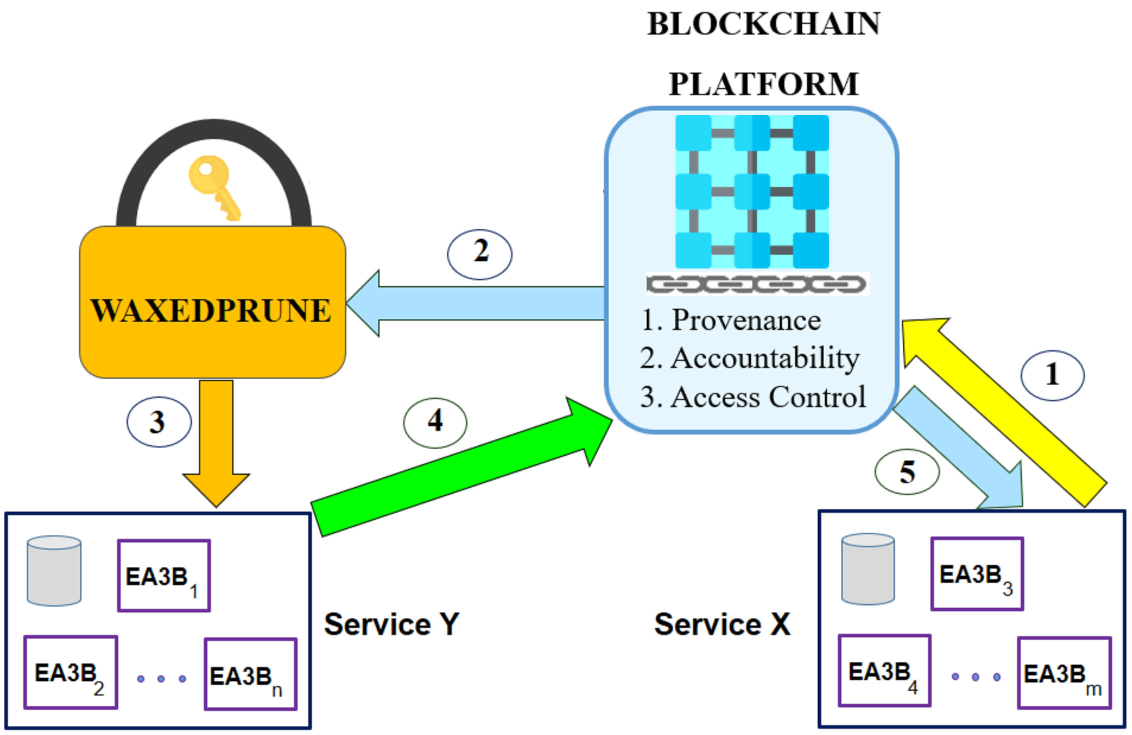


Figure 5-1. Blockhub: Blockchain-based Framework for SOA.

* Blockchain icon made by Freepik from www.flaticon.com, https://www.flaticon.com/free-icon/blockchain_1715679

Two untrusted services, e.g., X and Y, can securely exchange data in a controlled manner via smart contracts running in the blockchain network that regulate data access. Every access to the data subset is granted or rejected based on the policies established in these smart contracts and every successful request and transfer of data is logged as a record in the blockchain distributed ledger. Data are stored in encrypted form as EA3Bs. Thus, in addition to smart contracts, running in the blockchain network, there are access control policies, embedded into EA3Bs that control data access. Such a redundancy is useful for some blockchain platform such as Ethereum [53] which does not allow to update a smart contract code once it is deployed. In contrast, EA3B can be re-generated with the new access control policies embedded into it, when necessary.

For each data request, the following steps are executed:

Step 1. A service requests a particular data item, e.g., d_1, in the form of http get request. This request is registered in IBM Hyperledger Fabric network and recorded in a blockchain ledger. Smart contracts are enforced and if they allow service X to access d_1, the process continues at step 2.

Step 2. Data request is forwarded to a service X daemon, running in WAXEDPRUNE framework, where data leakage check is made and trust level of service X is checked (see Fig. 3-3). If data leakage alert is not raised and trust level of X is sufficient, then the process continues at step 3.

Step 3. WAXEDPRUNE forwards the data request to the EA3B, which stores the requested data subset d_1. Let us say, it is EA3B$_1$. In our case, data request is forwarded to service Y, which can be hosted either on the same site as WAXEDPRUNE or not. EA3B$_1$'s kernel evaluates role and attributes of X and enforces access control policies. Based on this evaluation, accessible data subset d_1 is retrieved and decrypted from EA3B$_1$.

Step 4. http response with decrypted d_1 is sent to service X through blockchain network, where the response is registered in a public ledger.

Step 5. http response with decrypted d_1 is sent to the requesting service X. In turn, service X's site may host other EA3Bs, since this use case deals with decentralized network architecture and peer-to-peer data communications.

5.2 Related Work

Permissioned blockchain solutions, such as IBM Hyperledger Fabric, are being created to meet the needs of companies to restrict the access to only approved entities. Financial industries have created digital multiple ledgers for different purposes [60].

Recent security breaches [55] show that blockchain technology has several security vulnerabilities. Thus, in some cases, it is not necessary to use blockchain-based solutions [58] if the requirements can be met by regular industry solutions, such as databases, running on trusted sites.

Microsoft has announced that in collaboration with Blockstack Labs, ConsenSys and developers from all over the world, it is working on an “open source, self-sovereign, blockchain-based identity system that allows people, products, apps and services to interoperate across blockchains, cloud providers and organizations” [61].

Recent study [63] demonstrates that trust and immutability can be provided using lightweight blockchain technology in wireless sensor networks.

5.3 Challenges of Blockchain-based Technologies

Whether a blockchain is permissionless or permissioned, there are several key technical challenges that limit the powerful potential of a blockchain technology.

5.3.1 Performance

Blockchain is replicated to all the network participants and this imposes a high performance overhead. The performance prevents blockchain-based technologies from being widely used in the systems where many transactions need to be processed per second.

5.3.2 Access Control

“Ethereum” blockchain platform [53] does not allow to update smart contract code once it is deployed. Smart contract code is permanent [54]. Integration with WAXEDPRUNE and using an EA3B for role-based and attributes access control would address this problem since role-based access control can be moved from smart contracts to an EA3B, where policies are specified and enforced. EA3B can be re-generated with the new access control policies embedded into it, when necessary.

5.3.3 dDOS Attacks

The Krypton blockchain, Ethereum-based blockchains, coded in golang, was hacked by “51 crew” [55]. They were able to replicate and manufacture their own blockchain, and push that into production as a real Krypton chain with large amounts of hashing power and DDoS attack on Krypton chain nodes.

