Distributed Medical Data Gathering in Remote Locations

Samson Gejibo

PhD Candidate
University of Bergen | Department of Informatics
Samson.Gejibo@ii.uib.no

Supervisors

Khalid Mughal

Federico Mancini

University of Bergen | Department of Informatics

Leading Competitors in mHealth (Developed Nations)

m-Health has the potential to revolutionize affordable healthcare delivery.

Topics of this presentation

- Overview of Mobile Data Collection
- Use Case
- Android Client – Security Review
- Secure Communication
- Secure Cloud Storage
- Conclusions

Mobile Health System (mHealth)

Definition: Delivery of healthcare services via mobile communication devices.

Opportunity: By 2017, more mobile phones than people on the planet; currently three-quarters of the world's population have access to a mobile phone.

Goal: Facilitate medical and health information via instantaneous communication anywhere/anytime.

Healthcare Services: Disease outbreaks tracking, Diagnostic and treatment support, Communication and Training for healthcare workers, education and Awareness, Remote data collection, personalize medical advice in chronic diseases.

Mobile Data Collection System (MDC)

- Aims to replace paper based data collection with electronic data collection system.

✓ Better data quality

✓ Low-resource settings

✓ Fast decision making process

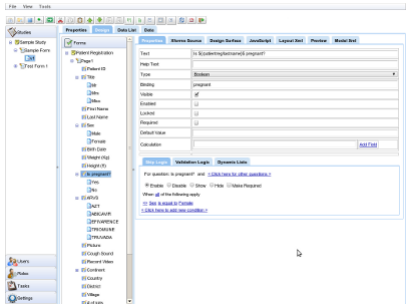
✓ Secure

Mobile Data Collection (MDC)

MDC – Data Flow

DATA MANAGER

DESIGN AND UPLOAD
A FORM DEFINITION



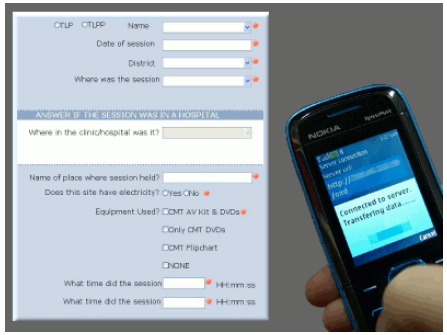
PROJECT
SERVER

ANALYSE AND
VIEW DATA

DATA VIEWER

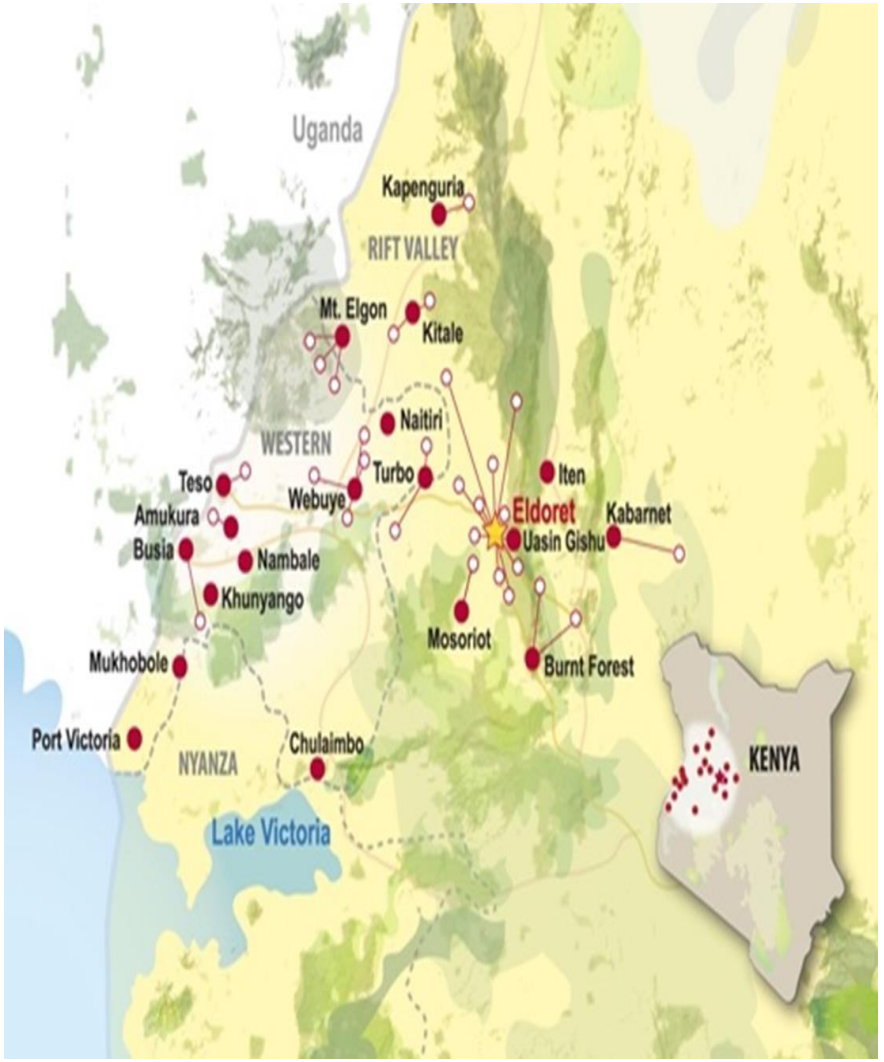
DOWNLOAD FORM
DEFINITION

UPLOAD FILLED
FORM



USE CASE

- The partnership between USAID-AMPATH
- One of the largest HIV treatment programs in sub-Saharan Africa
- Catchment area has some 3 million people and the program provides care to more than 130,000 active HIV patients through 26 parent and 26 satellite clinics.



USE CASE – Data Flow

Patient Registration

- When a patient comes into a clinic, the clinician fills out a highly structured paper form about the visit.
- The form goes into the chart.

