Security and Privacy Investigation of Existing mHealth Applications

Prepared by Nuray Baltaci Akhuseyinoglu

Outline

1) Security testing for Android mHealth apps (Knorr & Aspinall, 2015)
2) Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice (Papageorgiou et al., 2018)

Security testing for Android mHealth apps (Knorr & Aspinall, 2015)

Security testing for Android mHealth apps

- A testing method for Android mHealth apps which is designed using a threat analysis,
  - considering possible attack scenarios and vulnerabilities specific to the domain
  - rather than out-of-the-box Android security testing methods.
- Motivation:
  - mHealth apps collect and transmit private medical data and do not do a good job of securing it.
  - Current regulation only marginally addresses security requirements of mHealth apps, and doesn’t propose any testing process.
  - A carefully designed, transparent, reproducible security testing method for mHealth apps is highly desirable.

Security testing for Android mHealth apps

- A testing strategy which is a synthesis of known tools and techniques, with some novel, context-specific extensions.
- Main point: by focusing on a specific sub-category of apps, more precise security (and safety) issues can be addressed
- Proposed testing method is applied to **hypertension and diabetes management apps**
    - Blood pressure apps: systolic and diastolic blood pressure (mmHg), and sometimes heart rate (bpm).
    - Diabetes apps: blood glucose concentration (mmol/L or mg/dL).
    - discovered a number of serious vulnerabilities in the most popular applications!

Threat and Vulnerability Analysis

A. Threat modelling

- Three types of threat:
    - Unauthorized access to health data: Somebody discovers the health information belonging to an individual.
    - Tampering with health data: An attacker alters the health data that is recorded or reported by an app.
    - Reporting invalid health data. An app reports wrong information to the user or healthcare professional.
- Threat agents:
    - Health insurance companies
    - Employers
    - Data intelligence companies
    - Investigators

Threat and Vulnerability Analysis

B. Attack surface and vulnerabilities

A1. Eavesdropper

A2. Active attacker on the network

A3. Man in the Middle

A4. App shop owners

A5. App developers

A6. Malware developer

A7. Third parties

A8. Attacker with physical access to smartphone

A9. The user

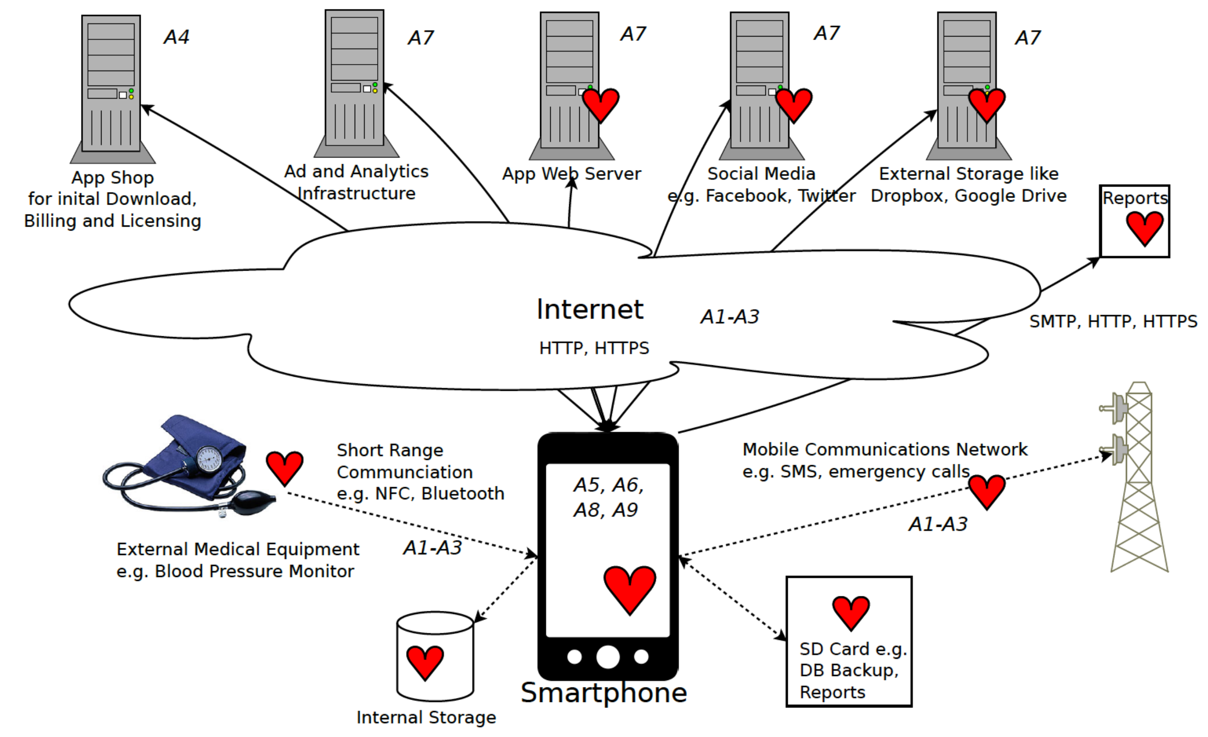
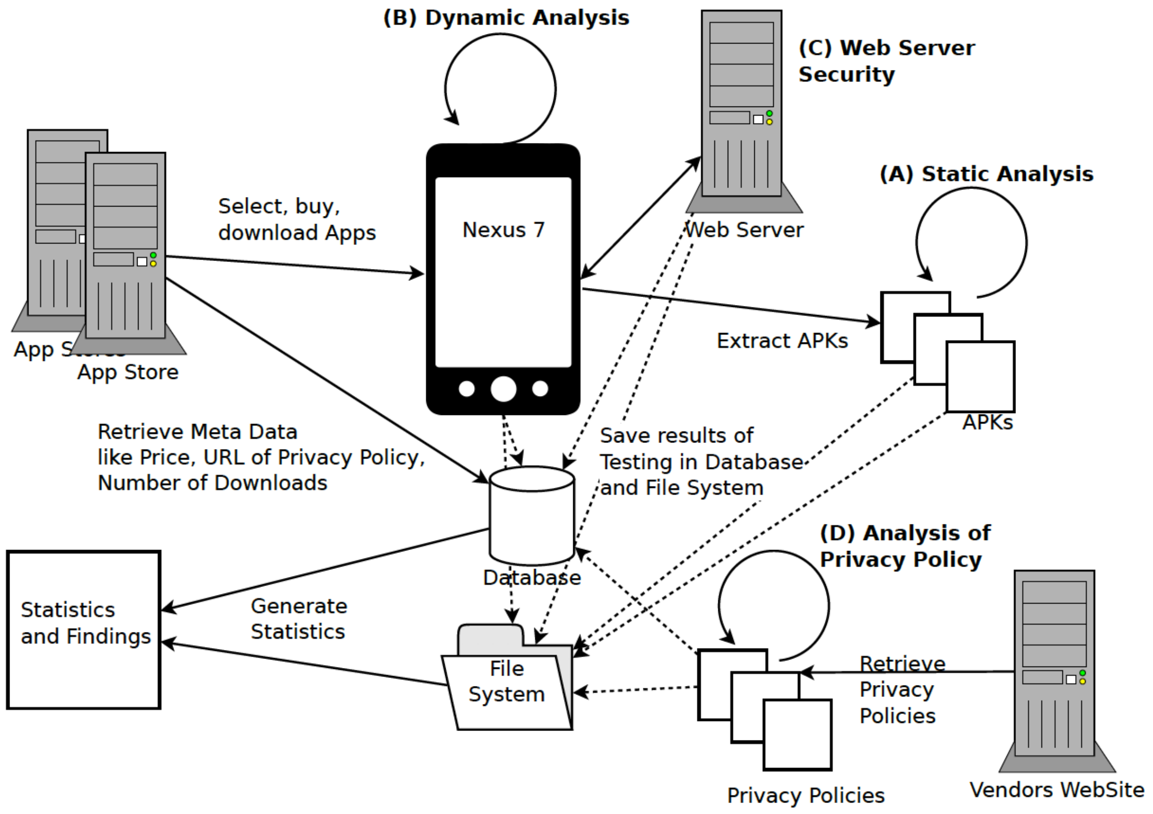

A4
App Shop for inital Download, Billing and Licensing
A7
Ad and Analytics Infrastructure
A7
App Web Server
A7
Social Media e.g. Facebook, Twitter
A7
External Storage like Dropbox, Google Drive
Reports
Internet
HTTP, HTTPS
A1-A3
SMTP, HTTP, HTTPS
Short Range Communciation e.g. NFC, Bluetooth
External Medical Equipment e.g. Blood Pressure Monitor
A1-A3
A5, A6, A8, A9
Smartphone
Mobile Communications Network e.g. SMS, emergency calls
A1-A3
Internal Storage
SD Card e.g. DB Backup, Reports

Fig. 1. An mHealth app in context. The heart indicates fine-grained medical data. *A1* – *A9* indicate possible attackers

Testing Method



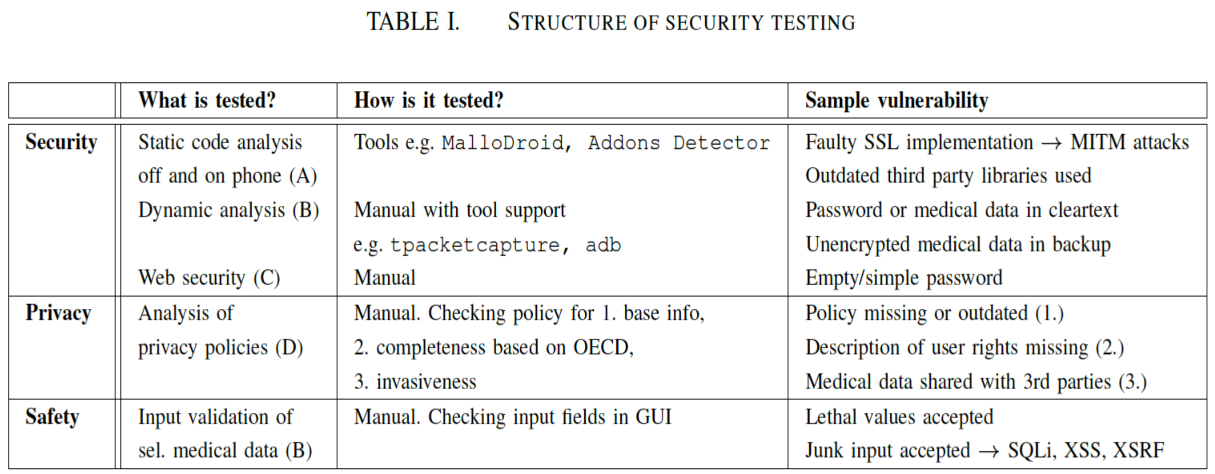


TABLE I. STRUCTURE OF SECURITY TESTING

	What is tested?	How is it tested?	Sample vulnerability
Security	Static code analysis off and on phone (A)	Tools e.g. `MalloDroid, Addons Detector`	Faulty SSL implementation → MITM attacks Outdated third party libraries used
	Dynamic analysis (B)	Manual with tool support e.g. `tpacketcapture, adb`	Password or medical data in cleartext Unencrypted medical data in backup
	Web security (C)	Manual	Empty/simple password
Privacy	Analysis of privacy policies (D)	Manual. Checking policy for 1. base info, 2. completeness based on OECD, 3. invasiveness	Policy missing or outdated (1.) Description of user rights missing (2.) Medical data shared with 3rd parties (3.)
Safety	Input validation of sel. medical data (B)	Manual. Checking input fields in GUI	Lethal values accepted Junk input accepted → SQLi, XSS, XSRF