5.4 Evaluation

5.4.1 Experiment 1. Data Request Round-trip Time in Blockhub Framework

I evaluated the performance overhead for a data request to an EA3B which, in “Blockhub”, goes through the IBM Hyperledger Fabric 1.0 platform first [56] and then is forwarded to “WAXEDPRUNE” framework [3], [5]. In this experiment, there are three web services in IBM Hyperledger network, running on the same host. RTT measurements are made for data requests sent to a local EA3B, i.e., stored on the same host as three web services in IBM Hyperledger Network, and for data requests sent to an EA3B hosted by a remote Google Cloud instance. Three web services are deployed on top of IBM Hyperledger Fabric network in NodeJS framework, provided by “Marbles” open-source project [57]. Service 1 sends a request for ID (16 bytes of data) to an EA3B, hosted by Service 2 on the same network node. Firstly, this data request is registered in the blockchain network and smart contracts are triggered. Then the data request is redirected to the corresponding EA3B. The overall transaction latency includes a blockchain transaction processing time and a Round-trip Time (RTT) for a data request, processed by an EA3B. Data request RTT for an EA3B is measured between the moments when data request processing is finished by the blockchain network and the moment when data, retrieved from an EA3B, are received by the client (Service 1). This includes authentication phase, when Service 1 authenticates itself to an EA3B, and evaluation phase, when access control policies and attributes of Service 1 are evaluated by an EA3B’s kernel. In addition, I measure the chaincode validation time, which is part of the blockchain transaction processing time.

I ran the data request for EA3B’s value of “ID” key 50 times and computed average values of chaincode validation time, blockchain transaction latency and RTT for EA3B.

Hardware, which hosts web services and IBM Hyperledger Fabric Network: Intel Core i7, CPU 860 @2.8GHz x8, 8GB DRAM

OS: Linux Ubuntu 14.04.5, kernel 3.13.0-107-generic, 64 bit

Hardware, which hosts EA3Bs on a Google cloud instance for remote data requests: Intel(R) Xeon(R) CPU@ 2.30GHz

OS: Linux Debian 4.9.65-3+deb9u2 (2018-01-04) x86_64, kernel 4.9.0-5-amd64, 64 bit

Blockchain platform: IBM Hyperledger Fabric ver. 1.0.x

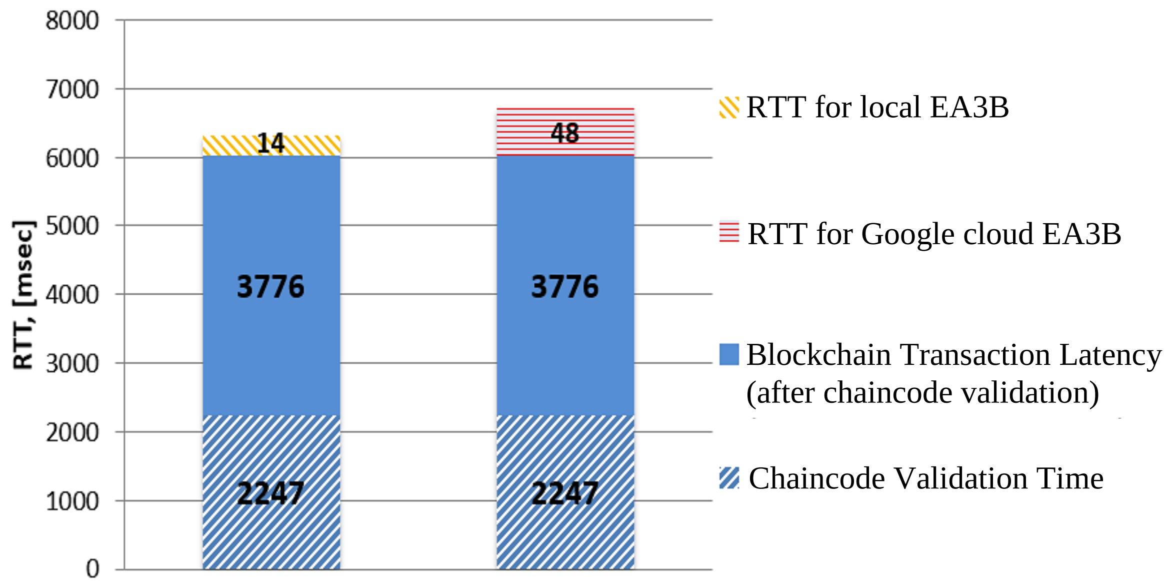


Figure 5-2. Transaction Latencies for Local and Google Cloud-based EA3Bs in Blockhub Framework. ©2018 IEEE

As it can be seen from Fig. 5-2, blockchain transaction latency is way greater than an EA3B RTT. Overall blockchain transaction latency, including chaincode validation time, is 6.02 [sec]. Benefits, provided by an EA3B, such as attribute-based access control and leakage detection, impose only 0.23% overhead when EA3B is hosted on the same localhost with the requesting service in the blockchain network, and 0.8% overhead when EA3B is hosted by a remote Google cloud instance. Hyperledger Fabric with versions newer than version 1.0.x, provides a way better performance [76], compared to version 1.0, but by the time when I was running performance experiments for a paper [30], IBM Hyperledger Fabric 1.1.x was not released yet. Microsoft Confidential Consortium (COCO) blockchain framework [21], [64] provides transaction latency 0.152 sec [28], but by the time when I was running performance experiments, it was not opensourced. Blockhub is agnostic to a blockchain-based platform, if the platform supports RESTful API and, preferably, NodeJS on top of blockchain network.

6. FUTURE WORK

6.1 Blockchain-based Software Development Framework

EA3B can carry different types of data, including software source code and binary executables. This idea leads to a concept of a Software Bundle (SB) (see Fig. 6-1), which is an EA3B with software modules (SMs) and binary executables, stored in encrypted form. SB concept is used to provide role-based and attribute-based access control with software spillage detection capabilities in a collaborative software development network across multiple security domains.

Similar process, illustrated on Fig. 5-1, applies for a secure exchange of software modules in a software development environment with multiple untrusted writers from multiple security domains. Every access, transfer or update of a software module is recorded in a blockchain network in a ledger, as illustrated on Fig. 6-2, and can be verified at any time in the future. Software access, updates and transfers cannot be repudiated by their invokers. This secures a collaborative software development, sharing and audit. Blockchain allows tracking and controlling what software components are shared between what entities across multiple security domains. Smart contracts provide access auditing, tracking capabilities and software leakage prevention. Leakage detection mechanism, inherited by an SB from an EA3B, enables detection of software spillages that can be made inadvertently or intentionally by malicious insiders.