USE CASE – Data Flow

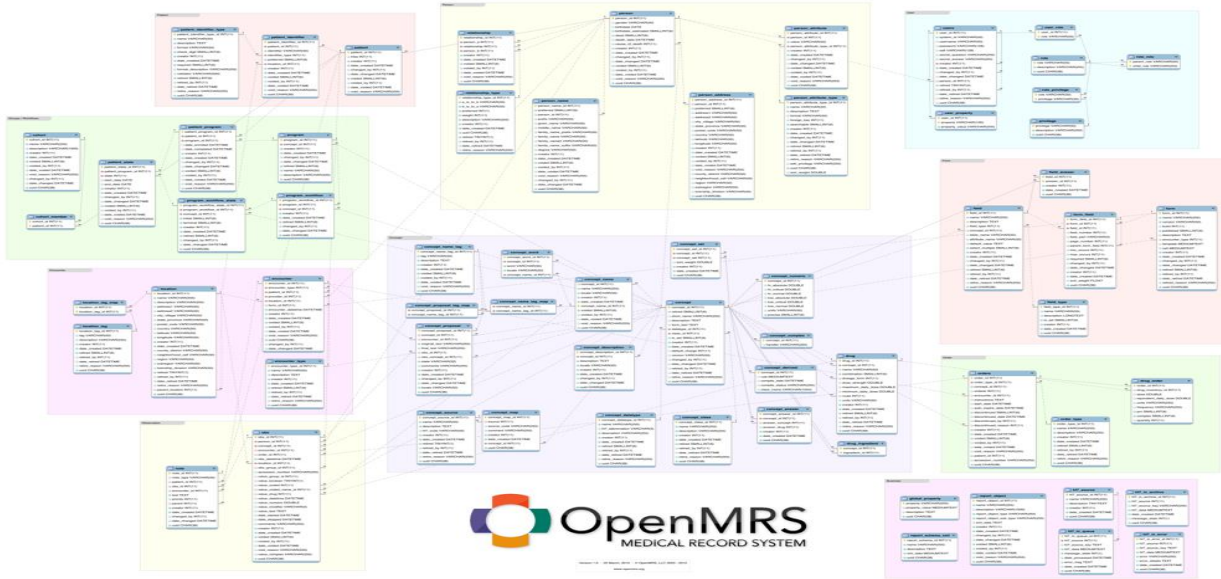
Data Entry into an EMR

- Data clerk with minimal computer skills and little medical knowledge enters all visit data from the encounter forms into the medical record system.

USE CASE – Data Flow

Patient Data Storing

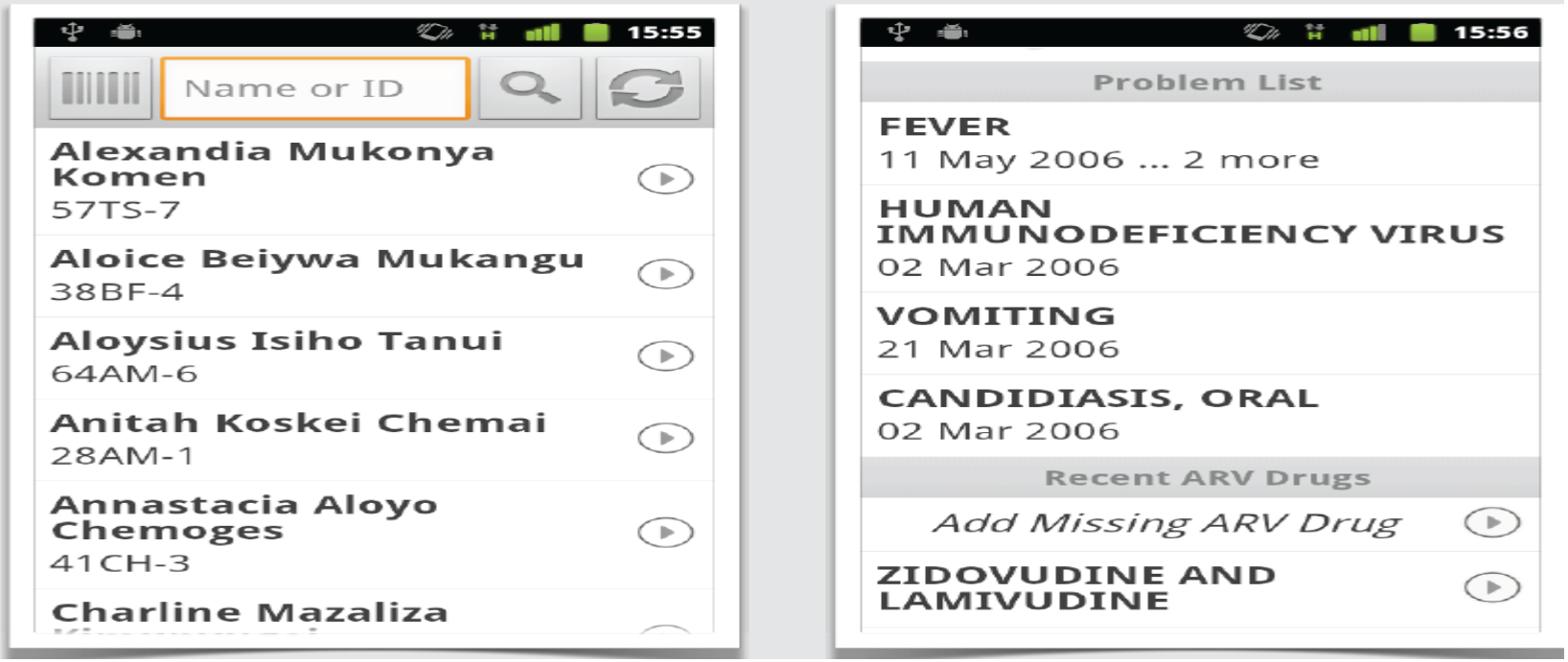
- The data is stored in openMRS, an open-source electronic medical record system
- Satellite servers in several locations are synchronized to central to get access to system data (either over a network connection or using USB).