Testing Method- A) Static Analysis

- based on the information contained in the APK file,
  - including the manifest and the compiled code (in the file classes.dex)
- First step: extract the apps’s permissions, which tell what actions are allowed:
  - like Internet, Bluetooth, NFC usage or access to the contacts in the address book
- AndroGuard* tool is used to extract permissions

* https://code.google.com/p/androguard/

Testing Method- A) Static Analysis cont’d

- Second step: testing for vulnerabilities
  - **Proper SSL usage (A3):** Faulty usage of SSL can be identified with MalloDroid
  - **Debug flag (A8):** checked with Drozer
  - **Content providers (A5, A6, A8):** checked with Drozer
  - **Use of encryption (A1, A8):** inspect disassembled smali code produced with the apktool
  - **Poor use of certificate parameters (A1, A3):** OpenSSL can extract certificate information (X.509) from the APK files

Testing Method- A) Static Analysis cont’d

- Second step: testing for vulnerabilities
  - **Code quality (A5, A6, A8):** FindBugs[1] to measure the number of likely-bug patterns
  - **Add-ons (A7):** Addons Detector tool[2] to identify and classify add-on libraries.
  - **Malware and Privacy scanners (A5):** several apps from security vendors to detect malicious apps:
    - Snoopwall Privacy App[3]
    - Clueful [4]
    - AVG Antivirus Security[5]
    - AVAST[6]
    - McAfee Security & Antivirus[7]
    - Recap vulnerability scanner[8]

1) http://findbugs.sourceforge.net/
2) https://play.google.com/store/apps/details?id=com.denper.addonsdetector
3) https://play.google.com/store/apps/details?id=com.snoopwall.privacyapp
4) https://play.google.com/store/apps/details?id=com.bitdefender.clueful
5) https://play.google.com/store/apps/details?id=com.antivirus
6) https://play.google.com/store/apps/details?id=com.avast.android.mobilesecurity
7) https://play.google.com/store/apps/details?id=com.wsandroid.suite
8) https://play.google.com/store/apps/details?id=com.palindrome.ph

Testing Method- B) Dynamic Analysis

- Mostly manual; the process is difficult to automate
- Investigation of whether abnormal and illegal inputs are accepted, and how exported data is stored or transmitted.
- The tests are:
  - **Input validation (A9):** Does the app accept abnormally high input values? With warning? Can junk values (non-numeric) be entered?
  - **Data leakage (A1, A6, A8):** Test all available export routes (e-mail, web server, social media) to see if data was stored or transmitted unencrypted.
    - Use of **tPacketCapturepro**, **DroidWall , Wireshark**
  - **Data wipe (A8):** Check if the app includes a feature to erase all stored medical data?
  - **Privacy policy (A7):** Is a reference to a privacy policy given in the app?
  - **Reasonable permissions (A5):** over-privileged or not?
  - **Secure backup and logging (A6, A8):** any unencrypted medical data in logs?
    - Use of adb (Android debug bridge)

Testing Method- C) Web Server Connection

- Following security issues are analyzed related to web security:
  - **Web Server connection (A1–A3):** Checking if a secure transport (https:) is used, by recording URLs
    - Why?
    - Record traffic to see if passwords or medical data could be sniffed in clear text.
  - **Web Server authentication (A2):** Investigate the effectiveness of the server authentication
    - By checking password policies
    - If weak passwords are allowed, an attacker may guess or brute force entry to the server.

Testing Method- D) Inspection of Privacy Policies

- **Collect basic privacy policy information:**
  - URL to privacy policy, number of words (indicating coverage), version and country of origin (jurisdiction)

- **Invasiveness:** Answers on the scale “Yes”, “No”, “Partly”:
  1) Can medical data be used for other purposes like marketing or research?
  2) Is the data stored by a third party?
  3) Is it passed on to 3rd parties generally, or in case of company mergers/acquisition?

- **Completeness:** Questions to test OECD privacy principles*, answering “Yes”, “No”, “Partly”:
  1) Accountability: Is the data controller named or detailed contact data given?
  2) Security safeguards: are safeguards described?
  3) Openness: are types of data collected described?
  4) Purpose specification: is the purpose and usage of the data selected described?
  5) Individual participation: are the rights of the individual described?

*http://www.oecd.org/internet/ieconomy/oecdguidelinesontheprotectionofprivacyandtransborderflowsofpersonaldata.htm

Case Study

- mHealth apps for diabetes and blood pressure selected in November 2014
  - 154 apps (55% diabetes, 35% blood pressure, 10% both)
  - The static analysis tests were applied to all the apps – part (A)
  - 72 most frequently downloaded apps for dynamic analysis – part (B)
  - 20 apps were addressed in parts (C) and (D).
- Invented a persona as a dummy patient for someone:
  - 170 cm tall and weighs 99 kg,
  - has dangerously high blood pressure of 200 mmHg/120 mmHg,
  - a physiologically improbable heart rate (333 bpm),
  - a blood glucose level of 111 mmol/L, which is lethal and perhaps suggests a user confused mg/dL with mmol/L.

Case Study – Test Environment and Tools

- Test device : a Nexus 7 running Android 4.4.2.
- A laptop running Ubuntu Linux 12.04 to connect to the device and to run the tools.
- Several Python (v2.7.3) scripts for data extraction, conversion and transferring.
- The Google Play Unofficial Python API* to extract meta data
    - like prices, download numbers, ratings and permission from Google's Play Store

* https://github.com/egirault/googleplay-api

Case Study- Findings

- **Encryption of health data is hardly ever provided.** Only one of the tested apps allows to encrypt.
- **Validation of health data is sketchy.** Many apps failed to make bounds checks in inputted health data and record invalid (or fatal) values.
- **Advertising is common and leaks package identities.** Of the apps tested, 74 include advertisement addons like AdMob:
  - often transmit the app’s package name in clear text in the HTTP header.
- **Privacy policies are often missing or inadequate.** The majority (over 80%) of apps do not link to their privacy policy in the corresponding app stores.

Discussion

Automated vs. manual testing

- Most steps of the testing in the dynamic analysis were manual. => difficult to automate, why?

Coverage

- Established secure coding and general testing guides:
    - CERT secure coding guidelines for Android*: proposed method covers 9 of 27 test cases, 8 are uncovered, 10 are not relevant for the test set.
    - OWASP [30]: entirely or partly covers 7/8 static categories, and 3/7 dynamic categories.

False positives in tool output

- Findings of static analysis tools can have false positives and (ideally) should be manually verified.

* https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=111509535
** https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Mobile_Security_Testing

Conclusion

- Comprehensive security testing for Android mHealth apps is expansive
- Some tools for automated testing are available
    - only cover a small area and by far not the entire spectrum required.
- Fully automated checking is desirable but unrealistic today.

Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice (Papageorgiou et al., 2018)

Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice

- Contributions:
  - Long term and in-depth security and privacy analysis of m-health apps from January 2016 to August 2017
  - Providing developers of the inspected apps with security reports
  - Investigating the way that app developers responded to the reports
  - Performing a GDPR (General Data Protection Regulation) compliance auditing procedure

- The alarming conclusion:
  - The majority of the analyzed apps does not meet the expected standards for security and privacy,
  - Thus endanger their users' sensitive personal data.

http://www.internationalinvestment.net/wp-content/uploads/2016/03/GDPR-740x360.jpg

Methodology

- The initial tests were performed from January to February 2016
  - using Android devices and apps downloaded from the official Android marketplace (i.e., Google Play).
- A re-evaluation process was run a year later, from July to August 2017
  - after notifying each app's vendor on the initially identified issues,
  - based on dynamic analysis tests,
  - in order to verify any conformance to the previously discovered findings.
- A GDPR compliance auditing procedure to determine whether the reviewed apps conform to the new EU legal requirements.

GDPR

- The General Data Protection Regulation (GDPR):
  - A new stringent legal framework for protecting individuals' personal data
  - Adopted by European Commission in 2016
  - replaced the 1995's Data Protection Directive and became directly applicable to all EU Member States on May 2018*, harmonizing the various national regulations across the EU.

- Enforcement of the GDPR in May 2018 within the EU was expected to meet with technical challenges: (Papageorgiou et al., 2018)
  - tracking and deleting user disseminated data to third parties
  - designing and developing internal procedures satisfying GDPR auditing and data protection requirements

https://eur-lex.europa.eu/eli/reg/2016/679/oj

Methodology-Data Collection

- Test Android apps retrieved from Google Play
  - due to the Android's Operating System (OS) popularity
  - a set of 1080 of the most popular apps from the ``Medical'' and ``Health and Fitness’’ categories

- Selected 20 apps by applying the inclusion criteria, categorized as:
  i. pregnancy and baby growth,
  ii. personal/family members' health agenda and symptoms assistants/checkers
  iii. blood pressure and diabetes support.

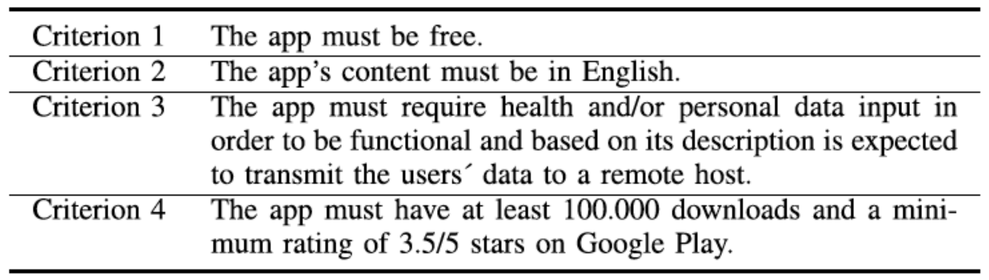


TABLE 1. Inclusion criteria.

Criterion 1	The app must be free.
Criterion 2	The app’s content must be in English.
Criterion 3	The app must require health and/or personal data input in order to be functional and based on its description is expected to transmit the users´ data to a remote host.
Criterion 4	The app must have at least 100.000 downloads and a minimum rating of 3.5/5 stars on Google Play.

Methodology-Assessment Methodology

- Main research questions:
    - Which parties have access to personal data from the app?
    - What exact data can each party access?
    - How safe is each communication channel?

Methodology-Assessment Methodology

- Steps to provide answers to research questions:
    1) Register personas in the app (fake user accounts)
    2) Collect permissions and inspect privacy policies
    3) Automated static code analysis
    4) Dynamic analysis: manual analysis of (intercepting) communications btw each app and 3rd party-Figure 1
    5) Analysis of the web server configuration to assess the security level of the HTTPS data transmission. => inspection of each packet to determine the content of it
    6) Summary of findings for each app vendor and informing them
    7) Estimate each apps' readiness against the GDPR's requirements

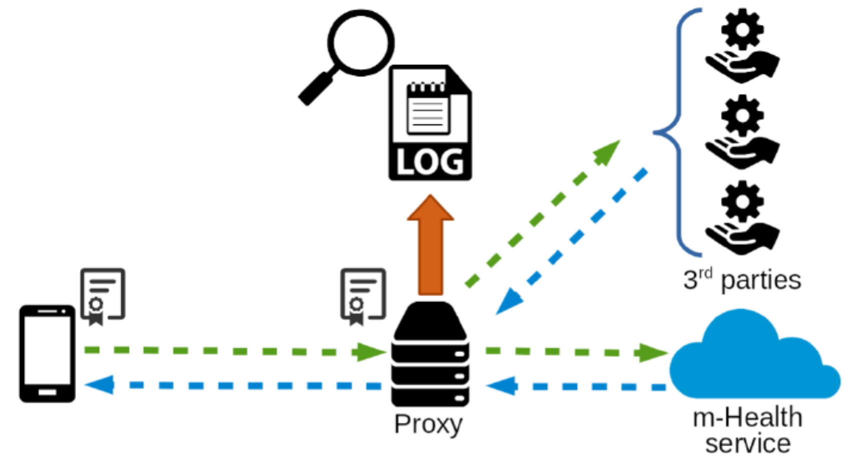


FIGURE 1. Scheme of the interception setup.