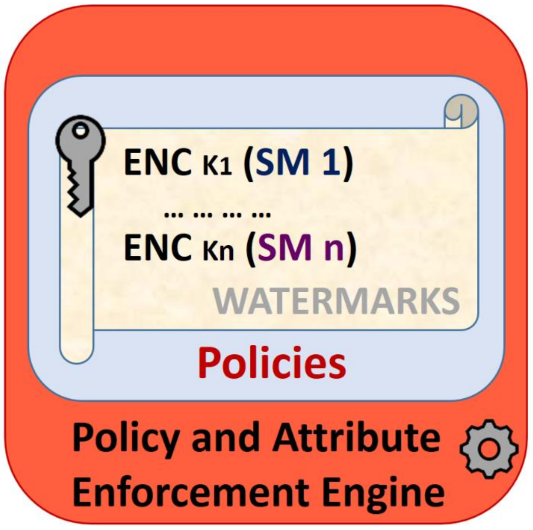


Figure 6-1. Software Bundle. ©2018 IEEE

As a future work, I plan to integrate this mechanism with one of the existing software repositories, such as GitHub or Bitbucket, or with one of the local internal software repositories. This will allow to record provenance data before software code is committed to the repository. Integrity of provenance records on software downloads and updates will be protected by a blockchain-based technology from malicious repository administrators and from unexpected corruptions.

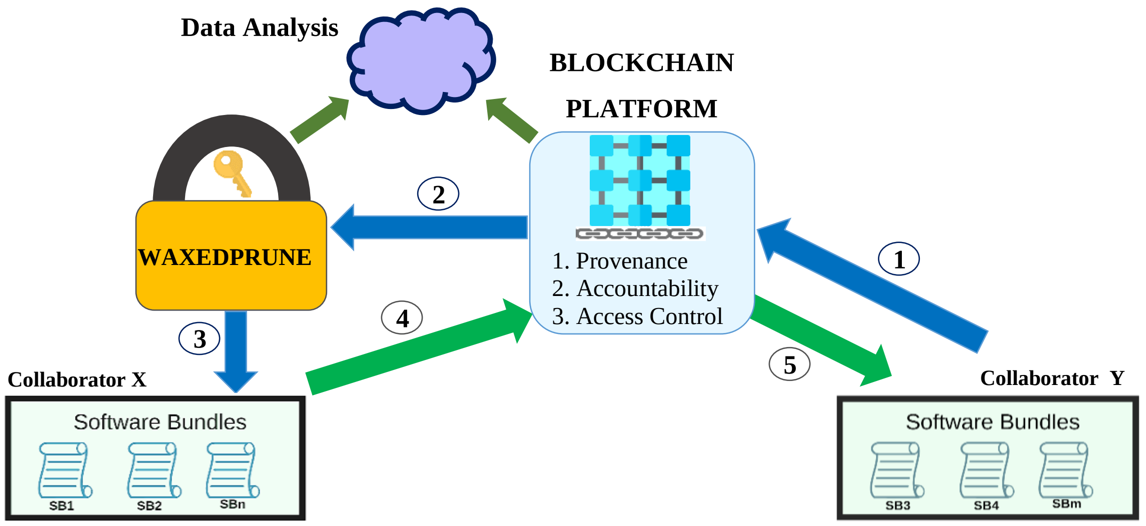


Figure 6-2. Blockchain-based Software Development Framework

* Blockchain icon made by Freepik from www.flaticon.com, https://www.flaticon.com/free-icon/blockchain_1715679

6.2 WAXEDPRUNE on Serverless Architecture

I plan to port WAXEDPRUNE framework on a serverless architecture instead of using NodeJS with servers running 24/ 7/ 365 [62].

"Serverless computing (or serverless for short), is an execution model where the cloud provider (AWS, Azure, or Google Cloud) is responsible for executing a piece of code by dynamically allocating the resources. And only charging for the amount of resources used to run the code" [59].

The Javascript code, used in WAXEDPRUNE, can be run inside stateless containers that can be triggered by HTTP GET/POST requests, trust level alerts, etc. Known cloud providers support serverless computing as follows:

a) Google Cloud: Cloud Functions;
b) Amazon AWS: AWS Lambda;
c) Microsoft Azure: Azure Functions.

6.3 Policy Aggregation Mechanism for EA3B

Currently, authorized services can update data in an EA3B on-the-fly. In the future, I plan to add to EA3B a support of access control policies updates that can be made by authorized services on-the-fly. The main problem with that is a policy aggregation algorithm: new policies added to an EA3B might contradict with the existing policies. Performance can also be an issue since an EA3B is a jar-file and generating it from scratch takes up to 10 seconds.

6.4 Improving Key Derivation Method for EA3B

Currently, symmetric AES key is derived per separate data subset, employing *SecretKeyFactory*, *PBEKeySpec* and *SecretKeySpec* methods of Java “crypto” library. Based on tests, recommended by NIST, these methods do not provide enough entropy in key generation. To make generated keys more secure and provide greater entropy, I plan to explore Key Derivation Functions (KDFs), which provide better key entropy.

6.5 Scalability Tests for EA3B

In my current evaluations for a Hospital Information System, implemented in a WAXEDPRUNE project, and for a V2X communication system, I used at most 4 web services and up to 50 data requests sent by one service to an EA3B. For future development of an EA3B technology and its potential industrial commercialization, it is necessary to perform extensive scalability tests with way more web services trying to access data from the same EA3B simultaneously.

REFERENCES

1. Ranchal, Rohit. "Cross-domain data dissemination and policy enforcement." (2015). PhD Thesis, Purdue University
2. Calzavara, Stefano, Riccardo Focardi, Niklas Grimm, and Matteo Maffei. "Micro-policies for web session security." In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pp. 179-193. IEEE, 2016.
3. Ulybyshev, Denis and Bharat Bhargava, "Secure dissemination of EHR," demo video 2017, [Online]. Available: https://www.dropbox.com/s/oxgy7xsovcrkel9/NGCRC-2017-WaxedPrune-Demo.wmv?dl=0 , accessed: Mar. 2019
4. Ranchal, Rohit, Denis Ulybyshev, Pelin Angin, and Bharat Bhargava, "Policy-based Distributed Data Dissemination," *CERIAS Security Symposium, April 2015 (poster)*
5. Ulybyshev, Denis, Bharat Bhargava, Miguel Villarreal-Vasquez, Aala Oqab Alsalem, Donald Steiner, Leon Li, Jason Kobes, Harry Halpin, and Rohit Ranchal. "Privacy-preserving data dissemination in untrusted cloud." In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 770-773. IEEE, 2017.
6. Ulybyshev, Denis, Aala Oqab Alsalem, Bharat Bhargava, Savvas Savvides, Ganapathy Mani, and Lotfi Ben Othmane. "Secure data communication in autonomous v2x systems." In *2018 IEEE International Congress on Internet of Things (ICIOT)*, pp. 156-163. IEEE, 2018.
7. Ulybyshev, Denis, Bharat Bhargava, and Aala Oqab-Alsalem. "Secure Data Exchange and Data Leakage Detection in an Untrusted Cloud." In *International Conference on Application of Computing and Communication Technologies*, pp. 99-113. Springer, Singapore, 2018.
8. Ulybyshev, Denis and Bharat Bhargava, "Secure dissemination of EHR," demo video 2016, [Online]. Available: https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0 , accessed: Mar. 2019
9. Popa, Raluca Ada, 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*, pp. 85-100. ACM, 2011.
10. "Lightweight data-interchange format JSON," [Online]. Available: http://json.org/ , accessed: Mar. 2019

