Database Security

Presenter: Denis Ulybyshev

PhD Student, CS448 TA
Computer Science, Purdue University

Cybersecurity Software Engineer,
Coze Health LLC

Outline

1. Problem Statement
2. Solution
    2.1. Data Privacy
    2.2. Role-based Access Control
    2.3. Attribute-based Access Control
    2.4. Data Leakage Detection
    2.4. Encrypted Search over Encrypted Database Records

3. Conclusions

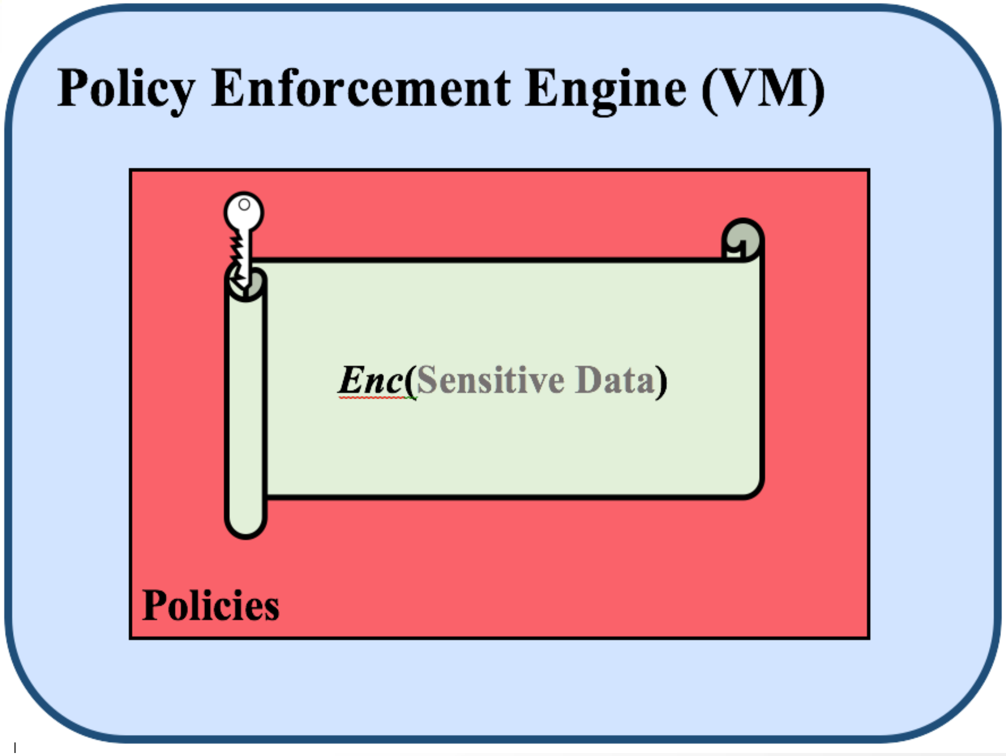
Problem Statement

- Provide secure storage and processing of database records
  - Confidentiality
  - Integrity
- Support role-based access control
- Support attribute-based access control
- Detect data leakages made by insiders to unauthorized parties
- Support encrypted search over encrypted database records

AB Core Design

Active Bundle (AB)
[17], [18], [1] is a self-protected structure that contains:

- *Sensitive data*:
  - Encrypted data items
  - Separate key per data subset
- *Metadata*: describe AB and its access control policies
  - Policies in JSON [10] or XACML [11] formats manage AB interaction with services and hosts



- *Policy Enforcement Engine* [15]: enforces policies specified in AB
  - Provides tamper-resistance of AB [1]

AB Example

Key-value pair stored in the Active Bundle:

*{ "ab.patientID" : "***Enc***(0123456789)" }*

*{ "ab.name" : "***Enc***('Monica Latte')" }*

Policy Examples:

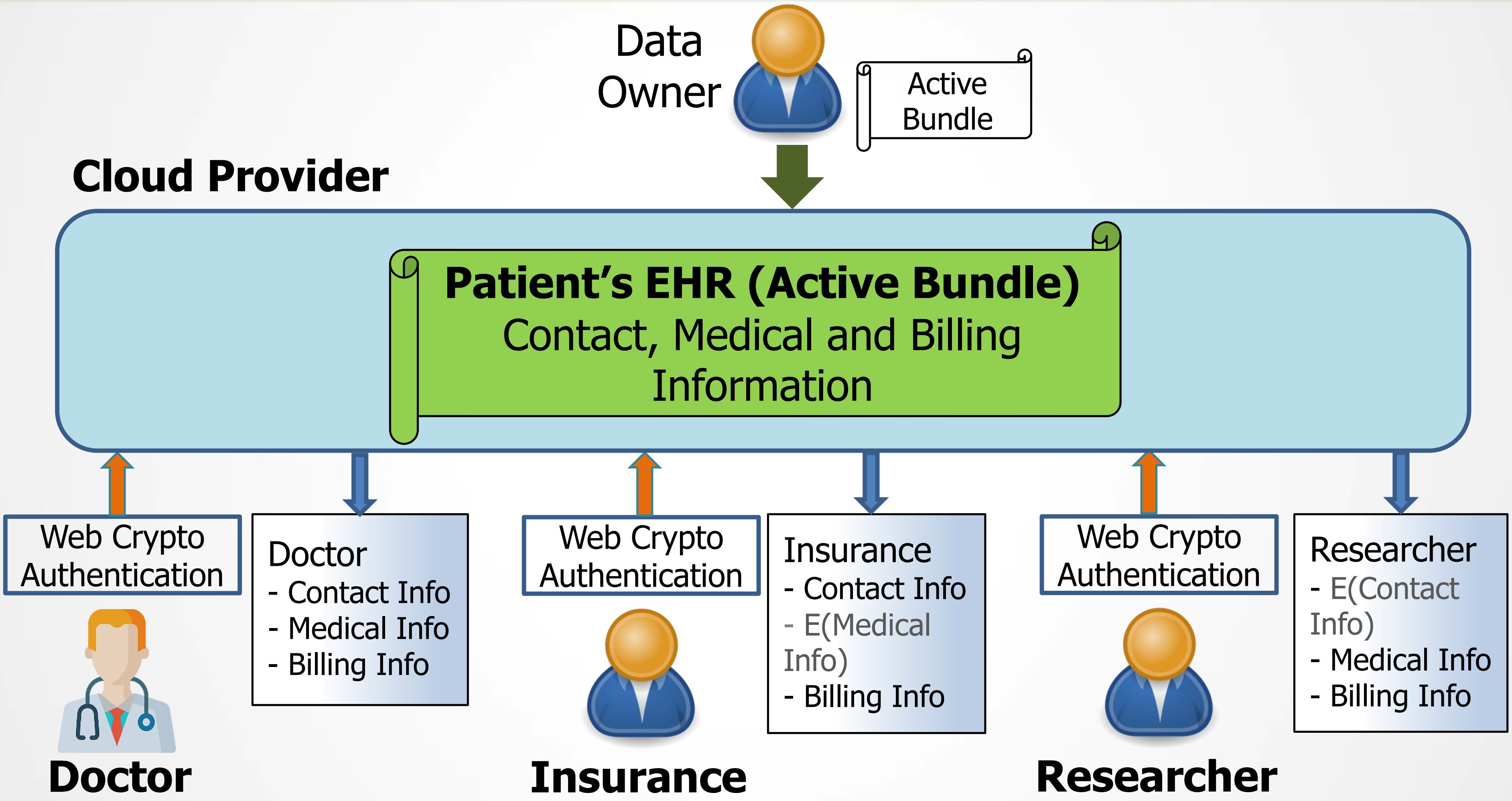
ALLOW	
Resource	patientID
Subject's Role	Doctor, Insurance, Researcher
Action	Read