Results-Manual Analysis

1) Privacy policies
   - Initial results on February 2016:
     - 10%: no reference to a privacy policy page
     - 5%: a link to privacy policy but a 404 error page
     - 5%: a link to a non-English privacy policy page
2) Permissions Analysis
   - permissions from the Manifest files of the apps APKs using python scripts
   - “dangerous” and “normal” permissions
   - Figure 2 => the number of apps that requested dangerous permissions

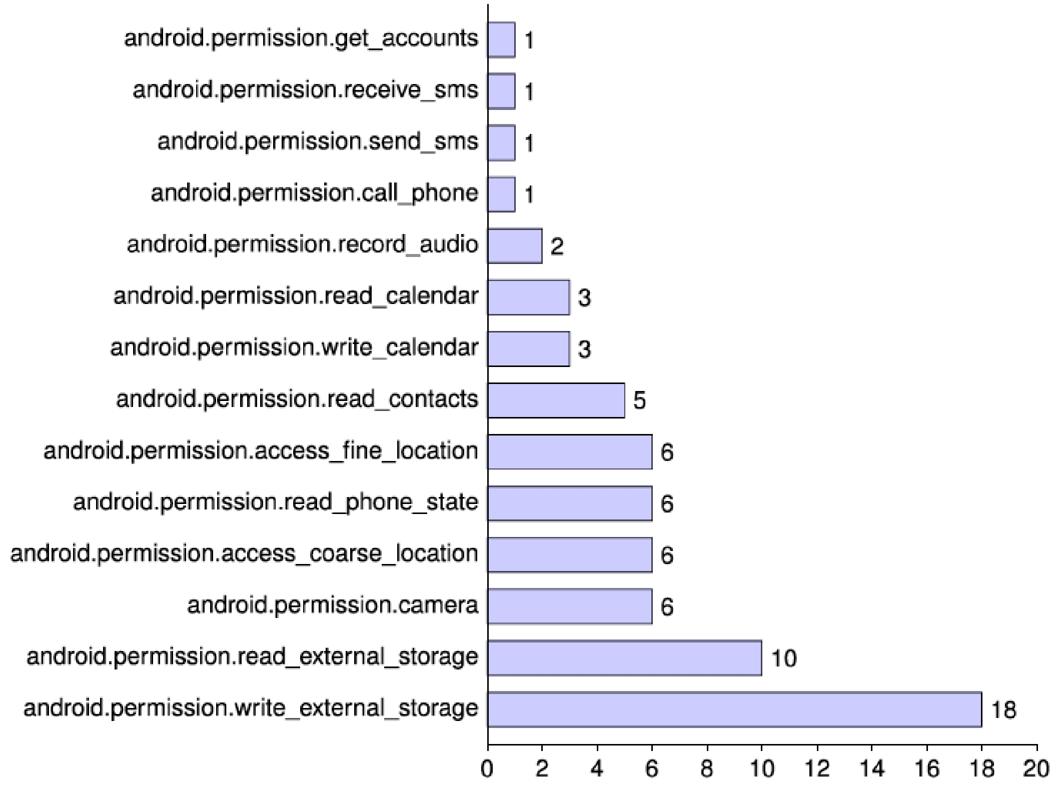


FIGURE 2. Summary of dangerous permission requests.

Results- Static Code Analysis

- To detect possible vulnerabilities
    - Analysis of the APK of each app using MobSF*
- Results of the analysis: security issues summarized in Table 3
- Some important findings:
    - Many apps do not connect using HTTPS and have several issues concerning AndroidWebViews components.
    - 45% of the apps tried to determine whether the device was rooted, a feature irrelevant to their goals.

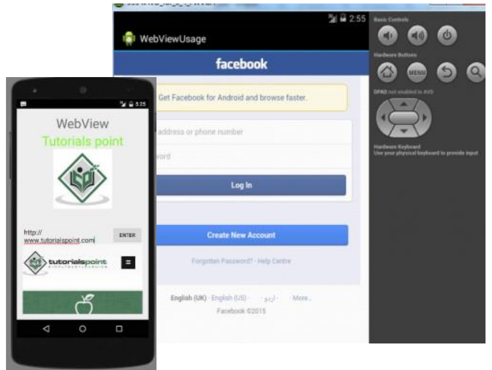
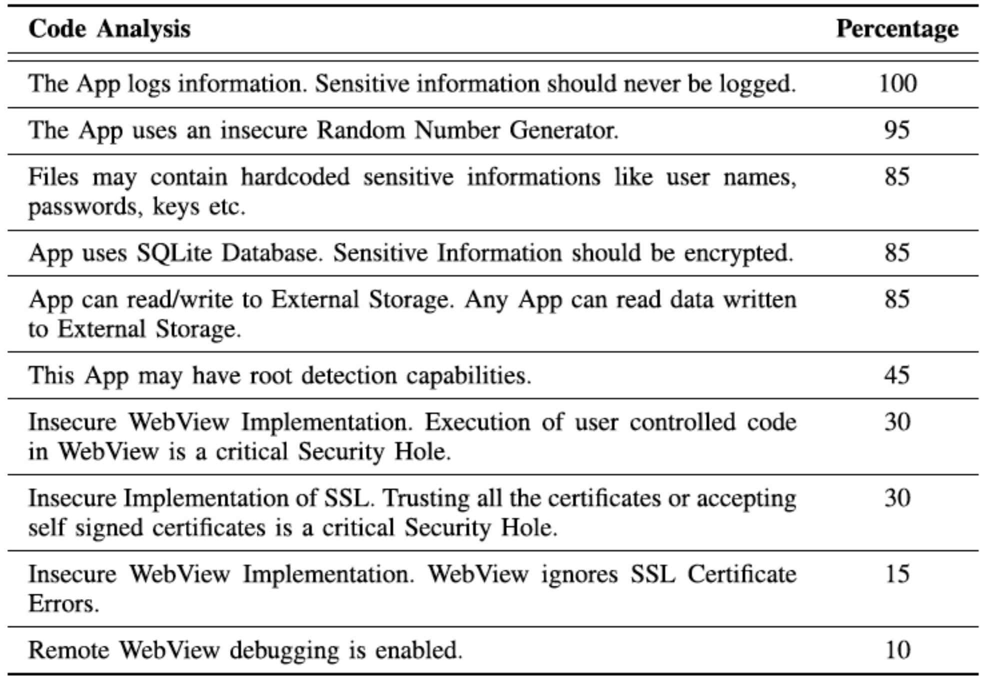


TABLE 3. Results of the static code analysis.



Code Analysis	Percentage
The App logs information. Sensitive information should never be logged.	100
The App uses an insecure Random Number Generator.	95
Files may contain hardcoded sensitive informations like user names, passwords, keys etc.	85
App uses SQLite Database. Sensitive Information should be encrypted.	85
App can read/write to External Storage. Any App can read data written to External Storage.	85
This App may have root detection capabilities.	45
Insecure WebView Implementation. Execution of user controlled code in WebView is a critical Security Hole.	30
Insecure Implementation of SSL. Trusting all the certificates or accepting self signed certificates is a critical Security Hole.	30
Insecure WebView Implementation. WebView ignores SSL Certificate Errors.	15
Remote WebView debugging is enabled.	10

* http://opensecurity.in/mobilesecurity-framework/
** https://proandroiddev.com/android-webviews-1cbe1ffb7a2b
*** https://www.tutorialspoint.com/android/android_webview_layout.htm

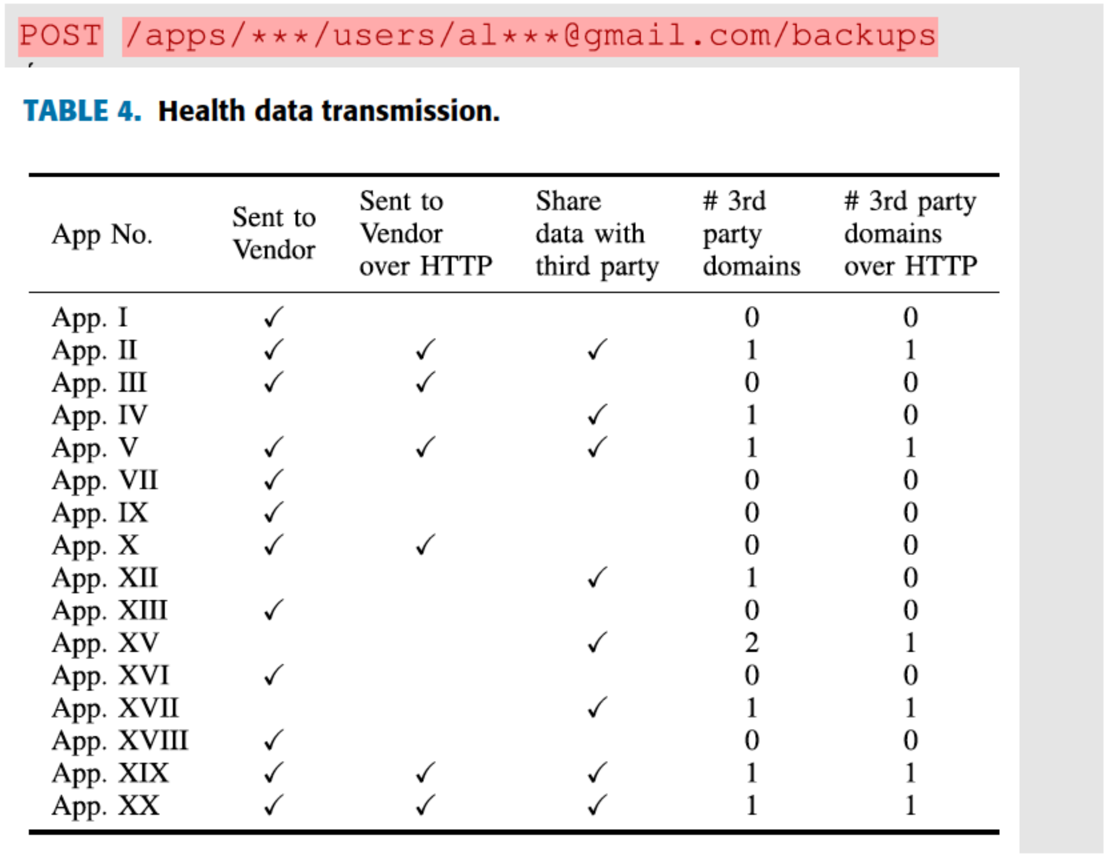
Results- Dynamic Analysis

- Evaluation of the security and privacy of apps
  - during transmission of sensitive and personal data over the Internet
- Findings are discussed based on the type of the data that apps transmit:
  1) Health-related data
  2) Location privacy
  3) User’s registration and log in security
  4) Email and device ID transmission
  5) Users’ search query privacy and OS type

Results- Dynamic Analysis cont’d

1) Health-related Data

- Capture all keywords and/or phrases related to the health status of the user
  - to identify the transmitted health information
  - using the Fiddler web debugging tool
- Experiment results:
  - 80% : transmit users' health-related data
  - 20%: store locally on the device
  - only 50%: transmit health-related data over HTTPS connections



```
POST /apps/***/users/al***@gmail.com/backups
```

TABLE 4. Health data transmission.