11. Anonymus, "Micro-policies for web session security," 2016, [Online]. Available: https://sites.google.com/site/micropolwebsese, accessed: Mar. 2019

12. "W3C Web Cryptography API," [Online]. Available: https://www.w3.org/TR/WebCryptoAPI/, accessed: Mar. 2019

13. "Web authentication: an API for accessing scoped credentials," [Online]. Available: http://www.w3.org/TR/webauthn , accessed: Mar. 2019

14. "Finding or verifying credit card numbers," [Online]. Available: http://www.regularexpressions.info/creditcard.html , accessed: Mar.2019

15. "WSO2 Balana Implementation," [Online]. Available: https://github.com/wso2/balana , accessed: Mar. 2019

16. Pearson, Siani, and Marco Casassa-Mont. "Sticky policies: An approach for managing privacy across multiple parties." *Computer* 44, no. 9 (2011): 60-68.

17. Othmane, Lotfi Ben, and Leszek Lilien. "Protecting privacy of sensitive data dissemination using active bundles." In *2009 World Congress on Privacy, Security, Trust and the Management of e-Business*, pp. 202-213. IEEE, 2009.

18. Lilien, Leszek, and Bharat Bhargava. "A scheme for privacy-preserving data dissemination." *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 36, no. 3 (2006): 503-506.

19. Naveed, Muhammad, Seny Kamara, and Charles V. Wright. "Inference attacks on property-preserving encrypted databases." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 644-655. ACM, 2015.

20. Xu, Z. J., Z. Z. Wang, and Q. Lu. "Research on image watermarking algorithm based on DCT." *Procedia Environmental Sciences* 10 (2011): 1129-1135.

21. "The Confidential Consortium Blockchain Framework. Technical Overview," [Online]. Available:https://github.com/Azure/coco-framework/blob/master/docs/Coco%20Framework%20whitepaper.pdf, accessed: Mar. 2019

22. "Windows media DRM," [Online]. Available: https://en.wikipedia.org/wiki/Windows_Media_DRM, accessed: Mar. 2019

23. Li, Fengjun, Bo Luo, Peng Liu, Dongwon Lee, and Chao-Hsien Chu. "Enforcing secure and privacy-preserving information brokering in distributed information sharing." *IEEE transactions on information forensics and security* 8, no. 6 (2013): 888-900.

24. Ulybyshev, Denis, Servio Palacios, Ganapathy Mani, Aala Oqab Alsalem, Bharat Bhargava, and Puneet Goyal. "On-the-fly Analytics over Encrypted Records in Untrusted V2X Environments." ICACEEE, 2018, Zurich.

25. Collberg, Christian, G. R. Myles, and Andrew Huntwork. "Sandmark – a tool for software protection research." *IEEE security & privacy* 99, no. 4 (2003): 40-49.

26. Arasu, Arvind, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, and Ramarathnam Venkatesan. "Orthogonal Security with Cipherbase." In *CIDR*. 2013.

27. Popa, Raluca Ada, Nickolai Zeldovich, and Hari Balakrishnan. "Guidelines for Using the CryptDB System Securely." *IACR Cryptology ePrint Archive* 2015 (2015): 979.

28. Russinovich, Mark, "Introducing the COCO framework," [Online]. Available: https://youtu.be/8s6JMmGJ-dY , accessed: Mar. 2019.

29. Boneh, Dan, and Matthew Franklin. "An efficient public key traitor tracing scheme." In *Annual International Cryptology Conference*, pp. 338-353. Springer, Berlin, Heidelberg, 1999.

30. Ulybyshev, Denis, Miguel Villarreal-Vasquez, Bharat Bhargava, Ganapathy Mani, Steve Seaberg, Paul Conoval, Robert Pike, and Jason Kobes. "'Blockhub': Blockchain-based Software Development System for Untrusted Environments." In *IEEE International Conference on Cloud Computing (CLOUD)*. 2018.

31. Qu, Chenyang, Denis A. Ulybyshev, Bharat K. Bhargava, Rohit Ranchal, and Leszek T. Lilien. "Secure dissemination of video data in vehicle-to-vehicle systems." In *2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW)*, pp. 47-51. IEEE, 2015.

32. Ranchal, Rohit, Bharat Bhargava, Lotfi Ben Othmane, Leszek Lilien, Anya Kim, Myong Kang, and Mark Linderman. "Protection of identity information in cloud computing without trusted third party." In *2010 29th IEEE symposium on reliable distributed systems*, pp. 368-372. IEEE, 2010.

33. Ranchal, Rohit, Bharat Bhargava, Pelin Angin, and Lotfi Ben Othmane. "Epics: A framework for enforcing security policies in composite web services." *IEEE Transactions on Services Computing* (2018).

34. Harding, John, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. *Vehicle-to-vehicle communications: readiness of V2V technology for application*. No. DOT HS 812 014. United States. National Highway Traffic Safety Administration, 2014.

35. "ETSI - TS 102 941 - intelligent transport systems (ITS); security; trust and privacy management engineering 360," 2018. [Online]. Available: http://standards.globalspec.com/ std/1530232/etsi-ts-102-941, accessed: Mar. 2019

36. Bhargava, Bharat, "Secure/resilient systems and data dissemination/provenance," NGCRC Project Technology Final Report, CERIAS, Purdue University, Sep.2017

37. Bhargava, Bharat, "Privacy-Preserving Data Dissemination and Adaptable Service Compositions in Trusted & Untrusted Cloud," NGCRC Project Technology Final Report, CERIAS, Purdue University, Sep.2016

38. Liu, Qiong, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. "Digital rights management for content distribution." In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21*, pp. 49-58. Australian Computer Society, Inc., 2003.

39. Sabadra, Priti, and Mark Stamp. "The MediaSnap® digital rights management system." In *Proceedings of Conference on Computer Science and its Applications, San-Diego, California*. 2003.