ALLOW	
Resource	name
Subject's Role	Doctor, Insurance
Action	Read

Adversary Model:

- Malicious client who tries to gain unauthorized access to encrypted data, stored in AB, and to bypass access control policy check
- Authorized insider who leaks data to unauthorized parties

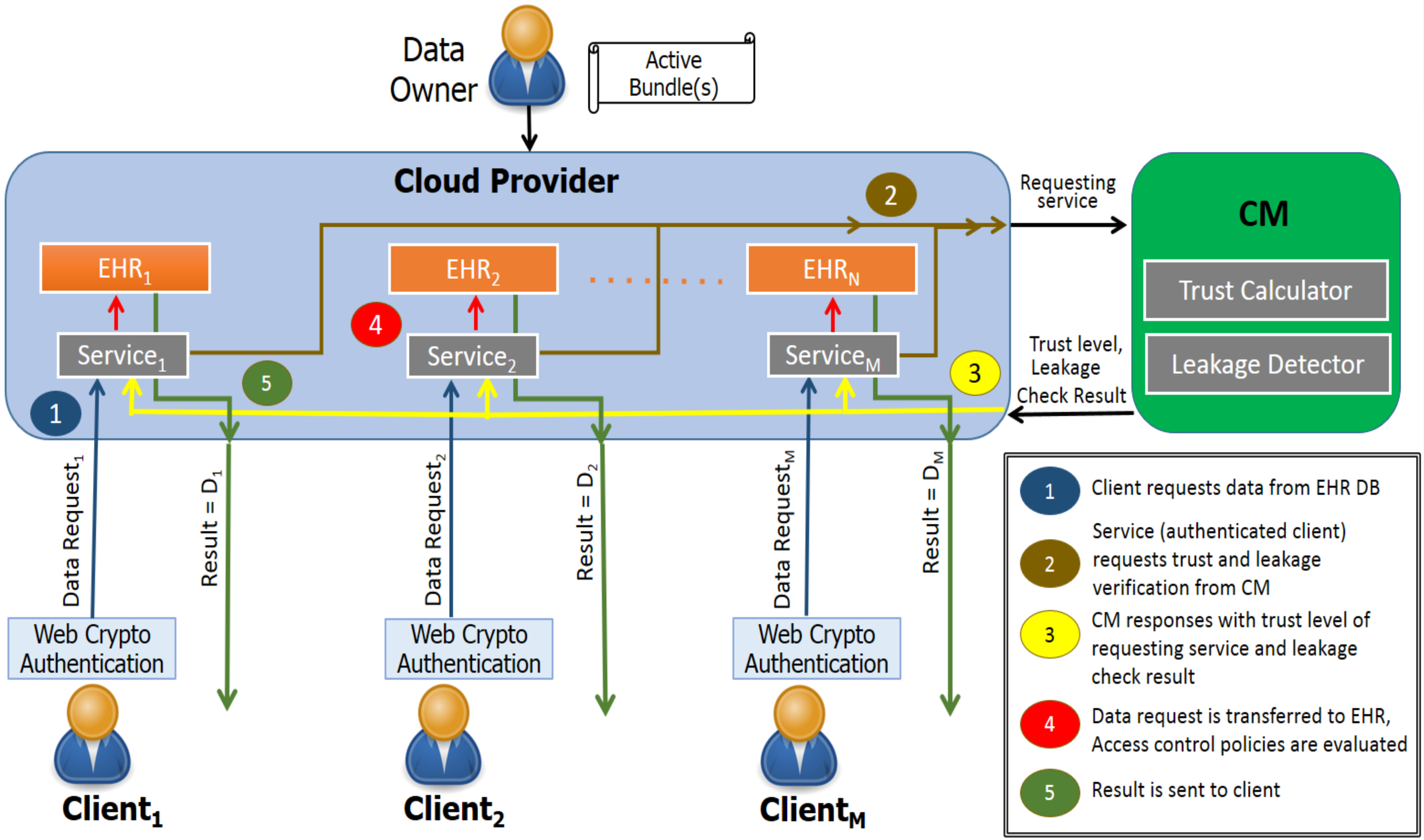
WAXEDPRUNE



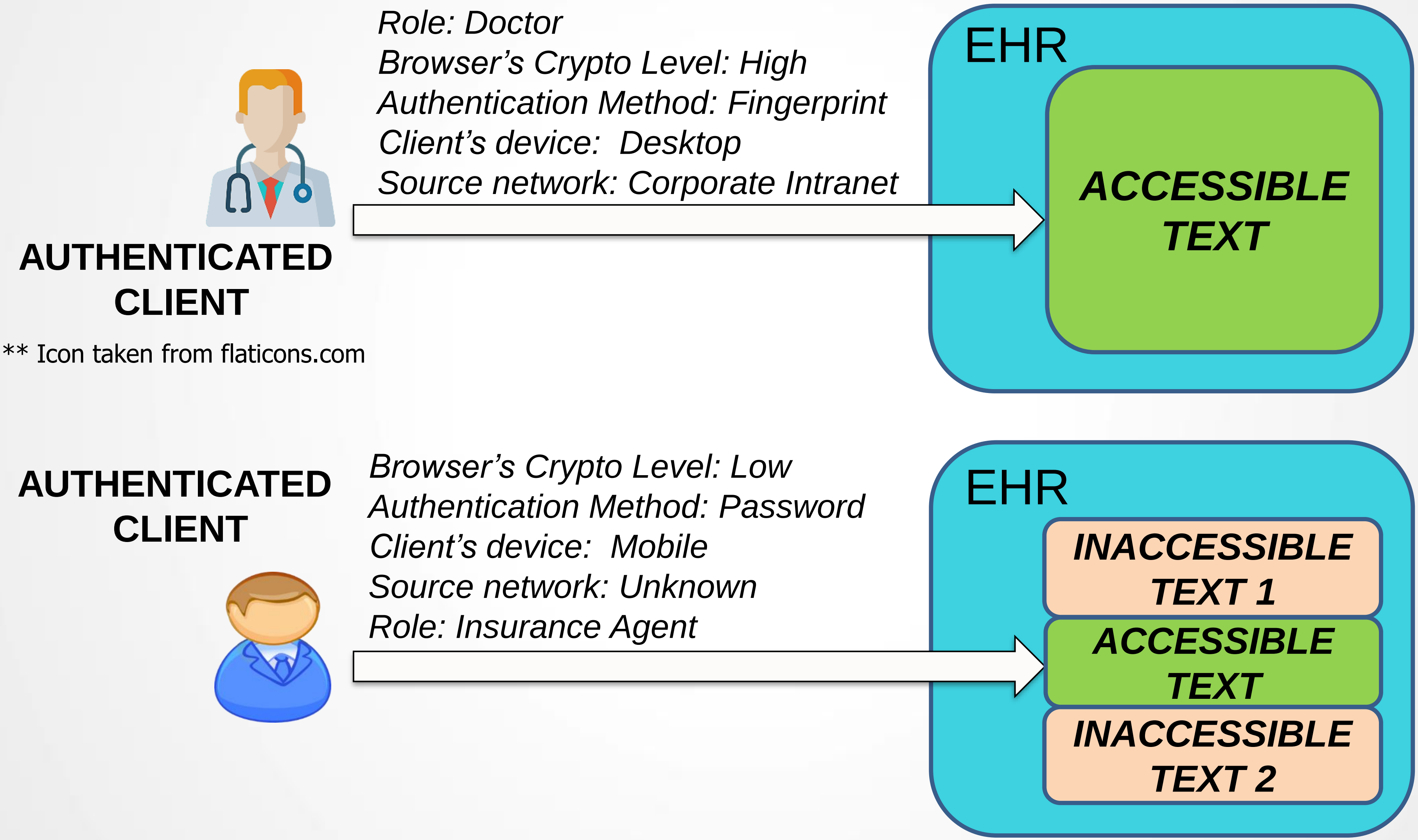
** Icon taken from flaticons.com

Cloud-based EHR Access Scenario (suggested by Dr. Leon Li, NGC)