App No.	Sent to Vendor	Sent to Vendor over HTTP	Share data with third party	# 3rd party domains	# 3rd party domains over HTTP
App. I	✓			0	0
App. II	✓	✓	✓	1	1
App. III	✓	✓		0	0
App. IV			✓	1	0
App. V	✓	✓	✓	1	1
App. VII	✓			0	0
App. IX	✓			0	0
App. X	✓	✓		0	0
App. XII			✓	1	0
App. XIII	✓			0	0
App. XV			✓	2	1
App. XVI	✓			0	0
App. XVII			✓	1	1
App. XVIII	✓			0	0
App. XIX	✓	✓	✓	1	1
App. XX	✓	✓	✓	1	1

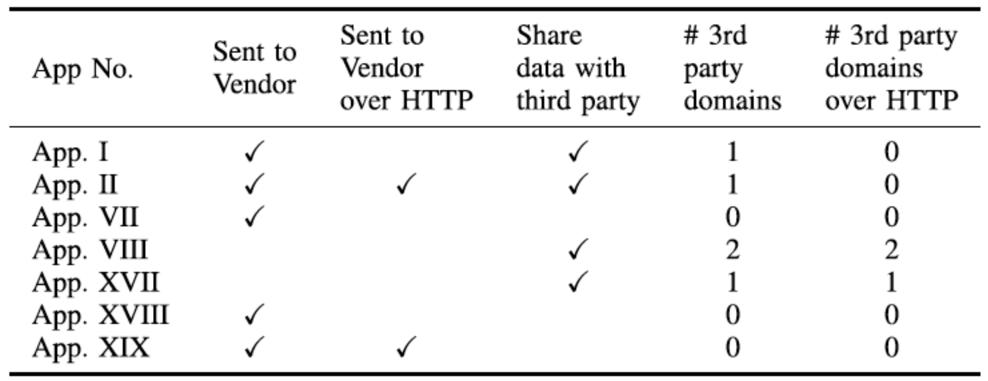
FIGURE 3. Part of a JSON response to a POST request over HTTP containing health-related data.

Results- Dynamic Analysis cont’d

2) Location Privacy

- Table 5: findings about transmitted user location information
- 35%: transmitted users’ geolocation information or their postal address either to their vendors or to third parties

TABLE 5. User’s location transmittion.



App No.	Sent to Vendor	Sent to Vendor over HTTP	Share data with third party	# 3rd party domains	# 3rd party domains over HTTP
App. I	✓		✓	1	0
App. II	✓	✓	✓	1	0
App. VII	✓			0	0
App. VIII			✓	2	2
App. XVII			✓	1	1
App. XVIII	✓			0	0
App. XIX	✓	✓		0	0

3) User's Registration And Log In Security

- 55%: asked for and transmitted users' passwords,
- 27%: do not use a secure connection (HTTPS)
- 45% of the apps that transmit users' passwords use GET requests => not a good security practice

Results- Dynamic Analysis cont'd

4) Email and device ID transmission

- 60% : share users' emails addresses with third parties.
- 89%: share users' device secure ID with third parties.

5) Users' search query privacy and OS type

- 25%: transmit users' search queries over the network,
  - only 20% of these apps use HTTPS
- all the apps: transmit search queries and send them to their vendor's domain
- 80%: send this information to third parties as well
  - 2 apps send their users' queries to 16 different third party domains!

Results- SSL Web Server Configuration

- Analyzed the web server configuration
  - To determine the security level of HTTPS data transmission.
  - Using a free online service SSL Labs from Qualys
- This online service:
  - enables the remote testing of web server's security
    - against a number of well-known vulnerabilities, such as Heartbleed or Drown
  - tests and rates the SSL web server configuration of each domain
    - with a letter grade scale (A, B, C, D, E, F, M, T)
  - Tests include:
    - the assessment of the certificate to verify that it is valid and trusted,
    - the inspection of the server configuration in three categories: a. protocol support, b. key exchange support and c. cipher support.

https://upload.wikimedia.org/wikipedia/commons/thumb/d/dc/Heartbleed.svg/220px-Heartbleed.svg.png
https://upload.wikimedia.org/wikipedia/commons/thumb/7/7a/DROWN_logo.svg/1200px-DROWN_logo.svg.png

Results- SSL Web Server Configuration cont’d

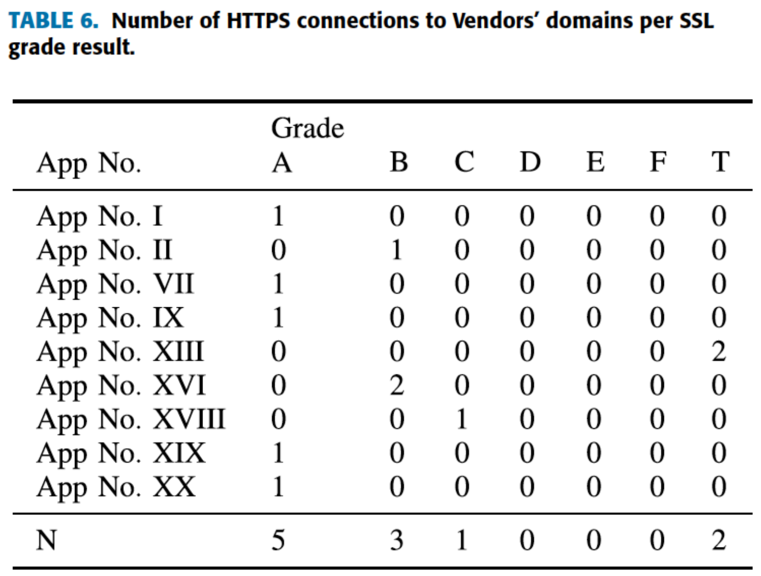


TABLE 6. Number of HTTPS connections to Vendors’ domains per SSL grade result.

App No.	Grade A	B	C	D	E	F	T
App No. I	1	0	0	0	0	0	0
App No. II	0	1	0	0	0	0	0
App No. VII	1	0	0	0	0	0	0
App No. IX	1	0	0	0	0	0	0
App No. XIII	0	0	0	0	0	0	2
App No. XVI	0	2	0	0	0	0	0
App No. XVIII	0	0	1	0	0	0	0
App No. XIX	1	0	0	0	0	0	0
App No. XX	1	0	0	0	0	0	0
N	5	3	1	0	0	0	2

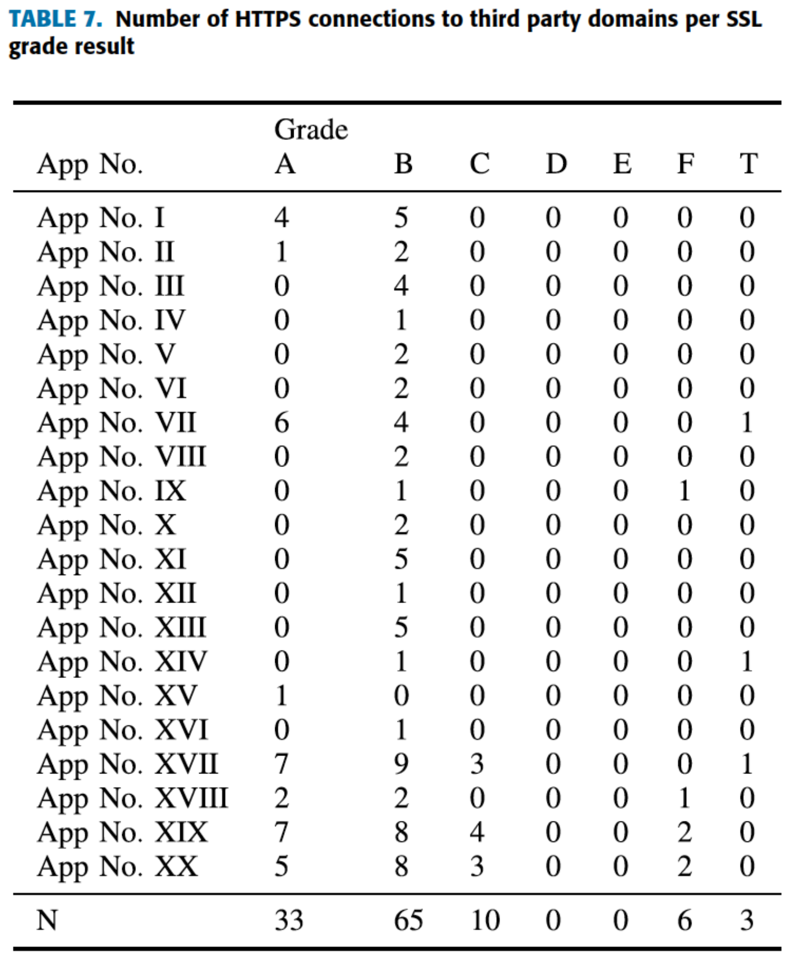


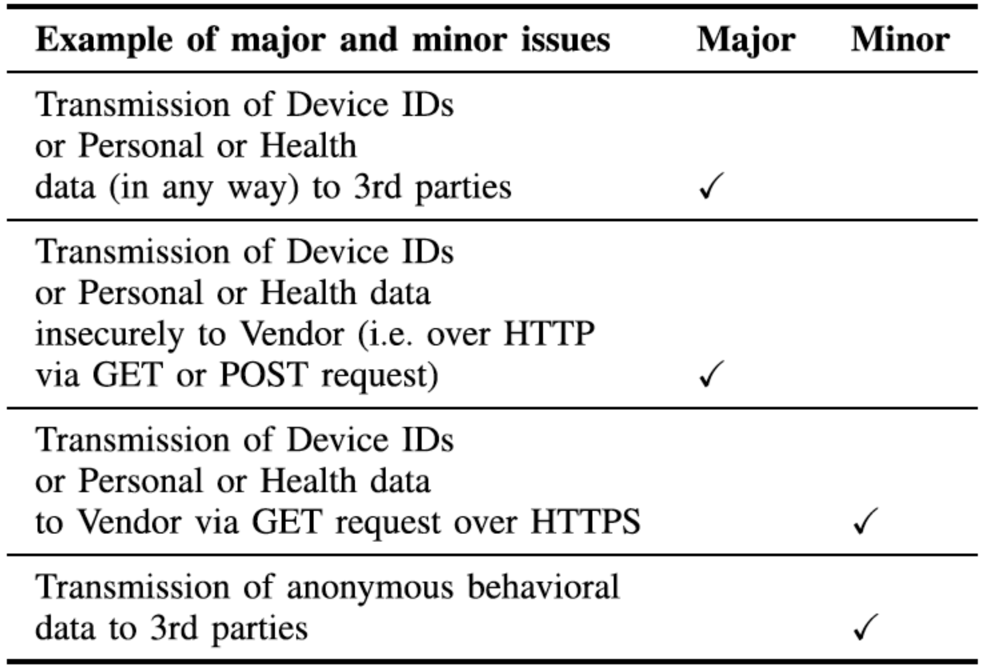
TABLE 7. Number of HTTPS connections to third party domains per SSL grade result

App No.	Grade A	B	C	D	E	F	T
App No. I	4	5	0	0	0	0	0
App No. II	1	2	0	0	0	0	0
App No. III	0	4	0	0	0	0	0
App No. IV	0	1	0	0	0	0	0
App No. V	0	2	0	0	0	0	0
App No. VI	0	2	0	0	0	0	0
App No. VII	6	4	0	0	0	0	1
App No. VIII	0	2	0	0	0	0	0
App No. IX	0	1	0	0	0	1	0
App No. X	0	2	0	0	0	0	0
App No. XI	0	5	0	0	0	0	0
App No. XII	0	1	0	0	0	0	0
App No. XIII	0	5	0	0	0	0	0
App No. XIV	0	1	0	0	0	0	1
App No. XV	1	0	0	0	0	0	0
App No. XVI	0	1	0	0	0	0	0
App No. XVII	7	9	3	0	0	0	1
App No. XVIII	2	2	0	0	0	1	0
App No. XIX	7	8	4	0	0	2	0
App No. XX	5	8	3	0	0	2	0
N	33	65	10	0	0	6	3

Results- Response to Security and Privacy Reporting

- Each app vendor is provided with a report of findings
- Reevaluation of the apps by re-running the dynamic analysis on their updated versions
- Results of reevaluation:
  - categorized as major & minor issues
  - analyze how many of them (in each category) remain in the re-evaluated version of the apps

TABLE 9. Example cases of major or minor issues.