40. Nickolova, Maria, and Eugene Nickolov. "Hardware-based and software-based security in digital rights management solutions." (2008).

41. J. Nevase, P. Chougale, S. Shewale and P. Bhosale, "Data Leakage Detection," Imperial Journal of Interdisciplinary Research, 3(5), 2017

42. Stamati-Koromina, Veroniki, Christos Ilioudis, Richard Overill, Christos K. Georgiadis, and Demosthenes Stamatis. "Insider threats in corporate environments: a case study for data leakage prevention." In *Proceedings of the Fifth Balkan Conference in Informatics*, pp. 271-274. ACM, 2012.

43. Kaur, Kamaljeet, Ishu Gupta, and Ashutosh Kumar Singh. "Data leakage prevention: e-mail protection via gateway." In *Journal of Physics: Conference Series*, vol. 933, no. 1, p. 012013. IOP Publishing, 2018.

44. Gupta, Ishu, and Ashutosh Kumar Singh. "A probability based model for data leakage detection using bigraph." In *Proceedings of the 2017 the 7th International Conference on Communication and Network Security*, pp. 1-5. ACM, 2017.

45. "Transparent Data Encryption (TDE)," [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-2017, accessed: Mar. 2019

46. "Always Encrypted (Database Engine)," [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-2017, accessed: Mar. 2019

47. "Always Encrypted vs. Transparent Data Encryption," [Online]. Available: https://sqltutorialtips.blogspot.com/2017/11/always-encrypted-vs-transparent-data.html, accessed: Mar. 2019

48. "SQL Server 2019 Preview Release Notes," [Online]. Available: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-ver15-release-notes?view=sqlallproducts-allversions, accessed: Mar. 2019

49. Gentry, Craig, Shai Halevi, and Nigel P. Smart. "Homomorphic evaluation of the AES circuit." In *Annual Cryptology Conference*, pp. 850-867. Springer, Berlin, Heidelberg, 2012.

50. Stephen, Julian James, Savvas Savvides, Russell Seidel, and Patrick Eugster. "Practical Confidentiality Preserving Big Data Analysis." In *6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)*. 2014.

51. Savvides, Savvas, Julian James Stephen, Masoud Saeida Ardekani, Vinaitheerthan Sundaram, and Patrick Eugster. "Secure data types: a simple abstraction for confidentiality-preserving data analytics." In *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 479-492. ACM, 2017.

52. "Blockchains vs centralized databases," March 17, 2016, [Online]. Available: https://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/, accessed: Mar. 2019

53. "Etherium Blockchain App Platform," [Online]. Available: https://www.ethereum.org/, accessed: Mar. 2019

54. "How to update a deployed smart contract," [Online]. Available: https://ethereum.stackexchange.com/questions/11938/how-to-update-a-deployed-smart-contract, accessed: Mar. 2019

55. "Krypton Abandons Ethereum-Based Blockchain After 51% Attack," May 31, 2017. [Online]. Available: http://krypton.rocks/blog/krypton-abandons-ethereum-based-blockchain-after-51-attack/, accessed: Mar. 2019

56. "IBM Blockchain 101: Quick-start guide for developers," 2016. [Online]. Available: https://www.ibm.com/developerworks/cloud/library/cl-ibm-blockchain-101-quick-start-guide-for-developers-bluemix-trs/index.html , accessed: Mar. 2019

57. "Marbles Demo," [Online]. Available: https://github.com/IBM-Blockchain/marbles, accessed: Mar. 2019

58. Gideon Greenspan, "Avoiding the Pointless Blockchain Project," 11/22/2015, [Online]. Available: http://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/, accessed: Mar. 2019

59. "What is Serverless?" [Online]. Available: https://serverless-stack.com/chapters/what-is-serverless.html , accessed: Mar. 2019

60. "Is Blockchain the Next Great Hope – or Hype?" January 11, 2017, [Online]. Available: http://knowledge.wharton.upenn.edu/article/blockchain-next-great-hope-hype/. Accessed: Mar. 2019

61. "Microsoft building Open Blockchain-based Identity System With Blockstack, ConsenSys," [Online]. Available: https://bitcoinmagazine.com/articles/microsoft-building-open-blockchain-based-identity-system-with-blockstack-consensys-1464968713/, accessed: Mar. 2019

62. "Node.js APIs on AWS—the pros and cons of Express versus Serverless," [Online]. Available: https://medium.freecodecamp.org/node-js-apis-on-aws-the-pros-and-cons-of-express-versus-serverless-a370ab7eadd7, accessed: Mar. 2019

63. Gordon, Gilroy. "Provenance and authentication of oracle sensor data with block chain lightweight wireless network authentication scheme for constrained oracle sensors." (2017), Master's Thesis, Saint Mary's University, Halifax, Nova Scotia, Jan. 2017

64. Russinovich, Mark, "Announcing the Confidential Consortium Blockchain Framework for Enterprise Blockchain Networks," [Online]. Available: https://azure.microsoft.com/en-us/blog/announcing-microsoft-s-coco-framework-for-enterprise-blockchain-networks/ , accessed: Mar. 2019

65. "Semantic Security," [Online]. Available: https://en.wikipedia.org/wiki/Semantic_security , accessed: Mar. 2019

66. Goldwasser, Shafi, and Silvio Micali. "Probabilistic encryption." *Journal of computer and system sciences* 28, no. 2 (1984): 270-299.

67. Paul Beame, "Lecture 11: Semantic Security vs Indistinguishability Security," 08 Feb. 2006, [Online].Available: https://courses.cs.washington.edu/courses/cse599b/06wi/lecture11.pdf , accessed: Mar. 2019

68. "Block cipher mode of operation," [Online]. Available: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_Block_Chaining_(CBC) , accessed: Mar. 2019

69. "Chosen plaintext attack," [Online]. Available: https://en.wikipedia.org/wiki/Chosen-plaintext_attack , accessed: Mar. 2019

70. "Oblivious transfer," [Online]. Available: https://en.wikipedia.org/wiki/Oblivious_transfer , accessed: Mar. 2019

71. Even, Shimon, Oded Goldreich, and Abraham Lempel. "A randomized protocol for signing contracts." Communications of the ACM 28, no. 6 (1985): 637-647.

72. Katz, Jonathan, and Yehuda Lindell, "Introduction to modern cryptography." CRC press, 2014.

73. "OpenCV library," [Online]. Available: https://opencv.org/ , accessed: Mar. 2019

74. Stamp, Mark. "Digital rights management: the technology behind the hype." J. Electron. Commerce Res. 4, no. 3 (2003): 102-112.

75. Rogaway, Phillip. "Evaluation of some blockcipher modes of operation." Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan(2011).