Framework Architecture



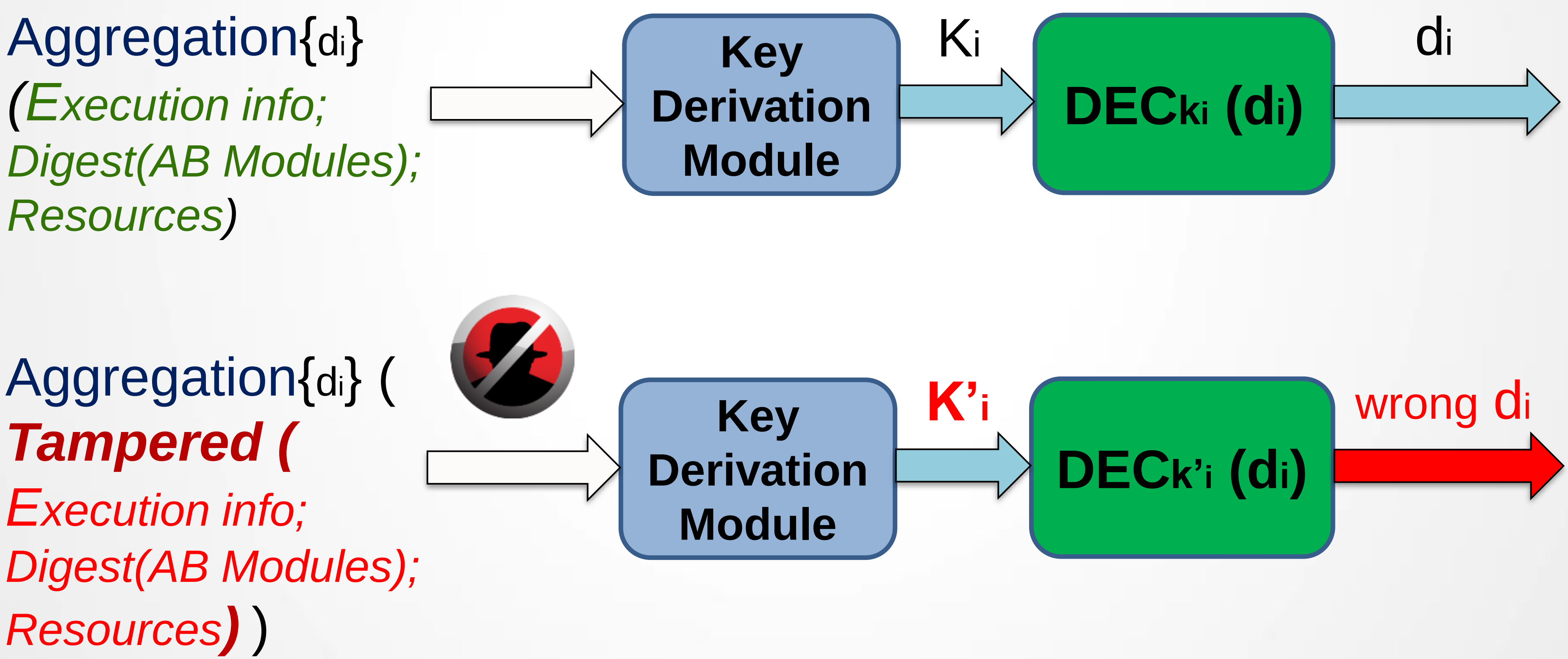
Role- and Attribute-based Access Control



** Icon taken from flaticons.com

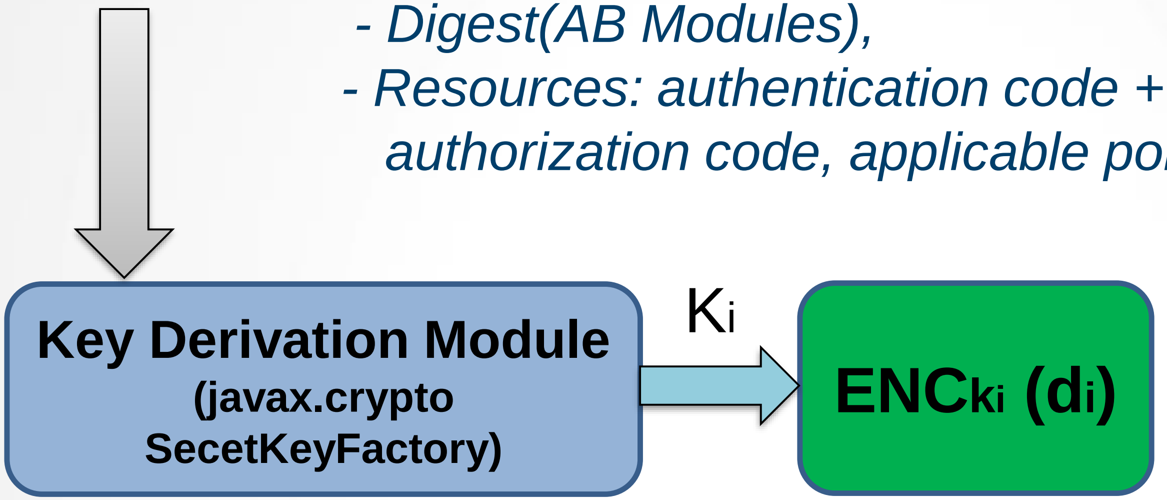
Tamper Resistance of AB

- Key is not stored inside AB [1], [5], [2]
- Separate symmetric key is used for each separate data set
- Ensure protection against tampering



Key Generation

Aggregation{d_i} (*- Generated AB modules execution info;*
- Digest(AB Modules),
- Resources: authentication code + CA certificate, authorization code, applicable policies + evaluation code)



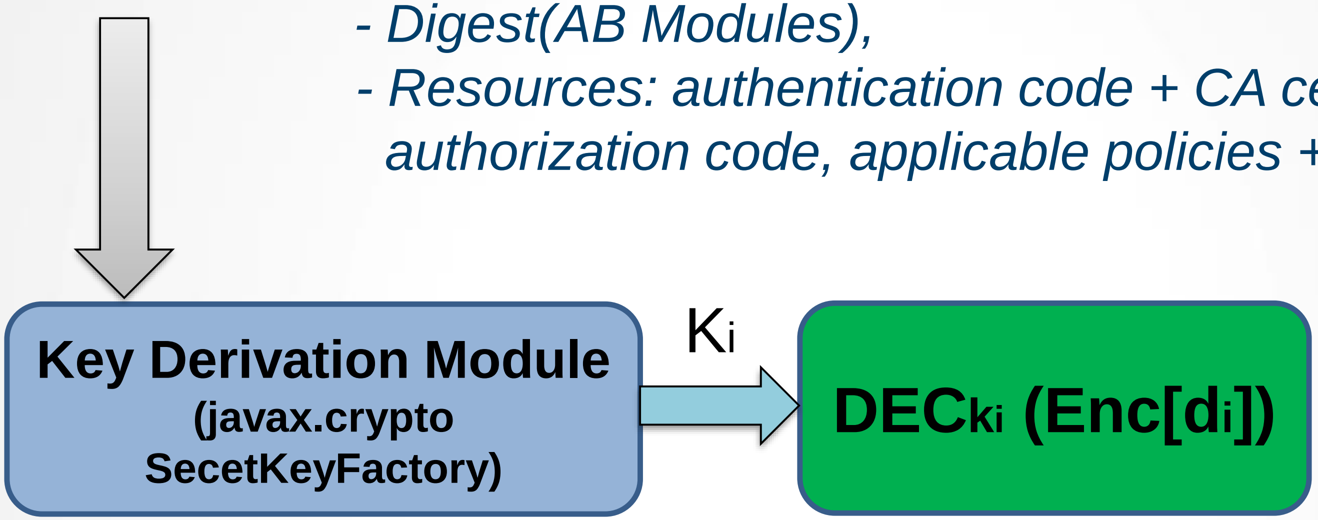
- AB Template [1] used to generate new ABs with data and policies (specified by data owner)
- AB Template includes implementation of invariant parts (monitor) and placeholders for customized parts (data and policies)
- AB Template is executed to simulate interaction between AB and service requesting access to each data item of AB

Key Generation (Cont.)