Example of major and minor issues	Major	Minor
Transmission of Device IDs or Personal or Health data (in any way) to 3rd parties	✓	
Transmission of Device IDs or Personal or Health data insecurely to Vendor (i.e. over HTTP via GET or POST request)	✓	
Transmission of Device IDs or Personal or Health data to Vendor via GET request over HTTPS		✓
Transmission of anonymous behavioral data to 3rd parties		✓

Results- Response to Security and Privacy Reporting

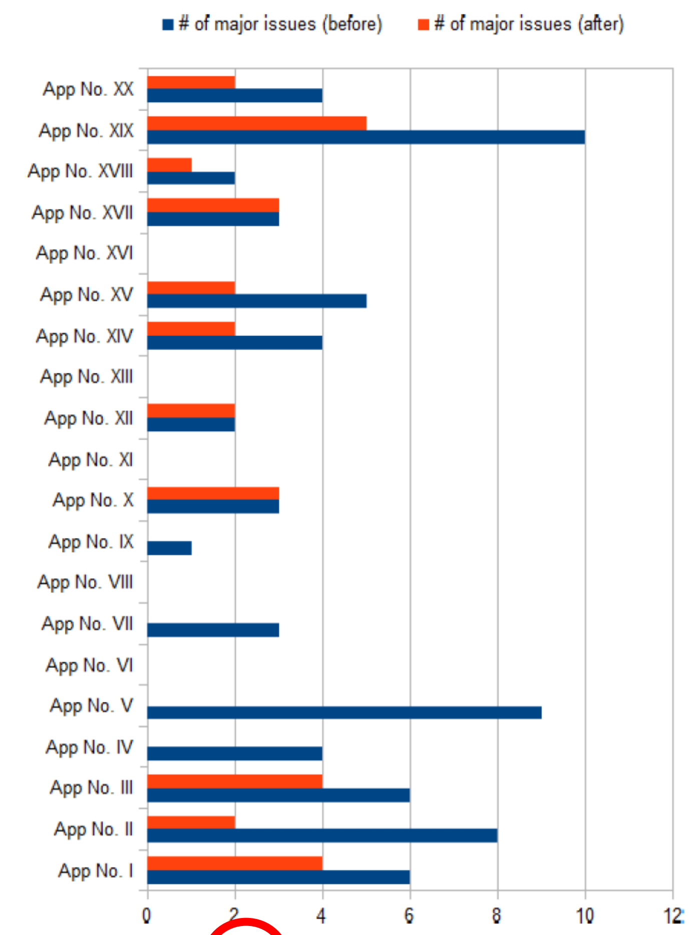


FIGURE 6. **Number of major issues per app before and after our reportings.**

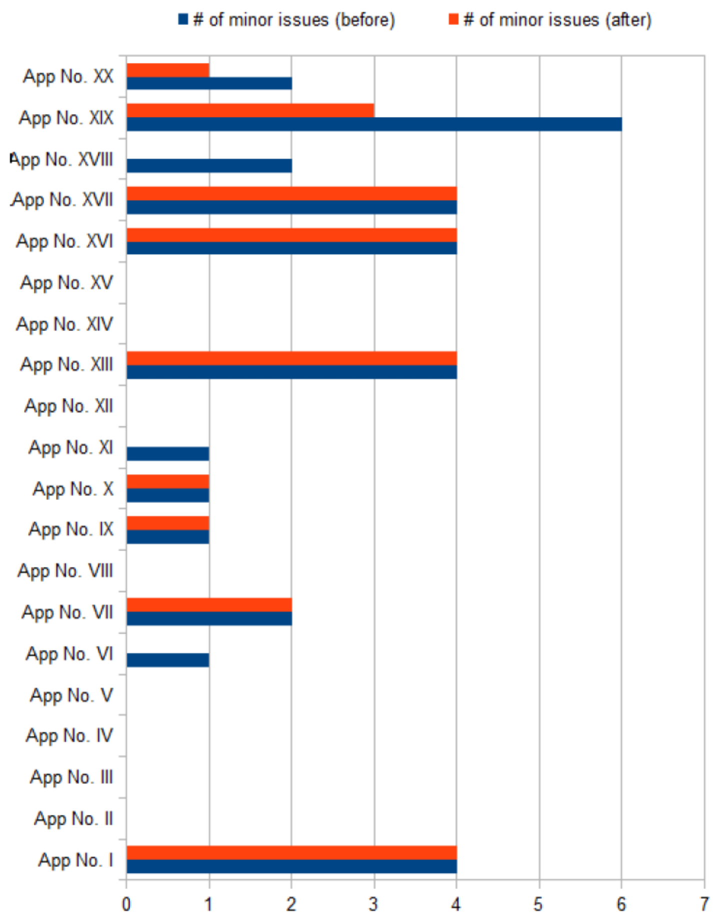


FIGURE 7. **Number of minor issues per app before and after our reportings.**

Results- GDPR Readiness Assessment

- Check whether the apps meet the legal data protection requirements specified in the GDPR's provisions.
- Findings divided into two categories:
  - “functional requirements’’
  - “non-functional requirements’’ of the GDPR

Results- GDPR Readiness Assessment cont’d

1) Functional Requirements

- Consent (I): provide information about privacy policy or/and term of use before registration
  - 11 out of the 19 apps
- Consent (II): ask for consent each time user provides additional information
  - Only one app
- Consent (III): require users to answer questions about their willingness to participate
  - None of the apps
- Right to withdraw consent: => allow for the erasure of any previously consented information
  - 7 out of the 19 apps
- Right to data portability: provide users with a mechanism to send their personal data to another entity
  - 7 out of the 19 apps

Results- GDPR Readiness Assessment cont’d

2) Non-functional Requirements

- Data Protection Officer: none of the apps provides any contact details for such a role.
- Profiling and marketing: 11 out of the 19 apps provide information on collection and processing of user data for profiling purposes
- Transfer to third parties: 8 out of the 19 apps notify their users in advance

Recent Approaches to PPM (Privacy Protection Mechanisms) for mHealth

CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring (Lin et al., 2013)

Motivation

- Problem addressed: client privacy and intellectual property of health monitoring service providers
- Traditional privacy protection mechanisms:
  - removing clients' personal identity information (such as names or SSN)
  - using anonymization techniques such as k-anonymity or l-diversity
- Fail to serve as an effective way in dealing with privacy of mHealth systems
  - due to the increasing amount and diversity of personal identifiable information
- These techniques might be insufficient to prevent reidentification attack!

Motivation

- Design a cloud-assisted privacy preserving mobile health monitoring system (CAM)
- To relieve the computational complexity on the company’s side:
    - proposed a new variant of proxy reencryption technique:
        - the company only needs to accomplish encryption once at the setup phase
        - shifting the rest computational tasks to the cloud
- To reduce clients’ decryption complexity:
    - Adapt the “outsourcing decryption” technique (for clients):
        - either clients’ query input or the decrypted decision is not revealed to the cloud

System Model

- Proposed mHealth monitoring program (CAM) is built upon branching program
- Use of the monitoring program introduced in the Microsoft MediNet project to construct a branching program as shown in Fig. 1.

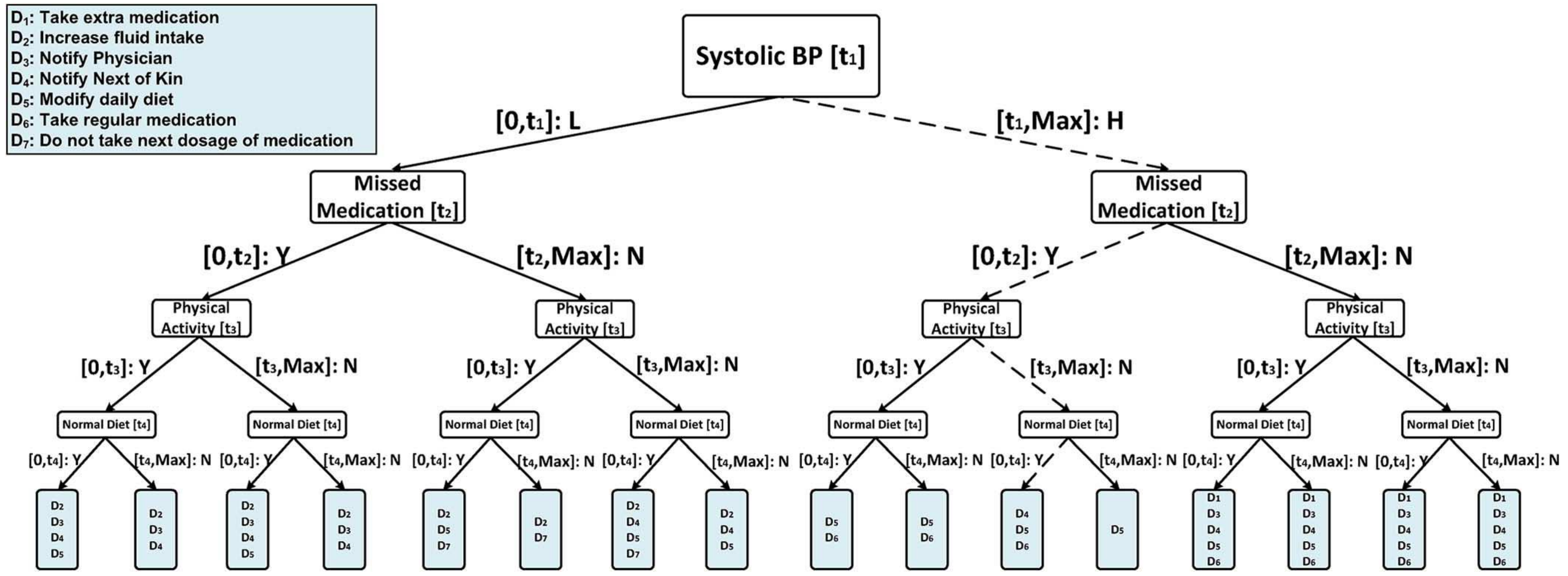


Fig. 1. Branching program in MediNet project.