USE CASE – Data Flow

Patient Clinical Summary

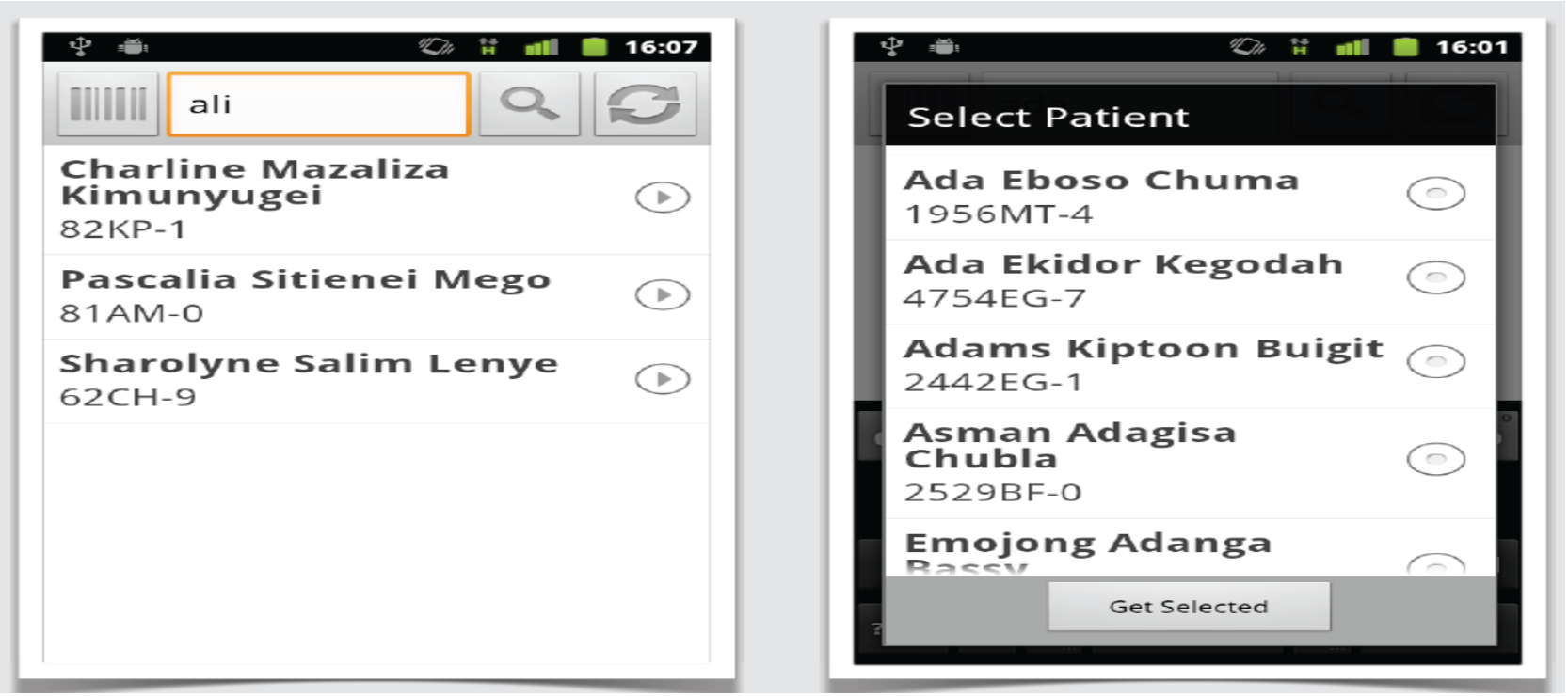
- Gives a nice overview of all the relevant patient data
- May have a reminder about the lab ordering schedule e.g. CD4 test, start/stop medications, referrals,.



USE CASE – Data Flow

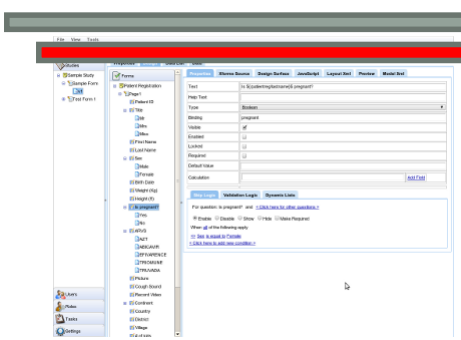
Local search and remote search

- Supports live searches against patient name and id
- if the patient isn't on the phone, it checks against all 130k patients on the server and download the record.



MDC – General Security Concerns

DESIGN AND UPLOAD
A FORM DEFINITION



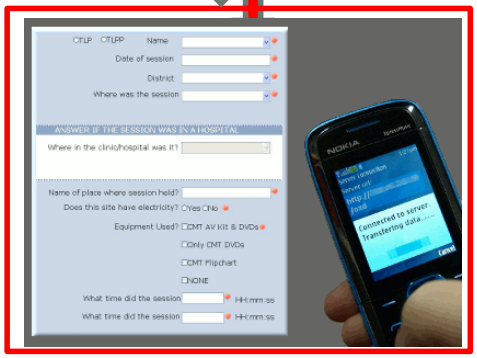
PROJECT
SERVER

ANALYSE AND
VIEW DATA

DATA VIEWER

DOWNLOAD FORM
DEFINITION

UPLOAD FILLED
FORM



- **ACCESS CONTROL**
- **MUTUAL AUTHENTICATION**
- **CONFIDENTIALITY**

- **DATA CONFIDENTIALITY**
- **DATA INTEGRITY**
- **LOCAL AUTHENTICATION**
- **LOCAL ACCESS CONTROL**

Why Android? – Partner Projects

OpenXdata

- Open-source , community supported
- Provide Java based Server side app and J2ME Client side app
- Large scale deployment in India, Pakistan, South Africa, Uganda, Guatemala,…
- First started at the University of Bergen