- Info generated during the execution and digest (modules) and AB resources are collected into a single value
- Value for each data item is input into a Key Derivation module (such as ***SecretKeyFactory, PBEKeySpec, SecretKeySpec*** from ***javax.crypto*** library)
- Key Derivation module outputs the specific key relevant to the data item
- This key is used to encrypt the related data item [1]

Key Derivation

Aggregation{d_i} (- *Generated AB modules execution info;*
- *Digest(AB Modules),*
- *Resources: authentication code + CA certificate, authorization code, applicable policies + evaluation code*)



- AB receives data item request from a service
- AB authenticates the service and authorizes its request (evaluates access control policies)[1]

1. "Cross-Domain Data Dissemination and Policy Enforcement", R. Ranchal, PhD Thesis, Purdue University, Jun. 2015.

Key Derivation (Cont.)

- Info generated during the AB modules execution in interaction with service, and digest (AB modules) and AB resources are aggregated into a single value for each data item [1]
- Value for each data item is input into the Key Derivation module
- Key Derivation module outputs specific key relevant to data item
- This key is used decrypt the requested data item
- If any module fails (i.e. service is not authentic or the request is not authorized) or is tampered, the derived key is incorrect and the data is not decrypted

Other Key Distribution Methods

- Centralized Key Management Service
  - TTP used for key storage and distribution
  - TTP is a single point of failure

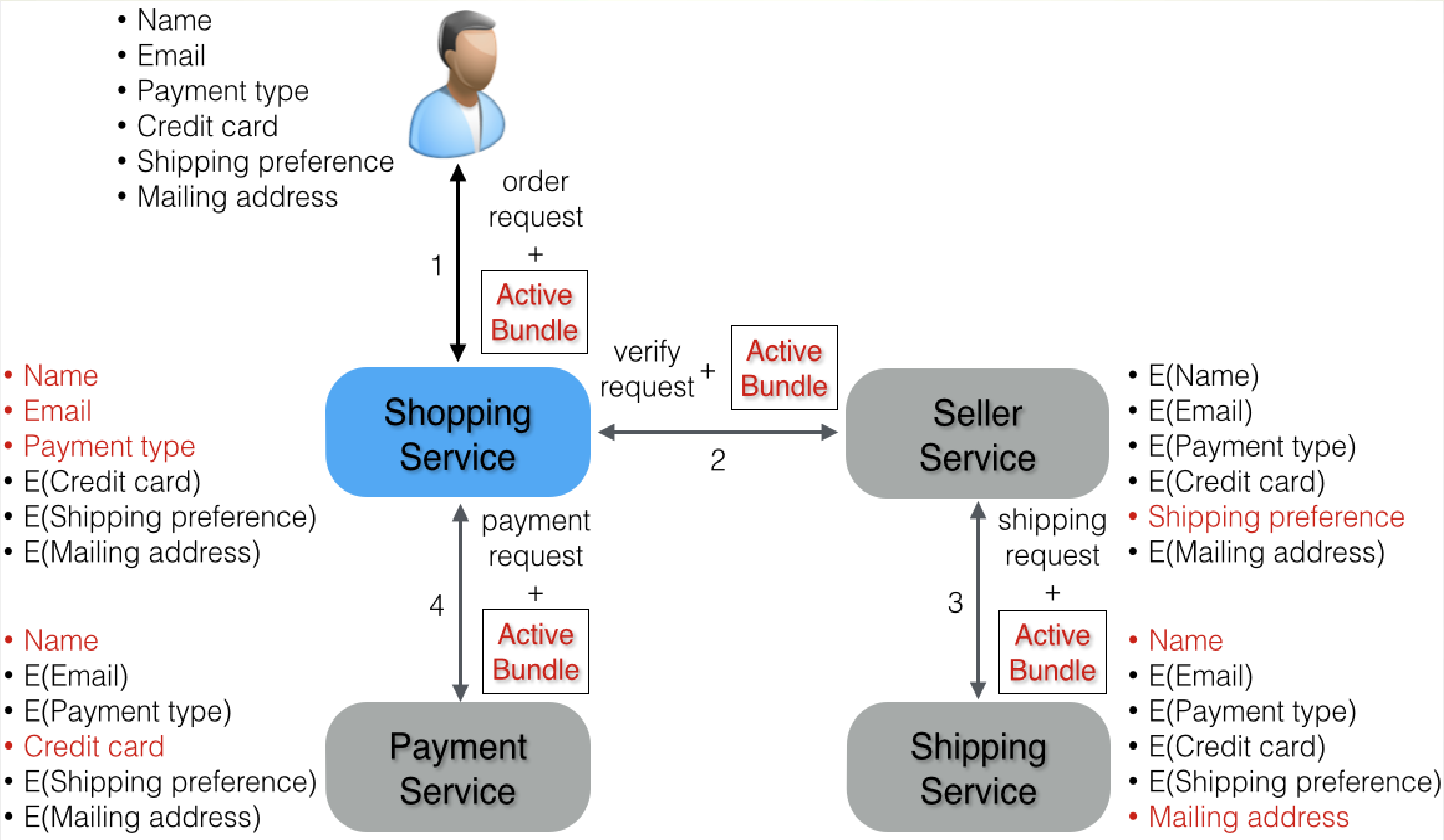
- Key included inside AB
  - Prone to attacks!