Input: (from client) systolic blood pressure, missed daily medication or not, have an abnormal diet, energy consumption of physical activity

Output: (from decision sup sys) recommendation on how clients can improve their conditions

System Model cont’d

- Example attribute vector input by a hypertension patient:

[Systolic BP: 150, Missed medication: 1, Energy Expenditure: 900 kcal, salt intake: 1000 milligrams]”

- Respective thresholds: t1=130, t2=0, t3=700, t4=1500 **=> What is recommendation?**

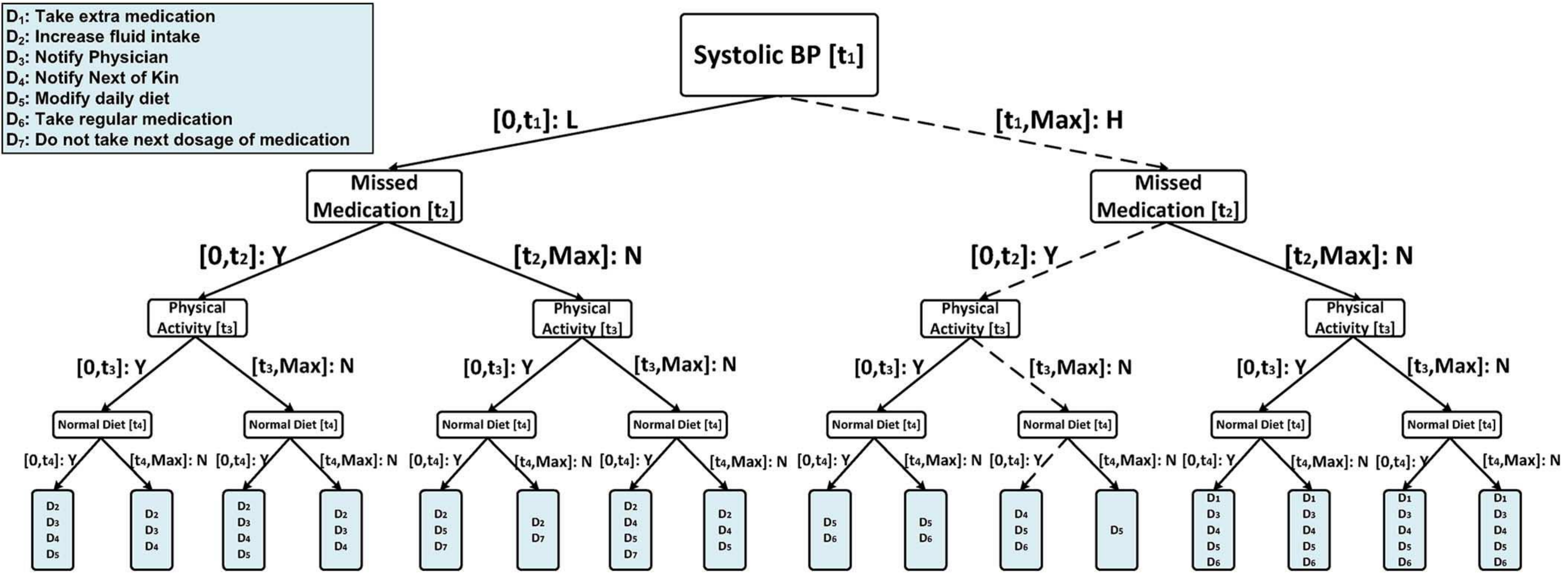


Fig. 1. Branching program in MediNet project.

System Model cont’d

- Idea: A monitoring program can be designed as a binary decision tree:
  - based on ranges of monitored measurements
  - represents measured data as an attribute vector: $\mathrm{v} = (\mathrm{v}_1, \ldots, \mathrm{v}_n)$
  - construct the binary branching tree with the leaf nodes as the final consultation to design the medical decision support system.
  - consists of
    - A nonleaf node is called a “decision node”,
    - leaf node is called a “label node”-> attached with classification information

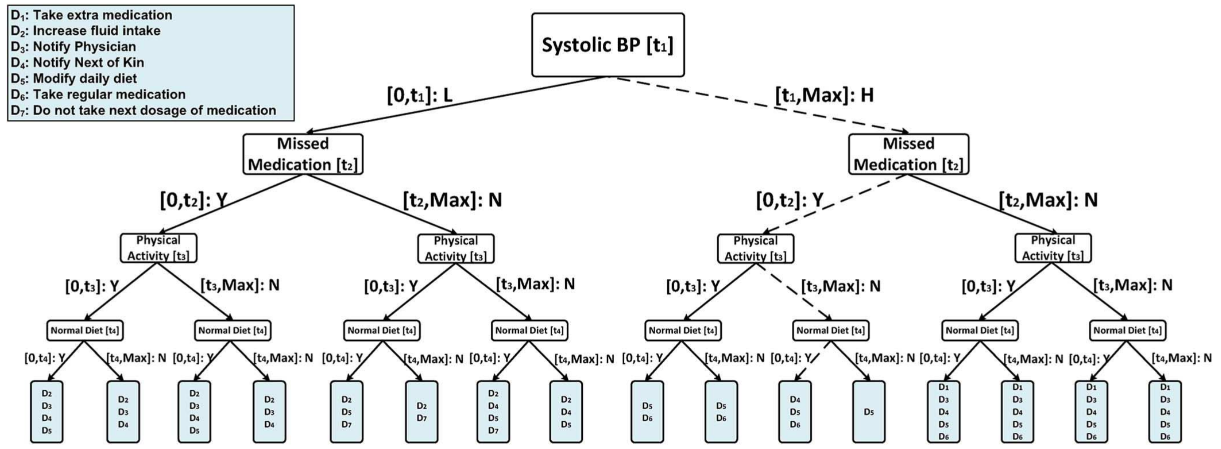


Fig. 1. Branching program in MediNet project.

System Model– Design of CAM

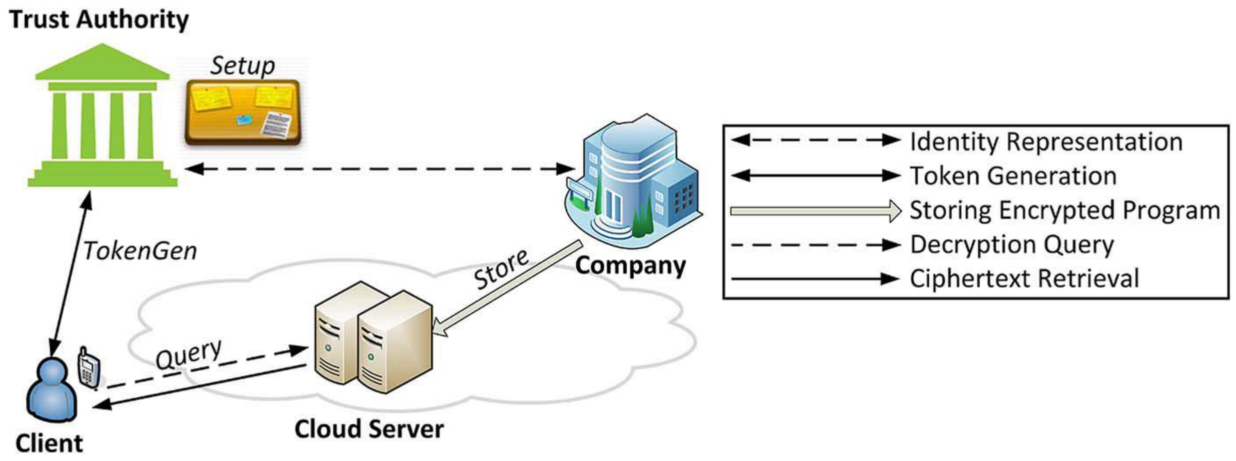
- CAM consists of four parties:

a) the cloud server

b) the company which provides the mHealth monitoring service (i.e., the healthcare service provider)

c) the individual clients and

d) a semi trust authority (TA)



- The company stores its encrypted monitoring data or branching program in the cloud.
- Individual clients collect their medical data and store them in their mobile devices, which then transform the data into attribute vectors.
- The attribute vectors are delivered as inputs to the monitoring program in the cloud.
- TA is responsible for distributing private keys to clients and collecting service fees from clients according to a certain business model such as “pay-per-use” model.

System Model– Design of CAM

Major Steps of CAM

Setup: Initial phase, TA runs the phase and publishes the system parameters.

Store: i) mHealth monitoring program (branch program) is generated and encrypted by the company.

ii) Encrypted text and company index are sent to the cloud

TokenGen: i)Client sends the company index to TA and poses a private query (attribute vector representing the collected health data) for a certain mHealth monitoring program.

ii) TA generates a token with master secret as a reply to client query.

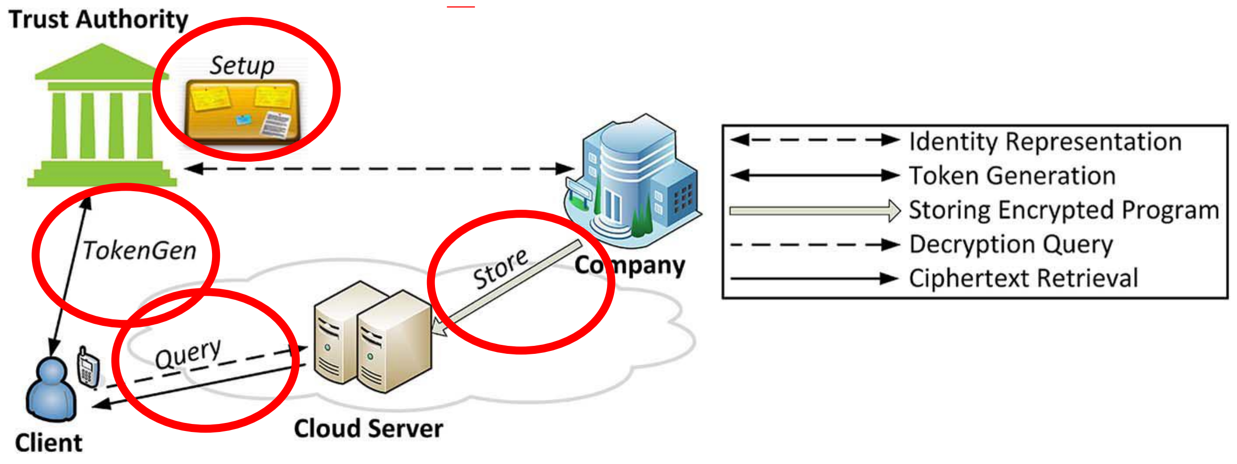
Query: Cloud runs this phase by:

i)Taking client token for the query

ii) Completing computationally intensive decryption task

iii) Return partially decrypted text to the client

iv) Client then decrypts the text completely to obtain the decision returned by monitoring program



Adversarial Model

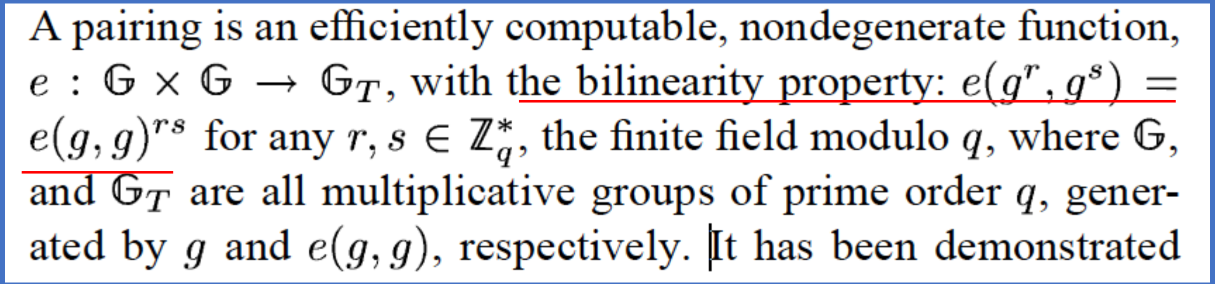
- A neutral cloud server:
    - neither colludes with the company nor a client to attack the other
- CAM design assumes an honest but curious model:
    - all parties should follow the prescribed operations and cannot behave arbitrarily malicious
- Also consider insider attack
    - could be launched by either malicious or nonmalicious insiders who behave normally,
    - but intend to discover information about the others’ information.