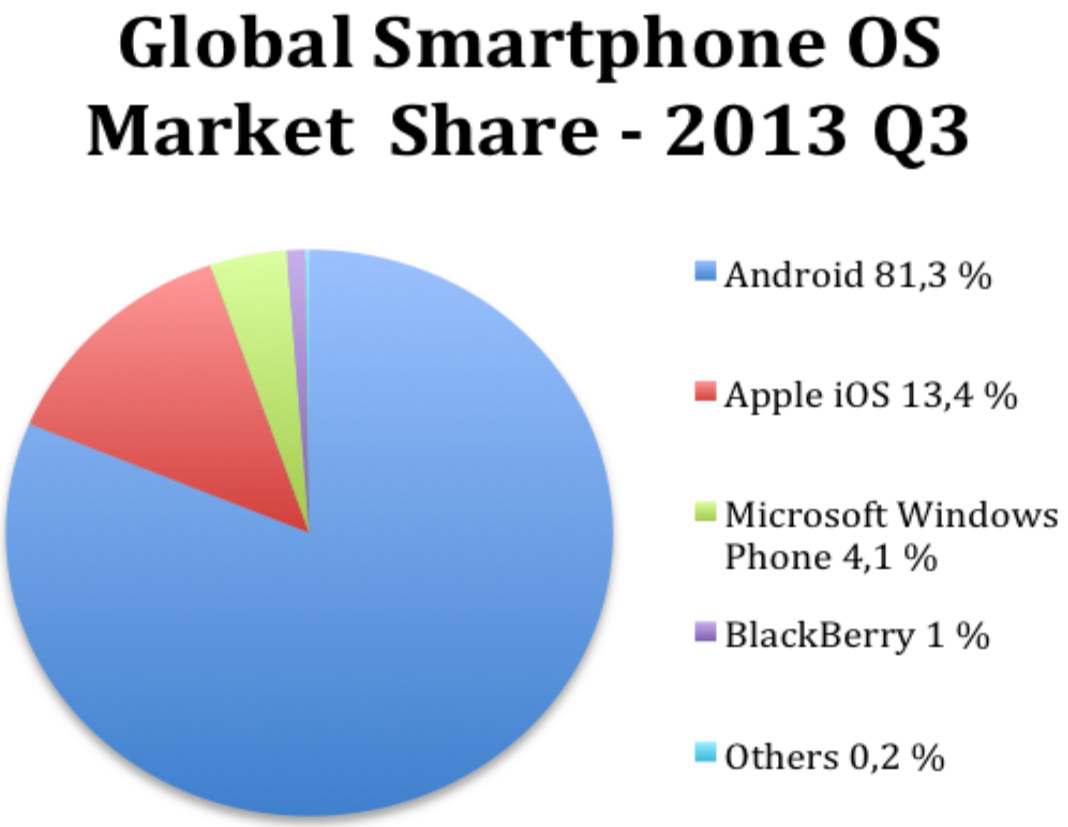
Open Data Kit

- Open-source, community supported
- Provide Java Cloud based Server app and Native Android client app
- Large scale deployment under Unicef, Millennium Development Projects, India, Kenya, South Africa, Peru, China,…
- First started at the University of Washington, Seattle

mUzima

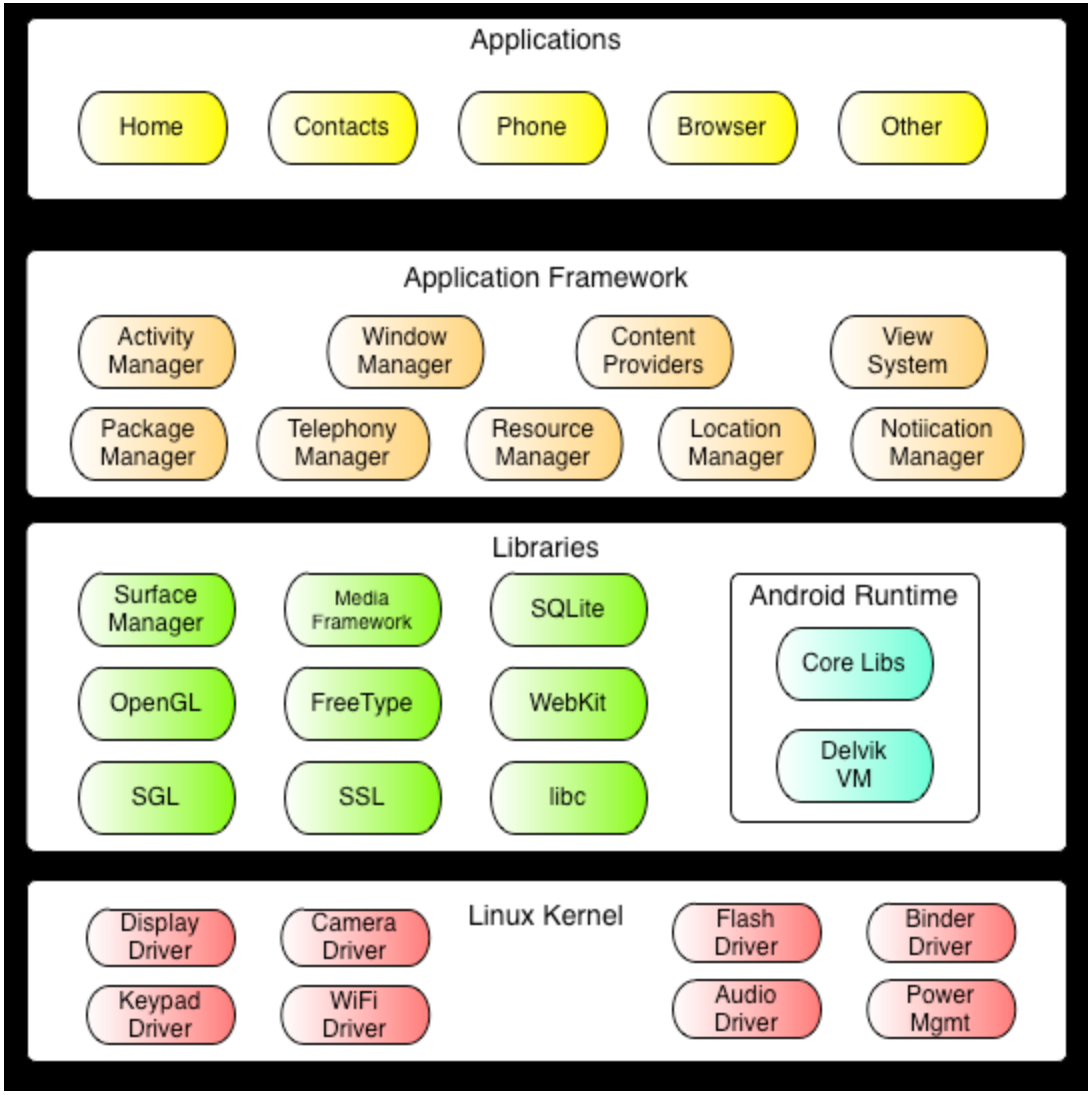
- Licensing not decided yet
- Provide Hybrid Android Client that works openMRS server
- Maintained by AMPATH (Moi University and Indiana University)
- Large scale deployment under AMPATH in Kenya
- First started at AMPATH