AB Use Cases

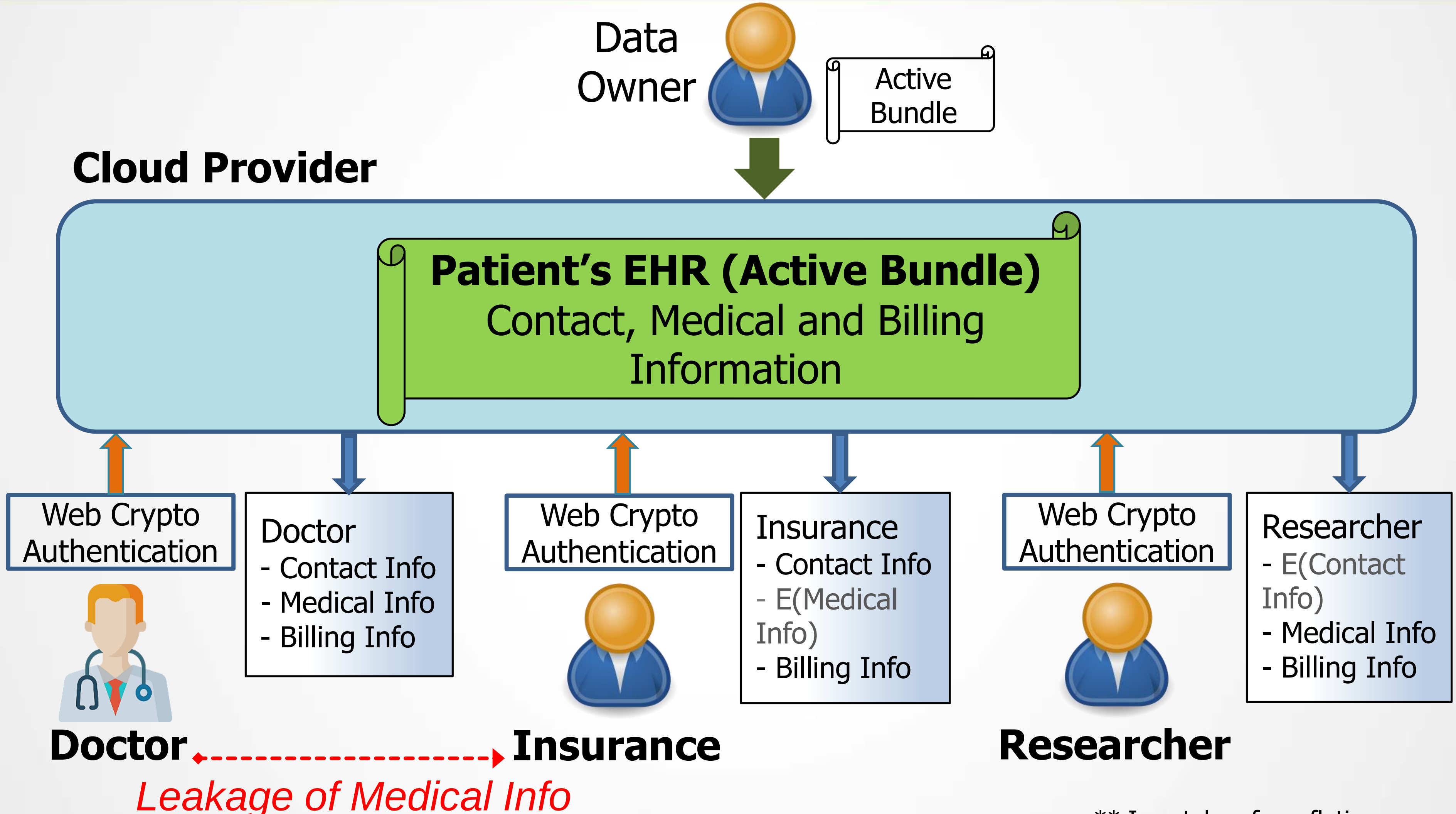
- **Hospital Information System (collection of EHRs)**
  - Doctor, Researcher and Insurance are authorized for different parts of patient's EHR [3]. [5], [8]
  - Database of EHRs is hosted by untrusted cloud provider
- **Secure Email**
  - Email is AB
  - Entire email can be sent to the whole mailing list
  - Recipients are authorized for different fragments of email
  - It is guaranteed for the sender that each recipient will only see those email fragments it is authorized for
  - No need for multiple mailing lists for different authorization levels
- **Secure dissemination of video data [2]**
  - Different policies used for video with and w/o human faces
- **Online shopping [4]**
  - Decentralized data accesses: data can travel across the services

15

AB in P2P network: Online Shopping



Data Leakage Detection



** Icon taken from flaticons.com

Cloud-based EHR Access and Leakage Scenario (suggested by Dr. Leon Li, NGC)

Recent Data Leakages Examples

Company	Time	Incident Details
Adobe Systems	Oct.2013	150 million accounts of software subscription database got leaked
Anthem	Feb.2015	78.8 million of PII records got leaked
Experian Information Solutions and T-Mobile, USA	Sep.2015	Data (SSN, credit card information) of about 15 million customers who applied for credit got leaked
U.S. Office of Personnel Management: Agency of the U.S. Federal government	Jun.2015	SSN, names, addresses, places of birth of 22 million people got leaked

Data leakage detection

How can data get leaked by authorized subject [7]?

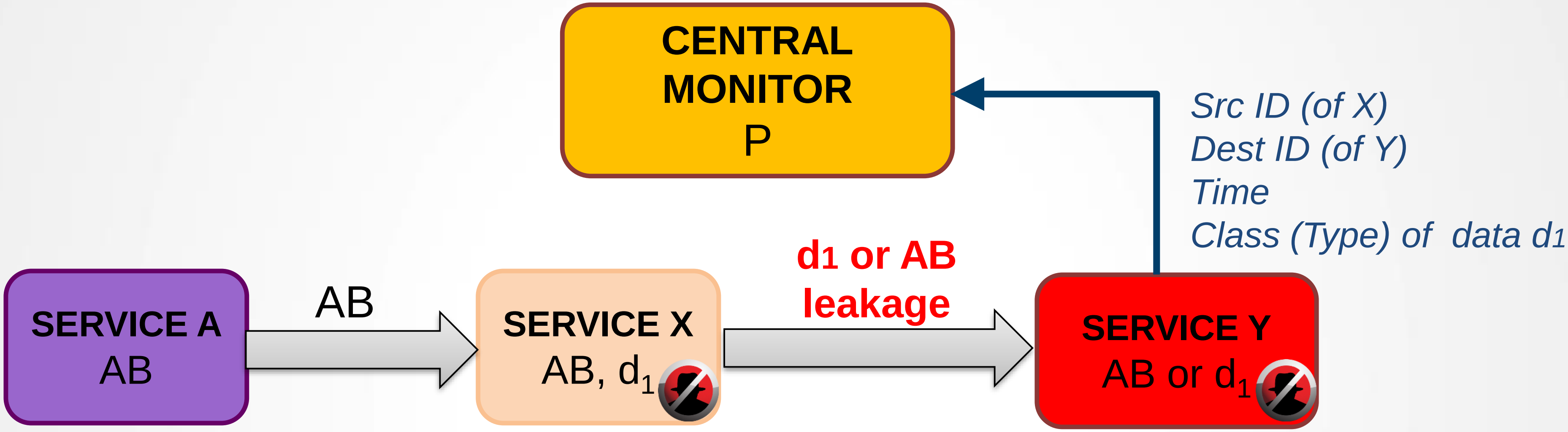
- In the form of encrypted data (the whole AB is leaked):
  - Data is protected by AB, but fact of leakage can be detected
  - Detection is based on enforcing access control policies by a Central Monitor (CM): how data is used by authorized party?
    - *When party tries to decrypt data from AB, CM is notified*
    - *Without CM acknowledgement decryption process will not proceed*
    - *CM checks whether data is supposed to be where they are*

Data leakage detection

How can data get leaked by authorized subject [7]?