76. “Answering your questions on Hyperledger Fabric performance and scale”, [Online]. Available: https://www.ibm.com/blogs/blockchain/2019/01/answering-your-questions-on-hyperledger-fabric-performance-and-scale/, accessed: Mar. 2019

VITA

EDUCATION

Ph.D. Computer Science, Purdue University, Cumulative/Major GPA: 3.62 / 3.78 May, 2019

M.S. Automatic Control Systems, Bauman Moscow State Technical University
(top-10 in Russia), Cumulative GPA: 3.93 out of 4 June, 2004

SUMMARY

Knowledgeable innovator in Information Security, Databases and Distributed Systems. Published twelve academic peer-reviewed publications in reputable conferences and journals, including seven first-author publications. 8 years of industrial experience in developing large-scale software for mass market, including firmware for printers, software for healthcare industry and industrial control systems. Has received four research and teaching awards. Two years of research in NSF project and four years of research for a corporate sector in the United States. Made significant contributions to writing five funded research proposals. Have an experience in writing NSF and NIH proposals. PhD Candidate in Computer Science at Purdue University, graduating in May, 2019.

ACADEMIC POSITIONS

1. Research Assistant (Department of Computer Science, Purdue University)

Project Title	Terms / Dates	Funded by	Project Description and Accomplishments
Situational Awareness and Targeted Information Propagation	Spring 2019	Bilsland Dissertation Fellowship from Purdue University	Aim to develop latent semantic indexing-based Twitter data mining engine. Twitter data is extracted using Twitter API for data related to Cambridge City, MA

Project Title	Terms / Dates	Funded by	Project Description and Accomplishments
Data Leakage Detection and Privacy-preserving Data Dissemination	Fall 2017 - Spring 2018	Corporate Partners of Purdue Univ. Computer Science Dept.: Northrop Grumman, Qualcomm, Intel, etc,…	Designed and developed “Blockhub” framework for privacy-preserving data communications. The approach provides secure cross-domain software distribution and development. Blockchain-based technology is applied to ensure integrity of provenance data and to record every software access, transfer and update in a blockchain public ledger. Project resulted in three published peer-reviewed conference papers [1], [6], [7]. Prototype demo video [17] is available.
Secure V2X (Vehicle-to-Everything) Systems	Spring 2017	Qatar National Research Fund (a member of Qatar Foundation)	Designed and developed a mechanism for secure Vehicle-to-Everything communications, enabling data protection at rest and in transit. Solution provides data confidentiality, integrity, role-based and attribute-based access control, as well as capabilities of building analytics over encrypted vehicle records. Experiments were conducted on a trade-off between vehicle’s safety and cybersecurity. Project resulted in published peer-reviewed papers in reputable conferences [2], [4].
Secure / Resilient Systems and Data Dissemination / Provenance	Spring 2017	Northrop Grumman Cybersecurity Research Consortium	Designed and implemented “WAXEDPRUNE” (Web-based Access to Encrypted Data Processing in Untrusted Environments) framework for data protection in transit and at rest, with capabilities of detecting several types of data leakages, made by insiders. Protection is provided by extended Active Bundles [5], as well as by digital/ visual watermarks, embedded into data and verified by a web crawler. Furthermore, support of search over encrypted data is enabled. Demo video [16] is available. Project resulted in a published peer-reviewed paper in a reputable Springer journal [3].
Privacy-Preserving Data Dissemination and Adaptable Service Compositions in Trusted & Untrusted Cloud	Spring 2016	Northrop Grumman Cybersecurity Research Consortium	Designed and implemented a framework for selective data dissemination based on roles and the following client's attributes: (a) security level of a web browser; (b) authentication method (password-based vs. hardware-based vs. fingerprint); (c) type of the network; (d) type of the device (mobile vs. desktop). Peer-reviewed paper [5] has been published in IEEE CLOUD 2017 conference. Prototype demo video [15] is available. The prototype was selected by Northrop Grumman to be demonstrated at their exhibition “Tech Expo 2016”

Project Title	Terms / Dates	Funded by	Project Description and Accomplishments
End-to-End Security Policy-Auditing and Enforcement in Untrusted Cloud	Spring 2015	Northrop Grumman Cybersecurity Research Consortium	Contributed to developing a privacy-preserving data communication framework that supports role-based and attribute-based access control in Service-Oriented Architecture. Attributes include trust level of services and context (e.g., emergency vs. regular context). The project won best poster award [18] at 16-th CERIAS Security Symposium (#1 out of 43 posters). Peer-reviewed paper [9] has been published.
Robust Distributed Wind Power Engineering	Spring 2013 – Fall 2014	NSF	Designed and implemented a robust crack detection algorithm for wind turbine blades, using vibro-accoustic analysis. Peer-reviewed paper [12] has been published. Source code [14] is available

2. Teaching Assistant (Department of Computer Science, Purdue University)

Course Title	Terms/Dates	Course Description
Information Systems / Relational Databases	Fall 2018, Fall 2016, Fall 2015	Relational Models, ER-diagrams, SQL/PLSQL; Dependencies and Normal Forms; Concurrency Control; NoSQL Databases; Database Security; Hadoop, Spark; Information Retrieval
Data Structures and Algorithms	Summer 2018	Basic data structures (array, linked list, stack, queue, heap, hash table, tree, trie, dictionary) and algorithms, using C/C++
Distributed Databases	Spring 2015	Concurrency Control Algorithms, Commitment Protocols (PAXOS), Privacy Preservation and Identity Management in Distributed Systems
Cryptography	Fall 2012	Symmetric Encryption (DES, AES); Asymmetric Encryption (Diffie-Hellman, RSA, Elliptic Curves); Digital Signatures; Hash Functions; HMAC; PKI (Public Key Infrastructure); Kerberos

RESEARCH INTERESTS

1. Data protection in transit and at rest, role-based and attribute-based access control, data leakage detection.
2. Cloud security.
3. Language-based security.
4. Database security: search through encrypted data, access control.
5. Distributed systems: blockchain-based technologies, concurrency control, commitment protocols.
6. Information Retrieval: web search, Search Engine Optimization.
7. Machine Learning: object detection, targeted information propagation, user profiling.
8. Vehicle-to-Everything Communication Systems.
9. Industrial Control Systems: IoT, SCADA systems, Programmable Logical Controllers.