Why Android? Market Share



Source: Gartner

Android stack



Linux kernel

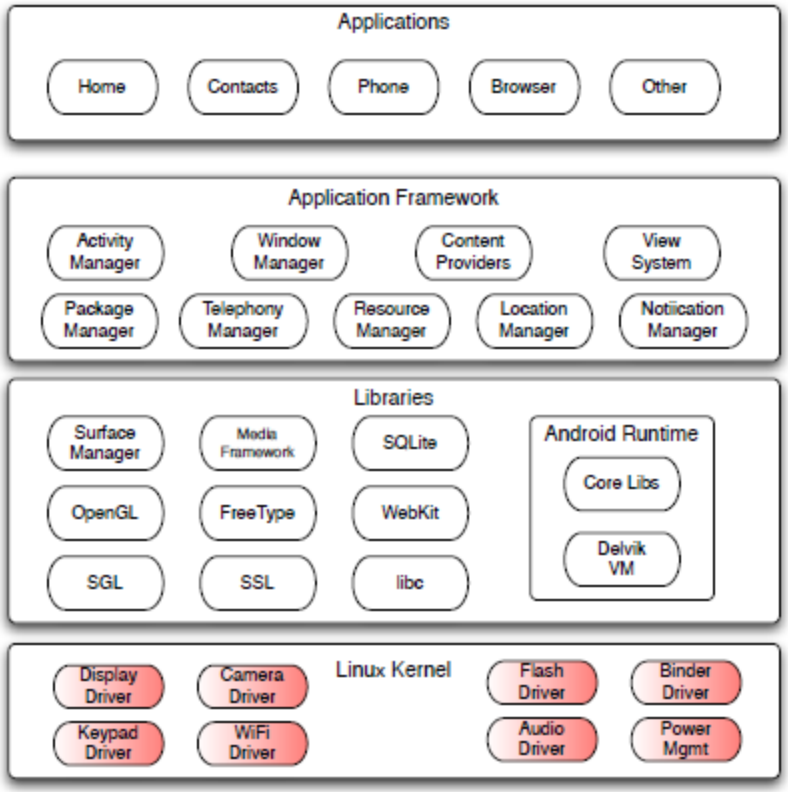
Android runs on Linux.

Linux provides:

- Hardware abstraction layer
- Memory management
- Process management
- Networking

Users never see Linux sub system

The adb shell command opens Linux shell



Native libraries

Pieces borrowed from other open source projects:

Bionic, a super fast and small license-friendly libc library optimized for Android

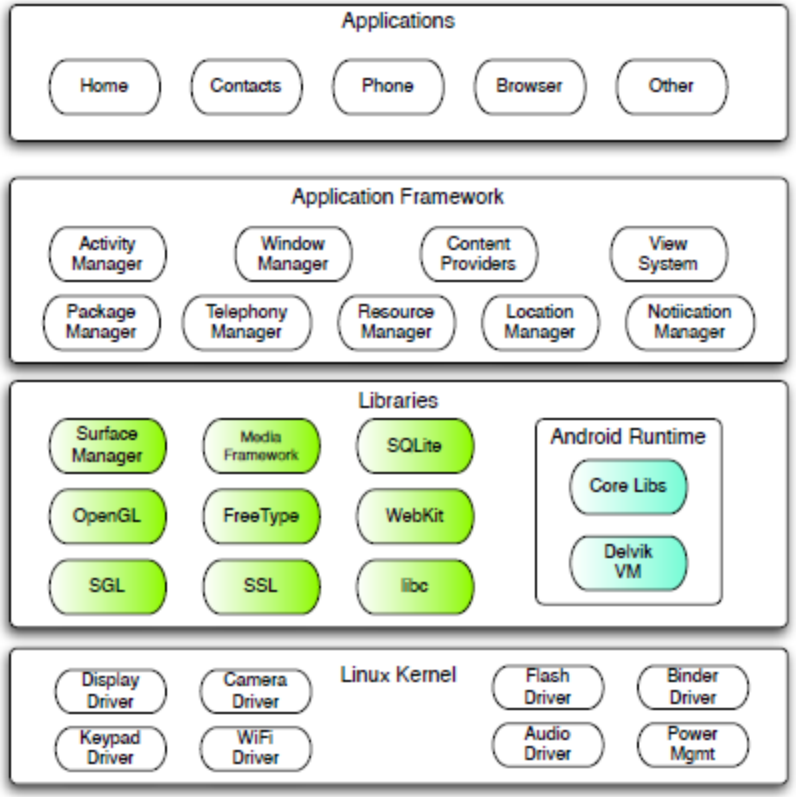
WebKit library for fast HTML rendering

OpenGL for graphics

Media codecs offer support for major audio/video codecs

SQLite database

Much more…



Dalvik

Dalvik VM is Android implementation of Java VM

Dalvik is optimized for mobile devices:

- Battery consumption
- CPU capabilities

Key Dalvik differences:

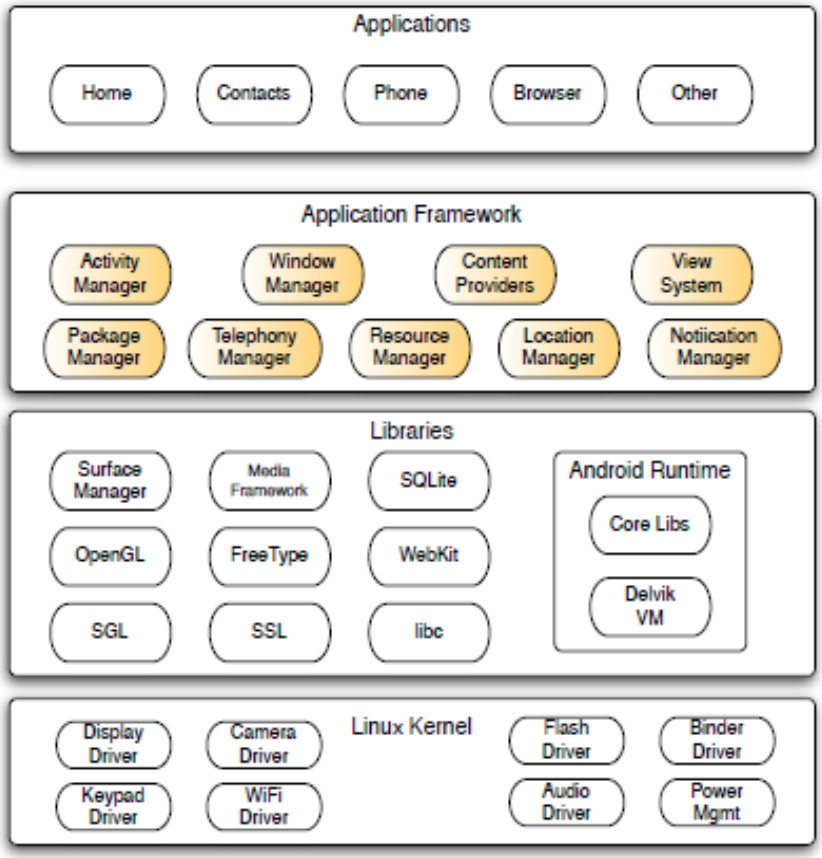
- Register-based versus stack-based VM
- Dalvik runs .dex files
- More efficient and compact implementation
- Different set of Java libraries than JDK

Application framework

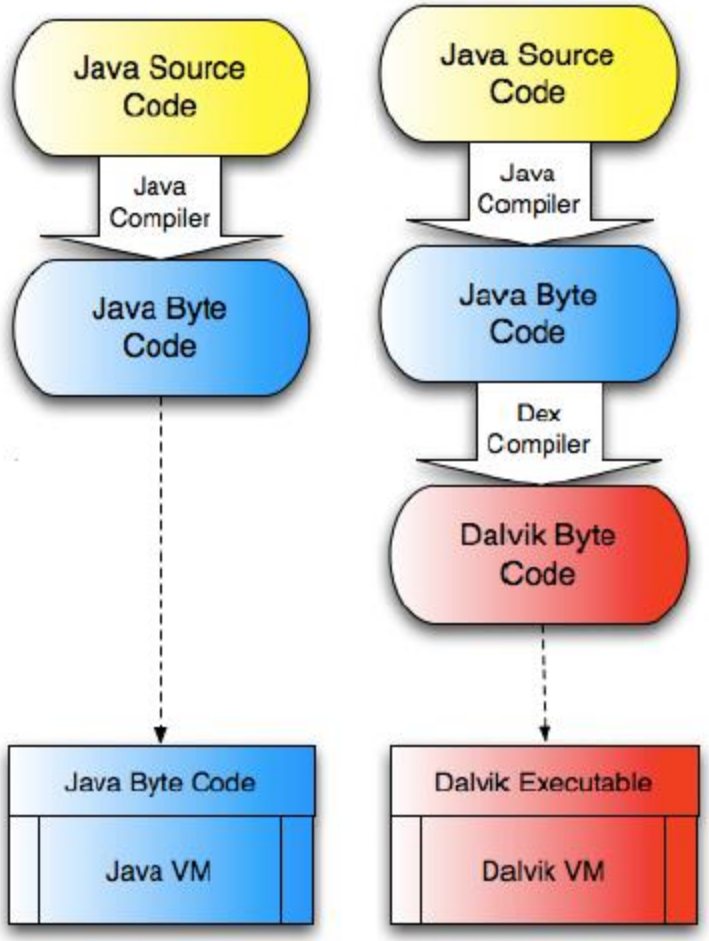
The rich set of system services wrapped in an intuitive Java API.

This ecosystem that developers can easily tap into is what makes writing apps for Android easy.

Location, web, telephony, WiFi, Bluetooth, notifications, media, camera, just to name a few.



Android vs Java



Application Signing

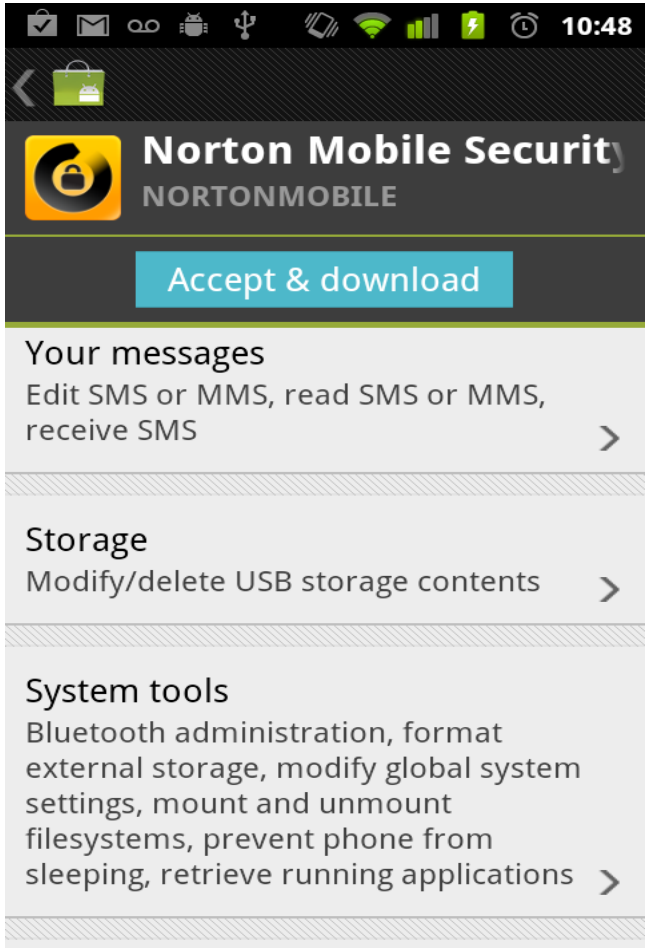
- All apps (.apk files) must be digitally signed prior to installation on a device (and uploading to Android Market)

- The embedded certificate can be **self-signed** (no CA needed!)