- In the form of decrypted (raw) data:
  - Data is not protected by AB anymore
  - Detection based on:
    - Digital watermarks embedded into data (e.g. png-images), provided images are accessible by a web crawler (watermarking checker)
    - Visual watermarks embedded into data

Core Design: Data Leakage Detection



AB contains:

- Enc [Data(D)] = $\{Enc_{k1}(d_1), \ldots, Enc_{kn}(d_n)\}$
- Access Control Policies (P) = $\{p_1, \ldots, p_k\}$

- Service X is authorized to read d_1 from AB
- Service X may leak decrypted d_1 or the entire AB to Y

Core Design: Data Leakage Detection

- When service tries to decrypt AB data, CM is notified about that: “Service Y tries to decrypt d_1 arrived from X“
- If CM is unreachable, decryption terminates
- CM checks against centralized DB of policies: whether d_1 is supposed to be at Y. If NO then:
  - Blacklist X, Y
  - Reduce their trust level
  - Mark data d_1 as compromised and notify services about it
  - Raise the level of d_1 classification

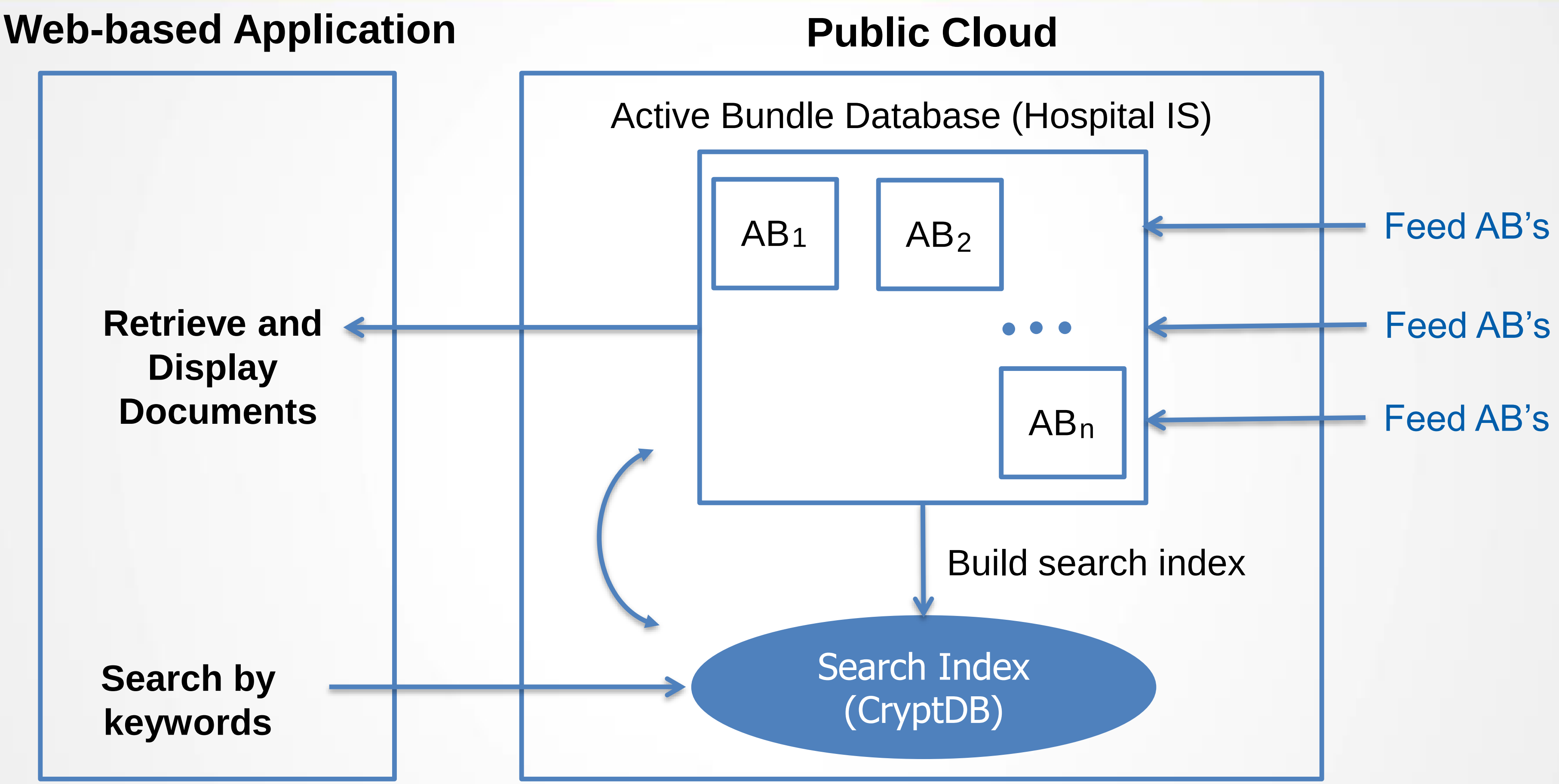
Anti-fragility

- **After leakage is detected, make system stronger against similar attacks**
  - Separate compromised role into two: *suspicious_role* and *benign_role*
  - Send new certificates to all benign users for *benign role*
  - Create new Active Bundle with new policies, restricting access to *suspicious_role* (e.g. to all doctors from the same hospital with a malicious one)
  - Increase sensitivity level for leaked data items, i.e. for diagnosis
  - Disable “Save As” functionality or exclude highly sensitive data from what can be stored locally

Data Leakage Mitigation Methods

- **Layered Approach:** Don't give all the data to the requester at once
  - First give part of data (incomplete, less sensitive)
  - Watch how it is used and monitor trust level of using service
  - If trust level is sufficient – give next portion of data
- **Raise the level of data classification** to prevent leakage repetition
- **Intentional leakage** to create uncertainty and lower data value
- **Monitor network messages**
  - Check whether they contain e.g. credit card number that satisfies specific pattern and can be validated using regular expressions [14]

Encrypted Search over Encrypted Data



Encrypted Search over Encrypted Data stored in Cloud (suggested by Dr. Leon Li, NGC)

Example:

select prescription from Hospital_IS where diagnosis = "Insomnia";

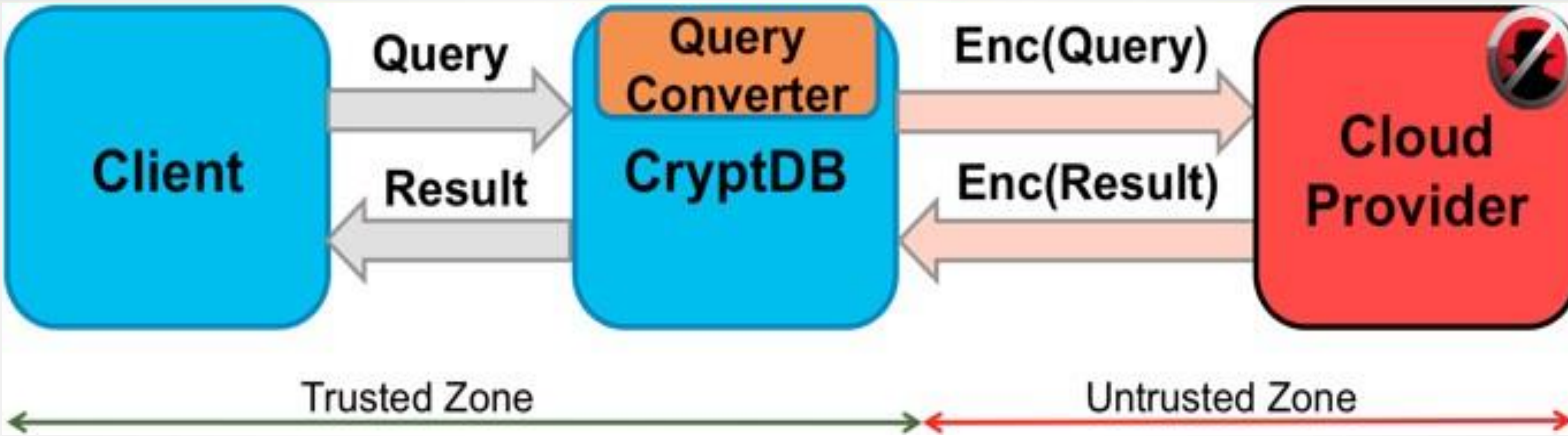
Encrypted Search over AB Database

- Collection agent gathers intelligence feeds (ABs)
- AB contains extra-attribute used for indexing
- CryptDB is a proxy to a database server
  - Stores encrypted data (keywords, abstract of AB) and provides SQL query capability over encrypted data
  - Never releases decryption key to a database
  - When compromised, only ciphertext is revealed and data leakage is limited to data for currently logged in users
- Subscription API provides methods for authorized access to data
  - Phase 1: filter out relevant ABs (e.g. top-20)
  - Phase 2: execute data request to relevant ABs only [6]
- Use case 1:
  - Get prescription for patients diagnosed with "***Insomnia***"

select prescription from Hospital_IS where diagnosis = "Insomnia";