Proposed Solution

- Attribute-based cryptographic te
  - Rationale: querying input to a
- Cryptographic building blocks of
  1) Variants of Boneh-Franklin
     - First IBE scheme
     - based on Bilinear pairing
  2) Homomorphic encryption
     - a semantically secure *additively homomorphic* public-key encryption technique

$$\mathtt{HEnc}(m_1 + m_2) = \mathtt{HEnc}(m_1) \star \mathtt{HEnc}(m_2)$$

  3) Multidimensional Range Query Based on Anonymous IBE (MDRQ)
  4) Decryption outsourcing
  5) Key private proxy reencryption (PRE)



A pairing is an efficiently computable, nondegenerate function, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, with the bilinearity property: $e(g^r, g^s) = e(g, g)^{rs}$ for any $r, s \in \mathbb{Z}_q^*$, the finite field modulo q, where $\mathbb{G}$, and $\mathbb{G}_T$ are all multiplicative groups of prime order q, generated by g and $e(g, g)$, respectively. It has been demonstrated

Key contributions

Proposed Solution

Decryption Outsourcing:

- The pairing-based IBE system has high cost of computation:
  - due to the bilinear pairing computation in the decryption steps
  - Especially costly for resource-constrained mobile devices!

- Decryption outsourcing to ease the computational complexity
  - Client transforms his secret key to the transformation key
  - An untrusted server uses it to transform the original ciphertext into an El Gamal encrypted ciphertext
  - Client only needs to compute simple exponentiation operations to obtain the underlying message

Proposed Solution

Key Private Proxy Reencryption (PRE) :

- Used for decryption outsourcing purpose
- allows an untrusted proxy server with a reencryption key (rekey) $rk_{A \rightarrow B}$ to transform a first level ciphertext encrypted for A (delegator) into the second level ciphertext that could be decrypted by B (delegatee)
- Key private property:
    - guarantees that no useful information about the underlying identities is leaked to the cloud
- In CAM:
    - mHealth monitoring program is encrypted by the provider company using an MDRQ scheme
    - The ciphertext is stored in the untrusted cloud.
    - The company delivers several reencryption keys to the cloud.

Proposed Solution

Key Private Proxy Reencryption (PRE) :

- Proposed approach: A new ID-based key private proxy reencryption scheme:
  - With lower cost of rekey generation comparing with the original encryption algorithm
  - Proposed rekey generation algorithm is run by TA rather than the company.
    - In traditional identity-based PRE systems, it is run by the company!

Proposed Solution

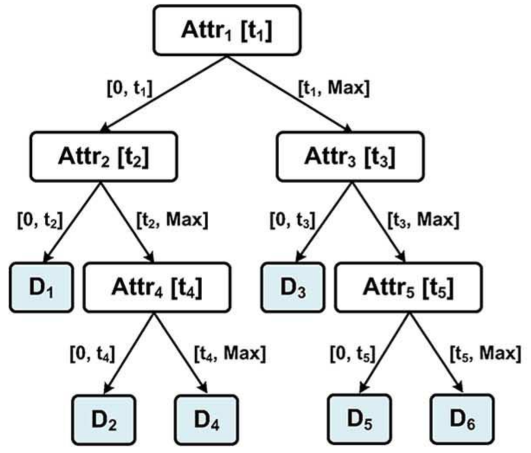
MDRQ (Multidimensional Range Query) Scheme:

- To reduce clients’ decryption complexity,
  - incorporate the recently proposed outsourcing decryption technique [25] into the MDRQ system
  - shifts clients’ computational complexity to the cloud
  - without revealing any information on either clients’ query input or the decrypted decision to the cloud.
- In MDRQ,
  - a sender encrypts a message under a range $[r_1, r_2]$ (or a range of -bit block)
  - a receiver with private keys falling into this range can decrypt the underlying message
  - can guarantee the privacy of both encrypted message and respective range

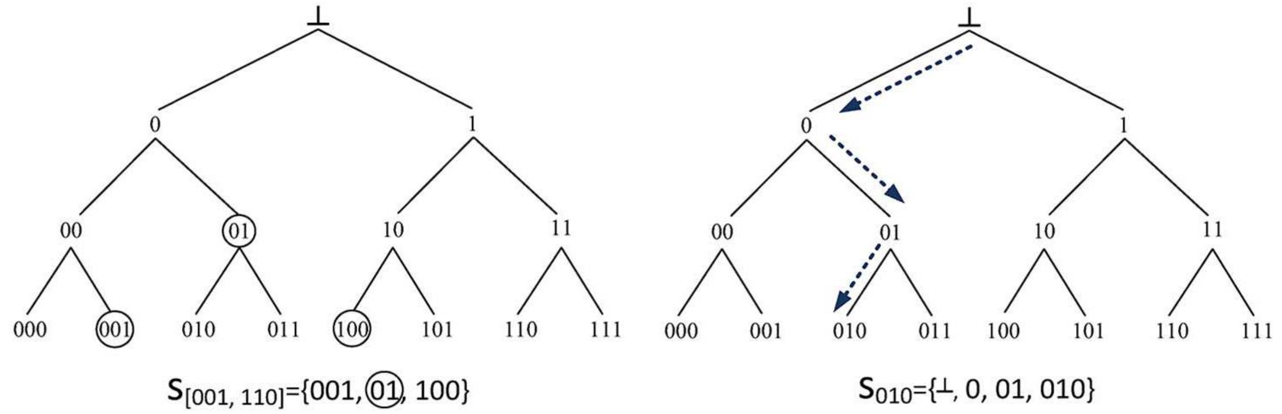
Proposed Solution

Multidimensional Range Query Based on Anonymous IBE (MDRQ)

Binary decision tree representation of an mHealth monitoring program



Binary bit represented tree



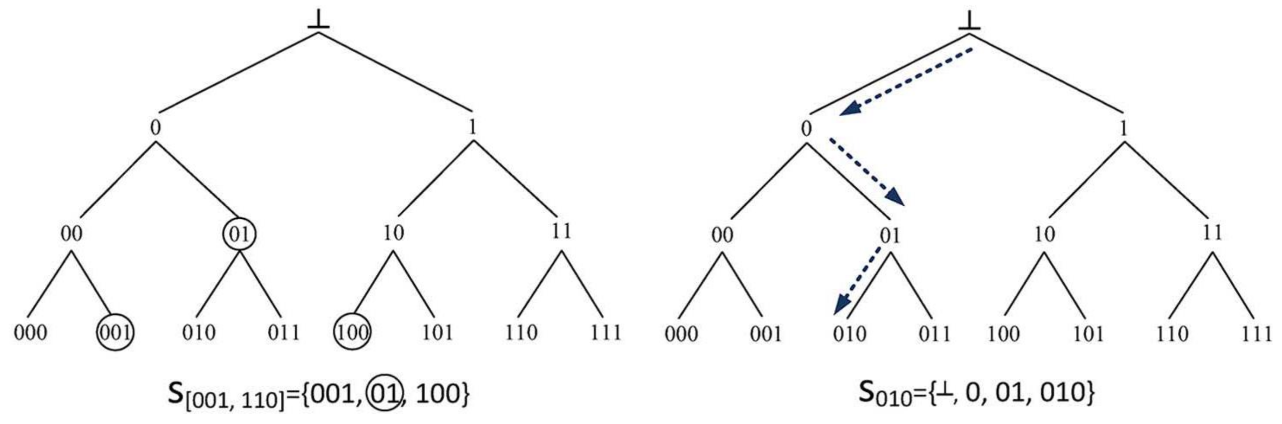
Attribute vector space: $v = (v_1, \dots, v_n)$

Binary bit block: $\{001, 01, 100\}$

Proposed Solution

Multidimensional Range Query Based on Anonymous IBE (MDRQ)

- The basic idea of MDRQ is as follows:
  - a C-level binary tree is employed to represent the C-bit data (or the range).
  - The root of this binary tree is labeled as $\perp$.
  - The left child node of a nonleaf node p is labeled as $p0$ and the right child node is labeled as $p1$.
  - As a result, all the leaves from left to right will be labeled with a binary string from $0, \ldots, 0$ to $1, \ldots, 1$.

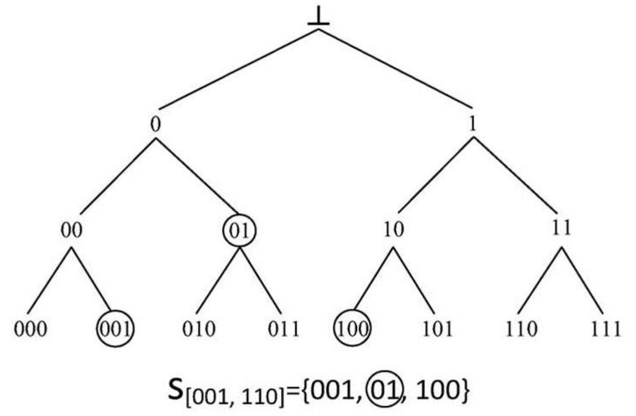


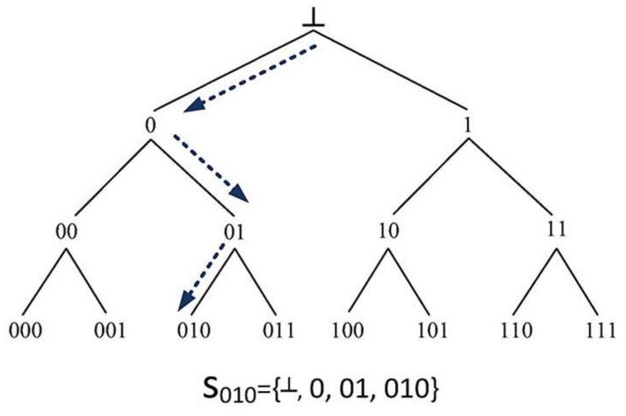
(b)

Proposed Solution

Multidimensional Range Query Based on Anonymous IBE (MDRQ)

- The basic idea of MDRQ is as follows:
  - To represent a range $[r_1, r_2] \subseteq [0, 2^C - 1]$, a minimum set of roots of subtrees covering all the leaf nodes in this range is used.
    - minimum root set to represent a range [001, 100] is $S_{[001,100]} = \{001, 01, 100\}$
  - To represent a C-bit data v, first find the respective leaf node, then use the collection of all nodes on the path from the root to this leaf node.
    - the collection $S_{010} = \{\perp, 0, 01, 010\}$ represents 010.

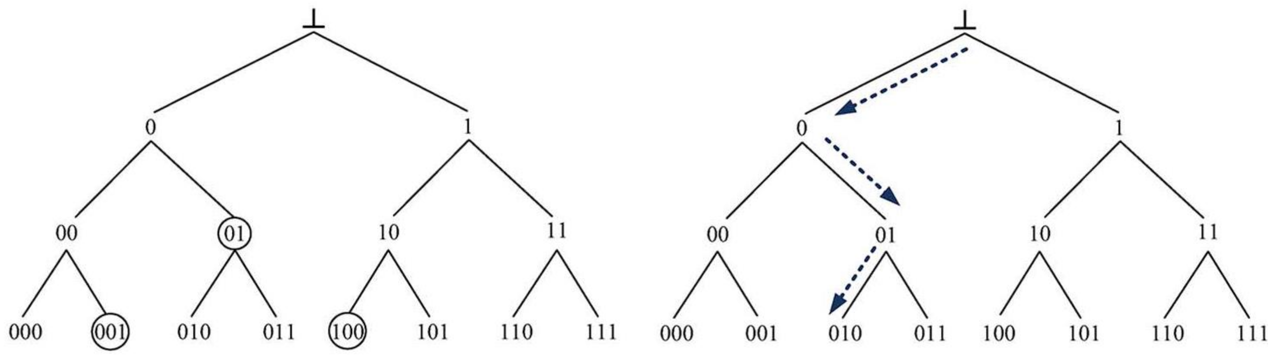




Proposed Solution

Multidimensional Range Query Based on Anonymous IBE (MDRQ)

- The basic idea of MDRQ is as follows:
    - How to test whether 010 belongs to the interval [001, 100] ?