- App signing on Android is used to:
  - Ensure the authenticity of the author on **the first install**
  - Ensure the authenticity of the author on **updates**
  - Establish **trust relationship** among apps signed with the same key (share permissions, UID, process)

Permission Granting

- Permissions are granted **once**, at the application install time
- All or nothing
- Permissions across different apps from the same author can share
- The responsibility lies with the user to make educated decisions



Application Distribution

Challenge: No single channel of distribution

- Google Play Store (aka Android Market)
- Side-loading

Piracy

- Reverse Engineering tool
  - ApkExtractor: Google play store or
  - Command line: adb pull /data/app/application.apk name.apk
  - APKTool: https://code.google.com/p/android-apktool/
  - Dex2jar: https://code.google.com/p/dex2jar/

Google Bouncer

- Upon application upload to Market, Google Bouncer scans it for known malware, spyware and trojans.
- Application is then run in a simulated environment (inside Google's cloud) and tested for hidden and malicious behavior (comparing it to previously analyzed apps)
- Already-installed malicious apps can be automatically removed (remote kill-switch)
- 40% decrease in the number of potentially-malicious

Gaining Access to Android User Data

Challenges

- ADB off by default
- Screen lock
- Code signing for updates and boot images
- Encryption
- Variety of device hardware and software configuration

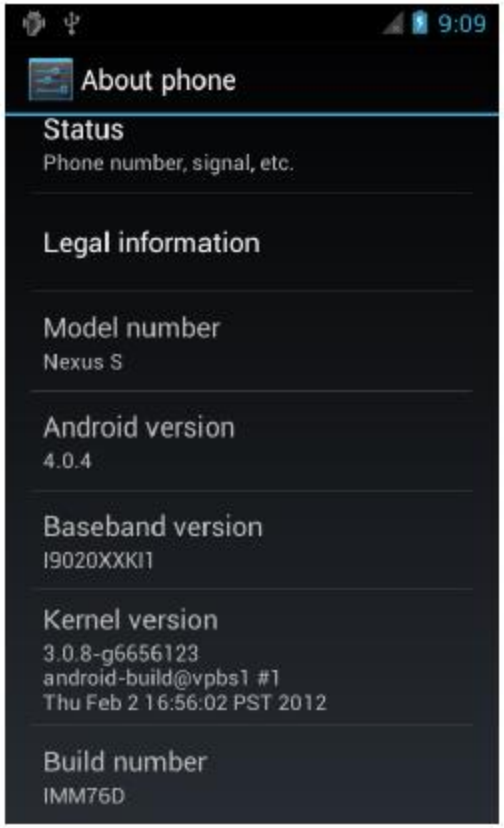
Bootloader

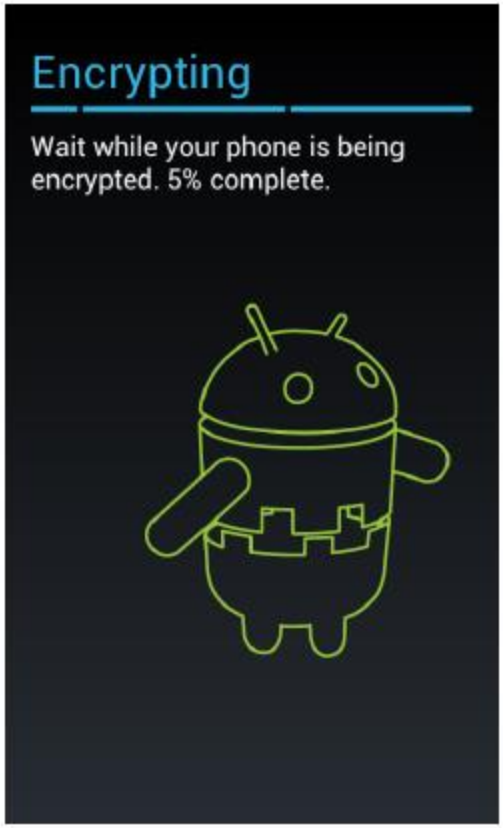
- Usually locked on an Android device
- Virtually impossible to flash a Custom ROM, forced attempts void warranty as well as may end up in bricks.
- Bypass Bootloader without factory reset:
  - Gold Card - specially formatted MicroSD card can bypass carrier ID check when flashing ROMs
  - White Card - special SIM card used as an authentication token to control access to diagnostic mode
  - Forensic Boot Image: Provide ADB root shell over USB which can be used to image the device
  - JTAG Primer: allows to image the NAND flash without booting the device
  - Serial Debug Cable: debug access via serial cables