!!! Note: due to vulnerabilities, recently discovered in CryptDB, it is recommended to use Microsoft SQL Server 2016 instead

Encrypted Search over Encrypted Records



<u>Use case 2:</u> law enforcement needs personal data of drivers who exceeded speed limit of 65 mph and went above 76 mph

<u>Initial Query</u>: `SELECT ID FROM IndexDB WHERE SPEED > 76`

<u>Converted query</u>: `SELECT c1 FROM Alias1`

`WHERE ESRCH (Enc(Speed), Enc(76));`

<u>Second phase query</u>: http get request for driver’s license number from VRs with relevant IDs from previous query

Encrypted Search over Encrypted Records

Index Database

Use case 3: ITS needs to figure out traffic pattern during rush hour. Speed between 55 and 65 => no traffic

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(84)	Enc(Chevrolet)	02/18/2018 23:59

Initial Query: `select ID from IndexDB WHERE speed between 55 and 65`

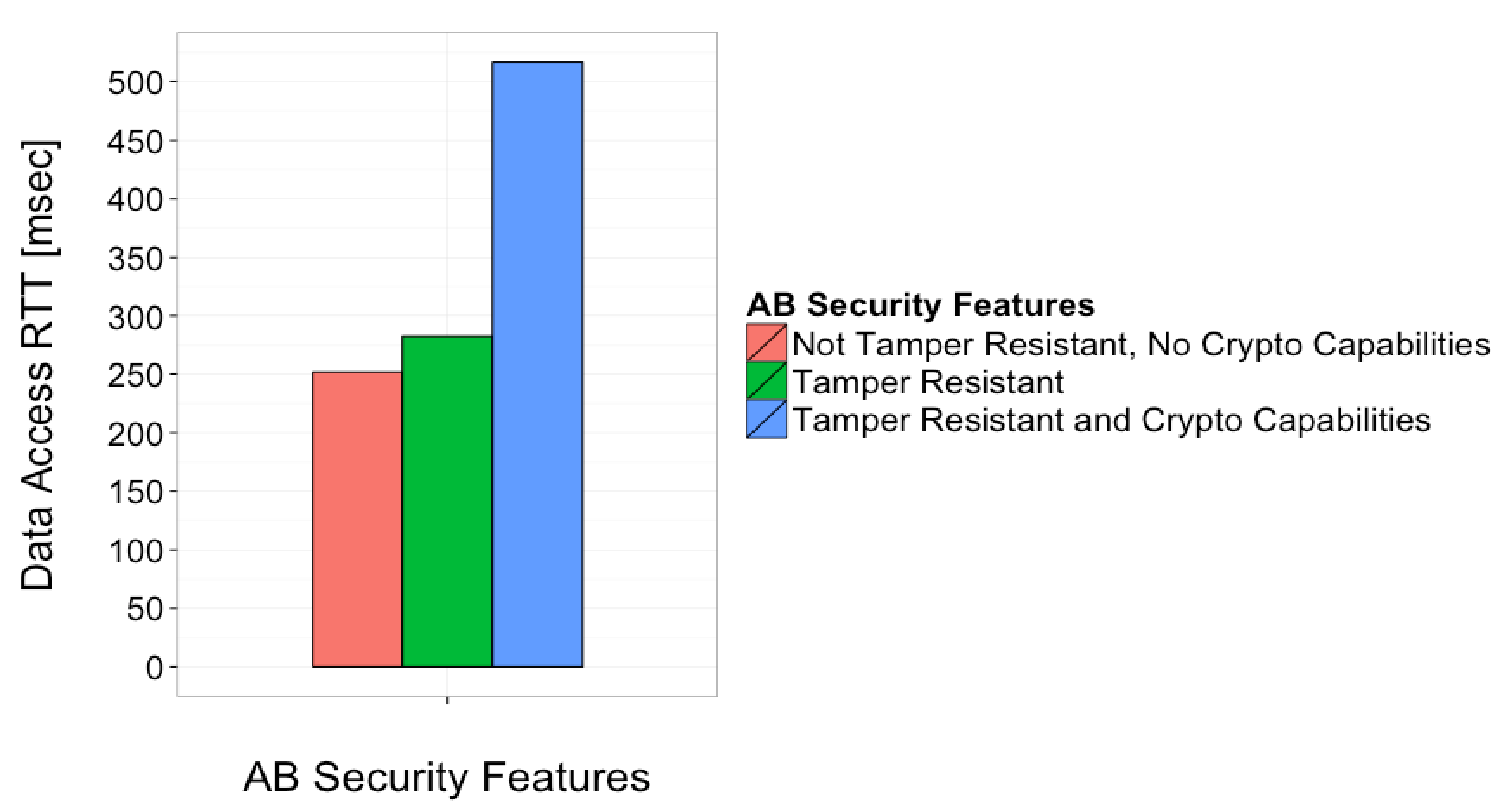
Converted query: `SELECT c1 FROM Alias1 WHERE ERANGE (Enc(Speed), Enc(55), Enc(65);`

Second phase query: http get request for vehicle's license plate number from VRs with relevant IDs from previous query

Operations supported by different encryption schemes

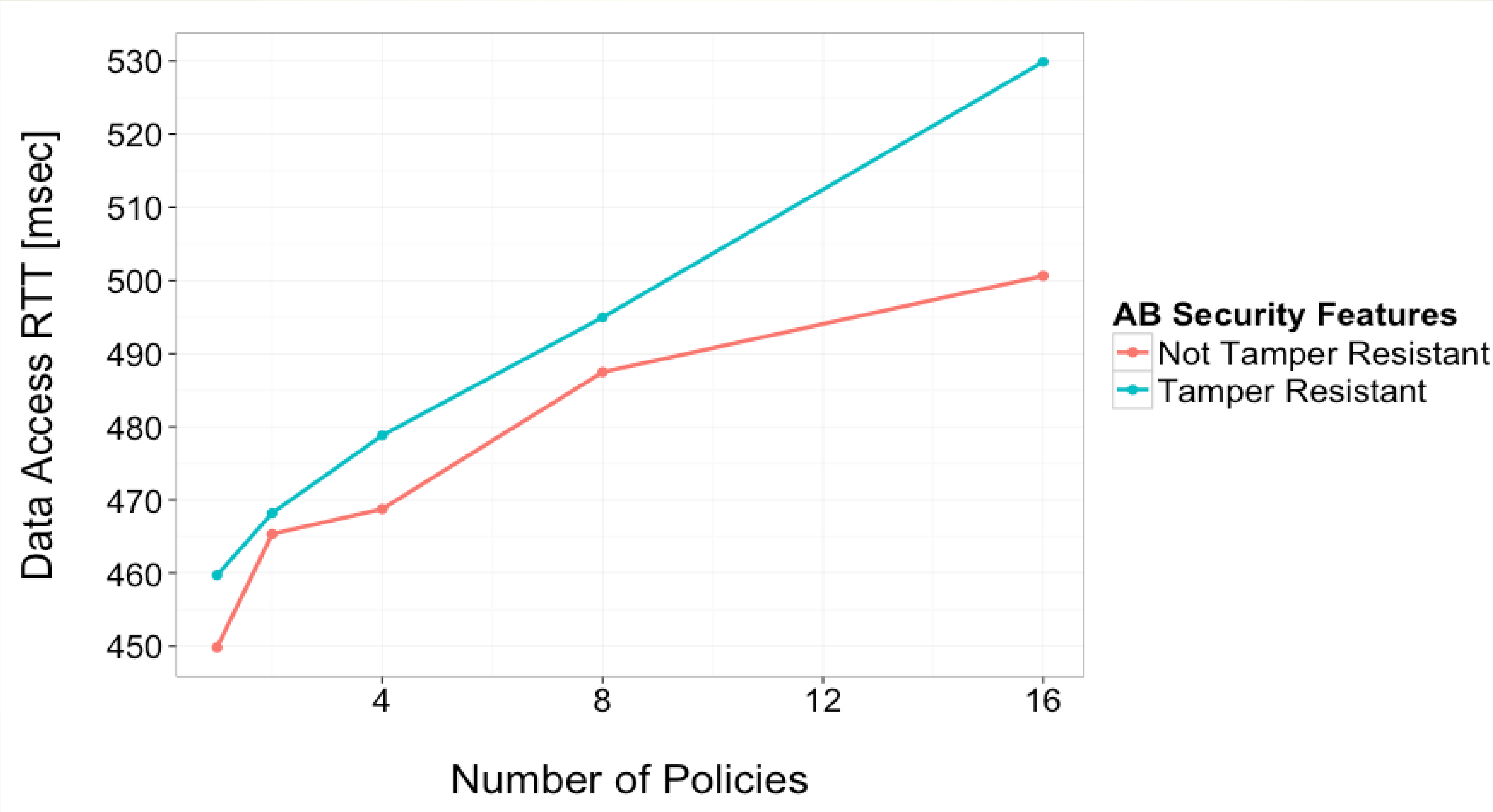
Crypto System	**Supported operations**	**Example**
Pallier (AHE)	+, SUM	Count sum of salaries
El-Gamal (MHE)	+, SUM, *	Count salary which is multiplication: hourly_wage * hours
OPE (Order-Preserving Encryption)	>, <, MIN, MAX, ...	Select patient from EHR_DB where AGE in between 25 and 35
SWP (SRCH)	Substring searches (LIKE in SQL queries)	Select prescription from EHR_DB where diagnosis LIKE %insomnia%
DET (deterministic)	Exact searches	Select patient from EHR_DB where Name = 'John Doe'