INDUSTRIAL POSITIONS

Cybersecurity Software Engineer

Company: Coze Health, LLC **June, 2018 - Dec. 2018**

Accomplishments:

- Contributed to development of a secure HIPAA-compliant software product (MVP prototype) for video conferencing, message chat, fax and electronic surveys, using end-to-end encryption, two-factor authentication and firewalls
- Contributed to developing cloud-based solutions for storing and processing encrypted Electronic Medical Records, using Amazon EC2 cloud infrastructure

Software Engineer (Intern)

Company: Flexware Innovations **May, 2017 - Aug. 2017**

Accomplishments:

- Designed and developed a meeting room calendar management system (based on Microsoft Outlook and Google calendars), integrated into cloud-based Automation System with “Ignition” SCADA
- Developed Failure-Mode-Effect Analysis (FMEA) project for battery management system (for “A123 Systems” company)

Software Engineer

Company: Raduga-Krovlia LLC **July, 2009 - July, 2012**

Accomplishments:

- Designed and developed automatic control systems for rolling mills. More than 100 items were sold and are being used by customers
- Developed a corporate website, applied search-engine optimization techniques, which brought the web site to Google Top-10 for several relevant search queries
- Developed a context web-advertisement methodology

Embedded Software Engineer

Company: Samsung Electronics **Apr, 2007 - Feb. 2009**

Accomplishments:

- Designed and developed firmware (mass-storage component) for Multifunction Peripherals (MFPs) and printers, including hard disk drivers. Thousands of printers and MFP were sold all over the World
- Designed and developed an automated firmware testing tool (for mass-storage component)

Software Developer, Technical Marketing Engineer

Company: Schneider Electric **Sep, 2003 - Jan. 2007**

Accomplishments:

- Designed and developed Energy Management Control Systems for compressor plants, gas-turbine power stations, high-voltage substations in integration with Siemens, OMRON hardware
- Deployed 16 Industrial Control Systems on customer sites
- Designed and developed software for Building Management systems: CCTV, Heat-Ventilation-Air-Conditioning, access control

PUBLICATIONS (peer-reviewed)

1. Ulybyshev, Denis, Miguel Villarreal-Vasquez, Bharat Bhargava, Ganapathy Mani, Steve Seaberg, Paul Conoval, Robert Pike, and Jason Kobes. "'Blockhub': Blockchain-based Software Development System for Untrusted Environments." In *IEEE International Conference on Cloud Computing (CLOUD)*. 2018
2. Ulybyshev, Denis, Aala Oqab Alsalem, Bharat Bhargava, Savvas Savvides, Ganapathy Mani, and Lotfi Ben Othmane. "Secure data communication in autonomous v2x systems." In *2018 IEEE International Congress on Internet of Things (ICIOT)*, pp. 156-163. IEEE, 2018.
3. Ulybyshev, Denis, Bharat Bhargava, and Aala Oqab-Alsalem. "Secure Data Exchange and Data Leakage Detection in an Untrusted Cloud." In *International Conference on Application of Computing and Communication Technologies*, pp. 99-113. Springer, Singapore, 2018.
4. Ulybyshev, Denis, Servio Palacios, Ganapathy Mani, Aala Oqab Alsalem, Bharat Bhargava, and Puneet Goyal. "On-the-fly Analytics over Encrypted Records in Untrusted V2X Environments." ICACEEE, 2018, Zurich.
5. Ulybyshev, Denis, Bharat Bhargava, Miguel Villarreal-Vasquez, Aala Oqab Alsalem, Donald Steiner, Leon Li, Jason Kobes, Harry Halpin, and Rohit Ranchal. "Privacy-preserving data dissemination in untrusted cloud." In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 770-773. IEEE, 2017.
6. Mani, Ganapathy, Denis Ulybyshev, Bharat Bhargava, Jason Kobes, and Puneet Goyal. "Autonomous Aggregate Data Analytics in Untrusted Cloud." In *2018 IEEE First International*

Conference on Artificial Intelligence and Knowledge Engineering (AIKE), pp. 138-141. IEEE, 2018.

7. Mani, Ganapathy, Bharat Bhargava, Pelin Angin, Miguel Villarreal-Vasquez, Denis Ulybyshev, and Jason Kobes. "Machine Learning Models to Enhance the Science of Cognitive Autonomy." In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 46-53. IEEE, 2018.
8. Sardesai, Shantanu, Denis Ulybyshev, Lotfi ben Othmane, and Bharat Bhargava. "Impacts of Security Attacks on The Effectiveness of Collaborative Adaptive Cruise Control Mechanism." In *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1-5. IEEE, 2018.
9. Qu, Chenyang, Denis A. Ulybyshev, Bharat K. Bhargava, Rohit Ranchal, and Leszek T. Lilien. "Secure dissemination of video data in vehicle-to-vehicle systems." In *2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW)*, pp. 47-51. IEEE, 2015.
10. Ulybyshev, Denis. "Comparison of fuzzy and regular Least-Squares Methods in the random noise filtering task," Trans. of 5-th Intl. Symp. “Intelligent control systems 2002”. – Caluga (2002), ISBN 5 – 7038 – 2049 – 9, pp. 320-323
11. Ulybyshev, Denis. "Fuzzy Least-Squares Method and its modifications for different kinds of fuzzy “AND” operation in the random noise filtering task," Trans. of Intl. Symp. “Reliability and Quality”. – Penza (2003), ISBN 5 – 94170 – 031 – 8, pp. 203-207
12. Myrent, Noah J., Douglas E. Adams, Gustavo Rodriguez-Rivera, Denis A. Ulybyshev, Jan Vitek, Ethan Blanton, and Tomas Kalibera. "A robust algorithm to detecting wind turbine blade health using vibro-acoustic modulation and sideband spectral analysis." In *33rd Wind Energy Symposium*, p. 1001. 2015.

THESIS

13. Ulybyshev, Denis. "Energy Management Control System for High-Voltage Substations." M.S. Thesis, Bauman Moscow State Technical University, Department of Automatic Control Systems, 2004.

DEMO VIDEOS & CODE REPOSITORIES

14. Crack detection application for wind turbine blades
https://github.com/Denis-Ulybysh/CrackDetection_ComparingBlades, accessed: Mar. 2019

15. WAXEDPRUNE: privacy-preserving attribute-based data communications prototype demo video, https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0, accessed: Mar. 2019
16. WAXEDPRUNE: data leakage detection and encrypted search over encrypted data prototype demo video, https://www.dropbox.com/s/oxgy7xsovcrkel9/NGCRC-2017-WaxedPrune-Demo.wmv?dl=0 , accessed: Mar. 2019
17. Blockchain-based privacy-preserving data communication in Intelligent Autonomous Systems https://www.dropbox.com/s/x3l8w9l49am2cnw/Demo_NGCRC_Bhargava_Compiled.mp4?dl=0, accessed: Mar. 2019