$S_{[001,\ 110]}=\{001, 01, 100\}$ $S_{010}=\{\perp, 0, 01, 010\}$

Proposed Solution

MDRQ (Multidimensional Range Query) Scheme:

- Anonymous identity-based encryption (A-IBE) is used to construct MDRQ
  - Preserves the privacy of both the receiver identity and the underlying message
  - The traditional IBE scheme can only preserve the privacy of an underlying message
- Sender encrypts a message m using all identities in a range $[r_1, r_2]$ (or a vector):
  - Treats each element in $S_{[r_1,r_2]}$(or S_v) as an identity in the identity space in the A-IBE scheme
- Receiver can decrypt message:
  - only if the attribute values of the receiver falls into the range $[\mathrm{r}_1, \mathrm{r}_2]$
  - By obtaining private keys corresponding to all identities in $\mathrm{S}_{[\mathrm{r}_1,\mathrm{r}_2]}$ from TA

Improvement on the Basic CAM Design

- Basic CAM has security weaknesses:
  - TA knows the identity representation set for a client’s attribute vector and can easily infer the **client’s** private attribute vector.
  - The cloud can find out identity representation for the private key of **the client** by running identity test in MDRQ => attribute vector revealed
  - The cloud can find out the **company’s** branching program since it has the private keys of all the system users

Improvement on the Basic CAM Design

- Two improvement suggestions:
  - First-level improvement on security:
    - **Client:** obliviously submit attribute vectors to **TA** and obtain the respective private keys without letting TA get any useful information on the private vector.
    - **Client**: runs the outsourcing decryption of MDRQ to ensure **the cloud** completes the major workload while obtaining no useful information on his private keys.
    - **Company**: permute and randomize its data using homomorphic encryption and MDRQ so that **neither the cloud nor a client** can get any useful information on its private information on branching program after a single query.

Improvement on the Basic CAM Design

- Two improvement suggestions:
  - Further improvement on performance:
    - Firs-level improvement meet desired security requirements
    - Huge computational overhead for the company to compute all ciphertexts for each of clients
    - Proposed improvement: The company generates one single ciphertext
    - The company obliviously delivers the identity representations and indexes of the attributes to TA
    - TA can generate the rekeys corresponding to the rest clients in the system
    - Generated keys are sent to cloud and cloud generates the ciphertexts for the rest clients.

Evaluation-Efficiency

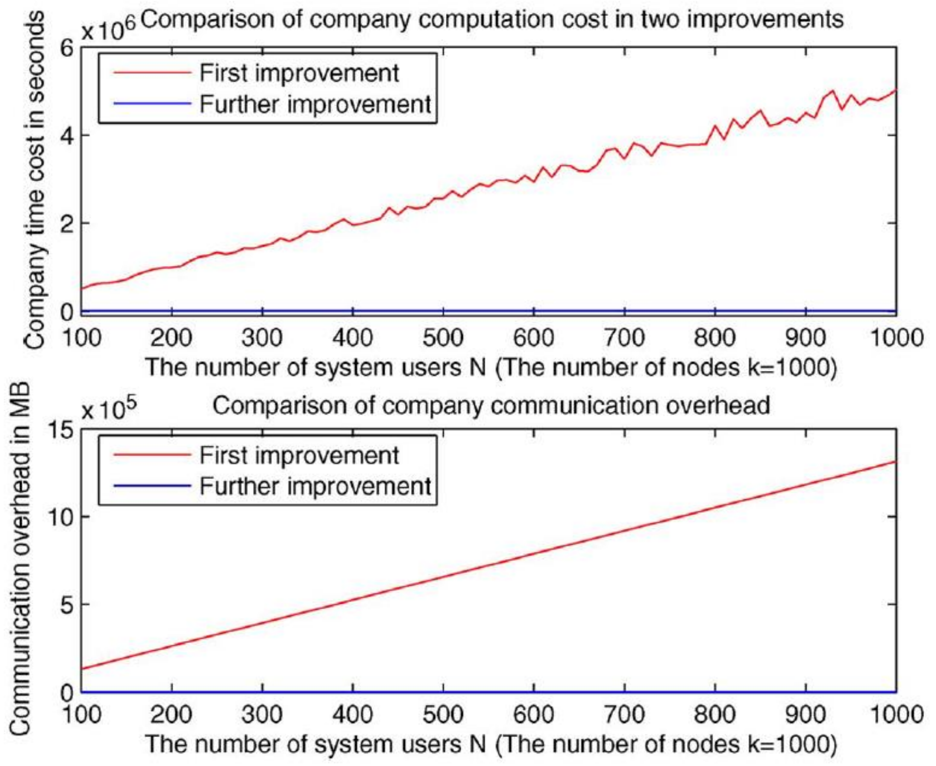


Fig. 5. Comparison of company computation and communication overheads in our two improved CAM designs.

Evaluation-Efficiency

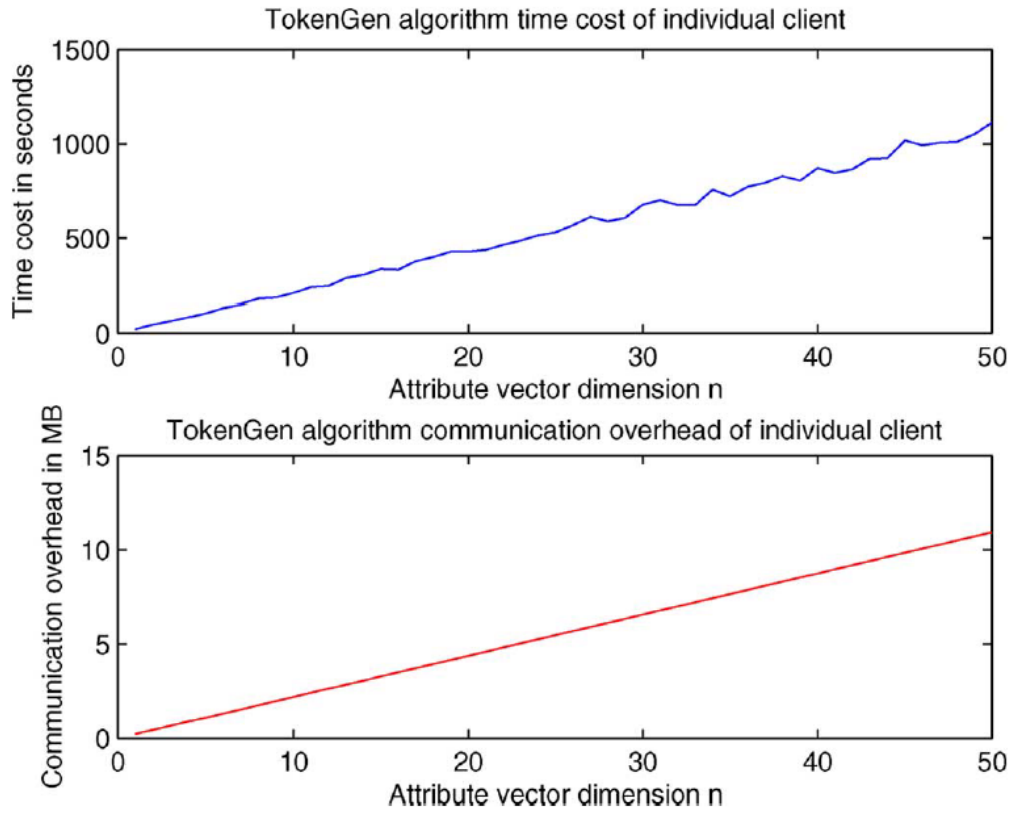


Fig. 6. Workload of individual token generation.

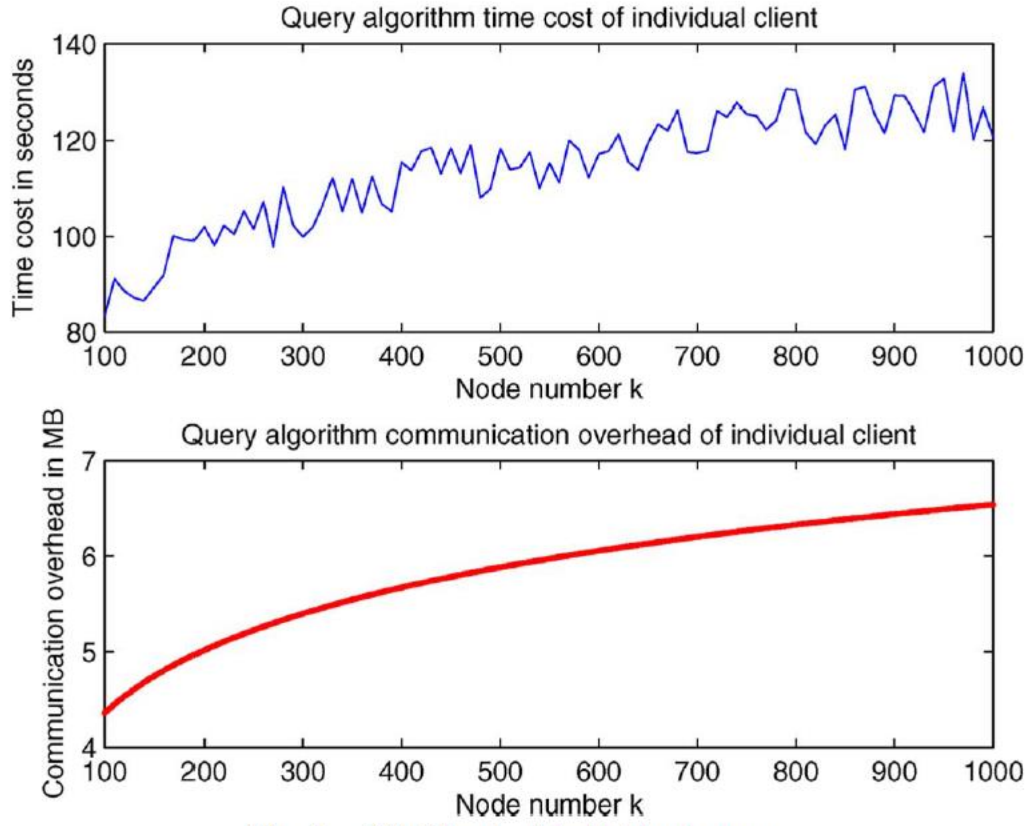


Fig. 7. Workload of individual query.

References

- Knorr, K., & Aspinall, D. (2015, April). Security testing for Android mHealth apps. In *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on* (pp. 1-8). IEEE.
- Papageorgiou, A., Strigkos, M., Politou, E., Alepis, E., Solanas, A., & Patsakis, C. (2018). Security and privacy analysis of mobile health applications: The alarming state of practice. *IEEE Access*, *6*, 9390-9403.