Evaluation



Performance overhead of Active Bundle with detection of browser's crypto capabilities on / off

Evaluation



Performance overhead of Active Bundle, hosted by Google Cloud

Conclusions

Secure Data Exchange in "WAXEDPRUNE" is based on [3]:

- Access control policies [16]
- Trust level of a subject (service, user)
- Context (e.g. emergency vs. normal)
- Security level of client's browser (crypto capabilities) [12], [13]
- Authentication method (password-based, fingerprint, etc)
- Source network (secure intranet vs. unknown network)
- Type of client's device: desktop vs. mobile (detected by Authentication Server)

Conclusions

- Assumption: hardware and OS are trusted

- To relax these assumptions, Intel SGX trusted platform might be used

- Data is extracted from Active Bundle at a server side and send to client via https

  - Data confidentiality is preserved

- Multiple types of Data Leakages are prevented/detected by using:

  - Active Bundles
  - Digital watermarks, embedded into data. Watermarks are checked by web crawlers,
  - Visual watermarks

Conclusions

- CryptDB never releases decryption key to a database

- provides database privacy
- protects database from curious or malicious cloud administrators

- CryptDB weak points:

- OPE is not secure in terms of revealing the order [19]
- Does not support queries having a + b*c
- Does not support SQL queries with “LIKE”

Solution: use Fully Homomorphic Encryption (FHE)

- It is 9x slower than CryptDB [9]
- Pallier and El-Gamal don’t reveal order

Contributions

WAXEDPRUNE contributes to Data Confidentiality and Integrity

- Dissemination does not require data owner’s availability
- TTP-independent for recipient’s key generation
- Trust level of subjects is constantly recalculated
- On-the-fly key generation
- Supports data updates for multiple subjects
- Agnostic to policy language and evaluation engine
- Tamper-resistance: data and policies integrity is provided

Contributions

WAXEDPRUNE contributes to Data Confidentiality and Integrity

- Supports encrypted search over database of ABs
- Provides prevention/detection of multiple types of data leakages, made by malicious authorized insiders, and leakage damage assessment
- Captures data provenance for use in leakage measure and forensics
- Compatible with industry-standard SOA/cloud frameworks
  - RESTful services
  - X.509 certificates

References

1. R. Ranchal, “Cross-domain data dissemination and policy enforcement,” PhD Thesis, Purdue University, Jun. 2015
2. C. Qu, D. Ulybyshev, B. Bhargava, R. Rohit, and L. Lilien. "Secure Dissemination of Video Data in Vehicle-to-Vehicle Systems.” 6th Intl. Workshop on Dependable Network Computing and Mobile Systems (DNCMS2015), Sep. 2015
3. D. Ulybyshev, B. Bhargava, M. Villarreal-Vasquez, D. Steiner, L. Li, J. Kobes, H. Halpin, R. Ranchal, A. Alsalem "Privacy - Preserving Data Dissemination in Untrusted Cloud", IEEE Cloud, June 2017
4. R. Ranchal, D. Ulybyshev, P. Angin, and B. Bhargava. “Policy-based Distributed Data Dissemination,” *CERIAS Security Symposium, April 2015* ***(Best poster award)***
5. D. Ulybyshev, B. Bhargava, L. Li, J. Kobes, D. Steiner, H. Halpin, B. An, M. Villarreal, R. Ranchal. “Authentication of User’s Device and Browser for Data Access in Untrusted Cloud,” *CERIAS Security Symposium, April 2016*.
6. D. Ulybyshev, A. Alsalem, B. Bhargava, S. Savvides, G. Mani, L. Ben-Othmane "Secure Data Communication in Autonomous V2X Systems", IEEE ICIOT, July 2018
7. D. Ulybyshev, B. Bhargava, A. Alsalem "Secure Data Exchange and Data Leakage Detection in Untrusted Cloud", Springer Journal on Applications of Computing and Communication Technologies, on 1-st Intl Conf. ICACCT 2018, pp. 99-113.
8. D. Ulybyshev, B.Bhargava, “Secure dissemination of EHR,” demo video

https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0

References

9. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. “CryptDB: Protecting confidentiality with encrypted query processing”. In ACM SOSP, 2011
10. “Lightweight data-interchange format JSON,” http://json.org/ , accessed: Oct.2018
11. “eXtensible access control markup language (XACML) version 3.0,” http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html, accessed: Oct. 2018
12. “W3C Web Cryptography API,”https://www.w3.org/TR/WebCryptoAPI/, accessed: Oct.2018
13. “Web authentication: an API for accessing scoped credentials,” http://www.w3.org/TR/webauthn , accessed: Oct.2018
14. “Finding or verifying credit card numbers,” http://www.regularexpressions.info/creditcard.html , accessed: Oct.2018
15. “WSO2 Balana Implementation,” https://github.com/wso2/balana , accessed: Oct.2018
16. L. Lilien and B. Bhargava, ”A scheme for privacy-preserving data dissemination,” IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 36(3), May 2006, pp. 503-506.
17. L. Ben Othmane and L. Lilien, “Protecting privacy in sensitive data dissemination with active bundles,” 7-th Annual Conf. on Privacy, Security and Trust (PST 2009), Saint John, New Brunswick, Canada, Aug. 2009, pp. 202-213
18. L. Lilien and B. Bhargava, ”A scheme for privacy-preserving data dissemination,” IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 36(3), May 2006, pp. 503-506.

References

19. Order-Preserving Encryption (OPE) and leakage: https://crypto.stackexchange.com/questions/37375/order-preserving-encryption-ope-and-leakage/37385 , accessed: Nov. 2018