PRESENTATIONS AT CONFERENCES, SYMPOSIA AND WORKSHOPS

18. Ranchal, Rohit, Denis Ulybyshev, Pelin Angin, and Bharat Bhargava. "PD3: policy-based distributed data dissemination." In *Proceedings of the 16th Annual Information Security Symposium*, p. 13. CERIAS-Purdue University, 2015. (best poster award, #1 out of 43)
19. "Secure / Resilient Systems and Data Dissemination / Provenance," NGC Research Consortium Symposium at Purdue University, Nov. 2017
20. Ulybyshev, Denis, Bharat Bhargava, Chenyang Qu, Rohit Ranchal, and Leszek Lilien, "Secure data dissemination in Vehicle-to-Vehicle Systems," 17th CERIAS Security Symposium, Apr. 2016
https://www.cerias.purdue.edu/assets/symposium/2016-posters/14B-A99.pdf, accessed: Mar. 2019
21. "Privacy-preserving Data Dissemination and Adaptable Service Compositions in Trusted and Untrusted Cloud," NGC Research Consortium Symposium, Apr. 2016

AWARDS AND FELLOWSHIPS

1. ***CERIAS Diamond Award for Outstanding Academic Achievement*** (#1 out of 50)
http://www.cerias.purdue.edu/site/people/student_awards/diamond **Apr. 2019**
2. ***2nd Best Poster Award at 20-th Annual Information Security Symposium*** **Apr. 2019**
(selected as #2 out of 44 by Corporate Partners of Computer Science Department, Purdue University). Poster: “Blockhub: Blockchain-based Data Communication System with Leakage Prevention and Detection”
3. ***Bilsland Dissertation Award Fellowship*** (research funds for Spring, 2019) **Aug. 2018**

4. ***Purdue Computer Science Corporate Partners Award*** (research funds for 2017-2018 Academic year). Pool of corporate partners, including Northrop Grumman, Qualcomm, Intel, Raytheon, Eli Lilly, ranked research proposal as #1 out of 21 **Apr. 2017**
5. ***Purdue Computer Science Harris Teaching Award*** for "Supporting Women in Computer Science" **Apr. 2017**
6. ***Best Poster Award*** at 16-th CERIAS Security Symposium **Mar. 2015**
   (selected as #1 out of 43 by Corporate Partners of Computer Science Department, Purdue University). Poster: "PD3: Policy-based Distributed Data Dissemination"
   Winner's certificate: https://www.cs.purdue.edu/homes/dulybysh/Images/CeriasCertificate-dulybysh.jpg

ACADEMIC ADVISOR

- Prof. Suresh Jagannathan **Aug.2012 – Dec. 2014**
- Prof. Bharat Bhargava **Jan.2015 – present**

LANGUAGES

- English (Good), Russian (Fluent), German (basic), Korean (basic)

PROFESSIONAL MEMBERSHIPS

- Member of Information Systems Security Association (ISSA), Indiana Chapter, **Jan.2018 – present**

UNIVERSITY SERVICE

- PhD Representative, Web-master in Computer Science Graduate Student Board at Purdue University **2012 – 2017**

LinkedIn: https://www.linkedin.com/in/denisulybyshev/

GitHub: https://github.com/Denis-Ulybysh

PUBLICATIONS

1. Ulybyshev, Denis, Miguel Villarreal-Vasquez, Bharat Bhargava, Ganapathy Mani, Steve Seaberg, Paul Conoval, Robert Pike, and Jason Kobes. "‘Blockhub’: Blockchain-based Software Development System for Untrusted Environments." In *IEEE International Conference on Cloud Computing (CLOUD)*. 2018
2. Ulybyshev, Denis, Aala Oqab Alsalem, Bharat Bhargava, Savvas Savvides, Ganapathy Mani, and Lotfi Ben Othmane. "Secure data communication in autonomous v2x systems." In *2018 IEEE International Congress on Internet of Things (ICIOT)*, pp. 156-163. IEEE, 2018.
3. Ulybyshev, Denis, Bharat Bhargava, and Aala Oqab-Alsalem. "Secure Data Exchange and Data Leakage Detection in an Untrusted Cloud." In *International Conference on Application of Computing and Communication Technologies*, pp. 99-113. Springer, Singapore, 2018.
4. Ulybyshev, Denis, Servio Palacios, Ganapathy Mani, Aala Oqab Alsalem, Bharat Bhargava, and Puneet Goyal. "On-the-fly Analytics over Encrypted Records in Untrusted V2X Environments." ICACEEE, 2018, Zurich.
5. Ulybyshev, Denis, Bharat Bhargava, Miguel Villarreal-Vasquez, Aala Oqab Alsalem, Donald Steiner, Leon Li, Jason Kobes, Harry Halpin, and Rohit Ranchal. "Privacy-preserving data dissemination in untrusted cloud." In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 770-773. IEEE, 2017.
6. Mani, Ganapathy, Denis Ulybyshev, Bharat Bhargava, Jason Kobes, and Puneet Goyal. "Autonomous Aggregate Data Analytics in Untrusted Cloud." In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 138-141. IEEE, 2018.
7. Mani, Ganapathy, Bharat Bhargava, Pelin Angin, Miguel Villarreal-Vasquez, Denis Ulybyshev, and Jason Kobes. "Machine Learning Models to Enhance the Science of Cognitive Autonomy." In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 46-53. IEEE, 2018.
8. Sardesai, Shantanu, Denis Ulybyshev, Lotfi ben Othmane, and Bharat Bhargava. "Impacts of Security Attacks on The Effectiveness of Collaborative Adaptive Cruise Control Mechanism." In *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1-5. IEEE, 2018.

9. Qu, Chenyang, Denis A. Ulybyshev, Bharat K. Bhargava, Rohit Ranchal, and Leszek T. Lilien. "Secure dissemination of video data in vehicle-to-vehicle systems." In *2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW),* pp. 47-51. IEEE, 2015.

10. Ulybyshev, Denis. "Comparison of fuzzy and regular Least-Squares Methods in the random noise filtering task," Trans. of 5-th Intl. Symp. "Intelligent control systems 2002". – Caluga (2002), ISBN 5 – 7038 – 2049 – 9, pp. 320-323

11. Ulybyshev, Denis. "Fuzzy Least-Squares Method and its modifications for different kinds of fuzzy "AND" operation in the random noise filtering task," Trans. of Intl. Symp. "Reliability and Quality". – Penza (2003), ISBN 5 – 94170 – 031 – 8, pp. 203-207

12. Myrent, Noah J., Douglas E. Adams, Gustavo Rodriguez-Rivera, Denis A. Ulybyshev, Jan Vitek, Ethan Blanton, and Tomas Kalibera. "A robust algorithm to detecting wind turbine blade health using vibro-acoustic modulation and sideband spectral analysis." In *33rd Wind Energy Symposium,* p. 1001. 2015.