Screen Unlock

- HID Brute Force: emulates USB keyboard typing PINs

Android Encryption

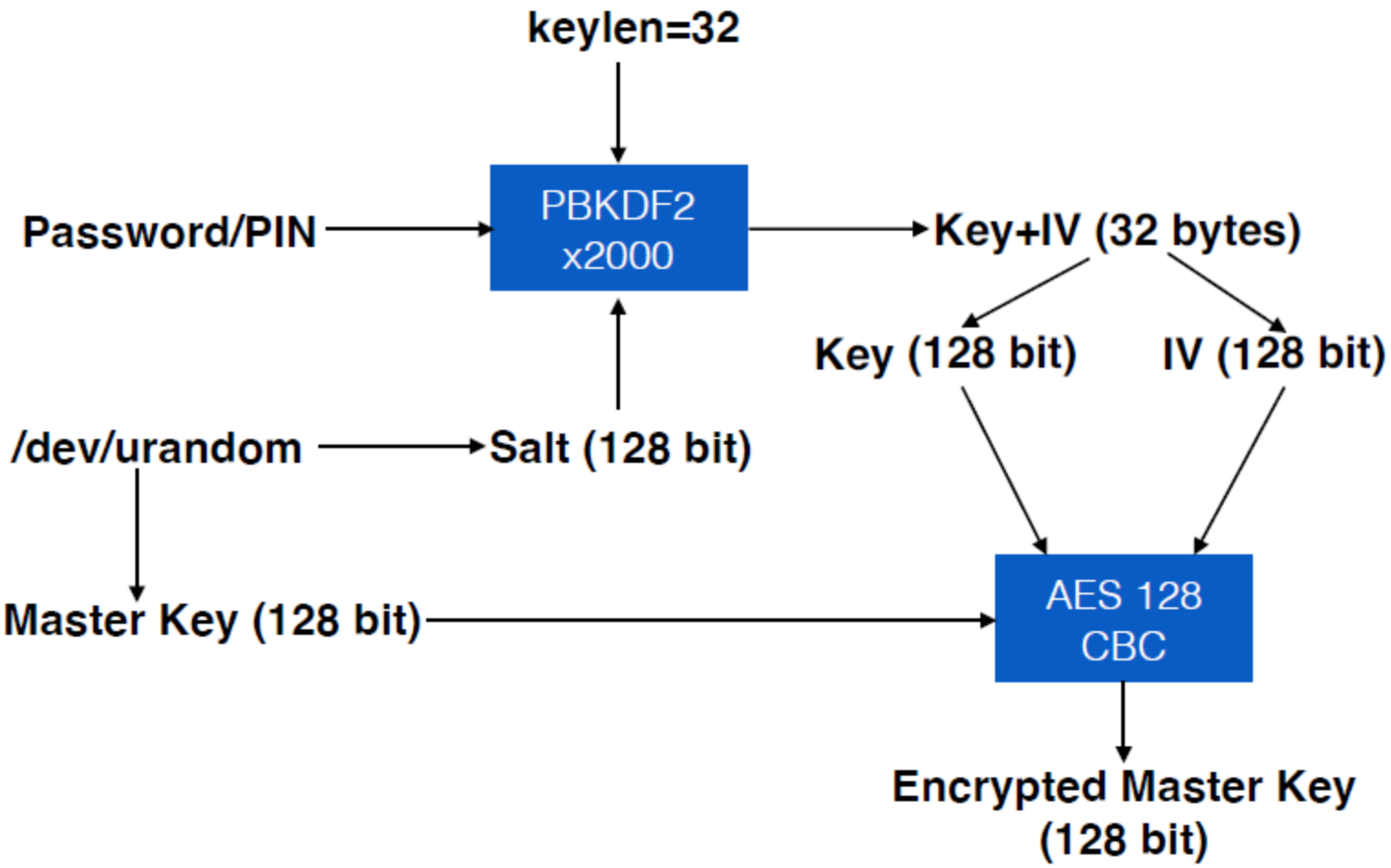




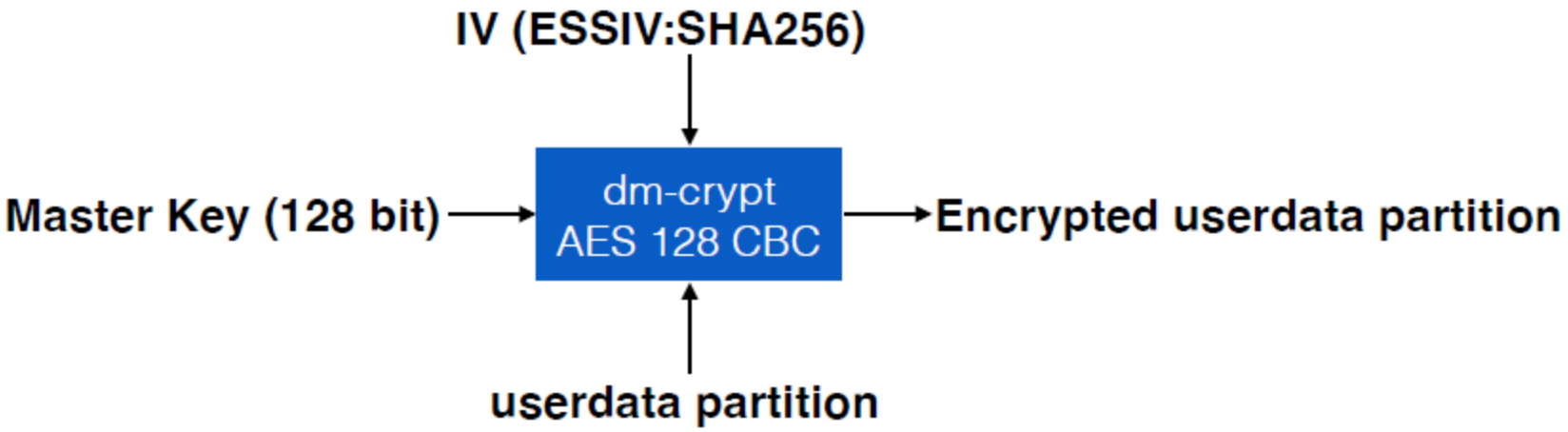
Android Encryption

- Supported since Android 3.0
- Based on dm-crypt
- AES 128 CBC
- Implementations may vary, e.g. Samsung has their own key management module

Android Encryption



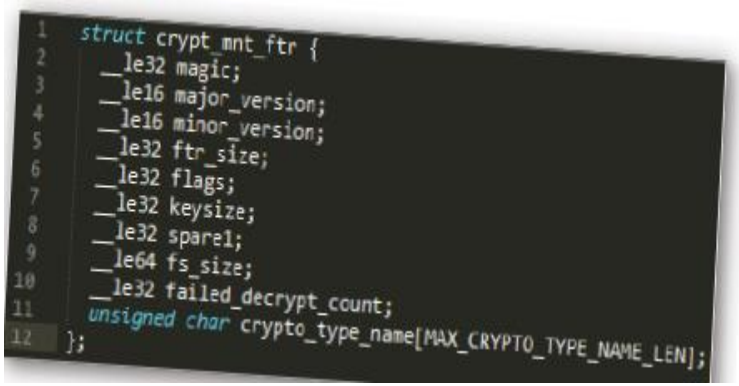
Android Encryption



Note: Encrypted salt-sector initialization vector (**ESSIV**) prevents warermarking attacks by generating IVs from a combination of the sector number with the hash of the key.

Cracking Encryption

- Encrypted Master Key + Salt stored in footer
- Footer stored at end of partition or in a footer file on another partition or as a partition itself
- Image device and locate footer + encrypted userdata partition



```
struct crypt_mnt_ftr {
    __le32 magic;
    __le16 major_version;
    __le16 minor_version;
    __le32 ftr_size;
    __le32 flags;
    __le32 keysize;
    __le32 spare1;
    __le64 fs_size;
    __le32 failed_decrypt_count;
    unsigned char crypto_type_name[MAX_CRYPTO_TYPE_NAME_LEN];
};
```

Cracking Encryption

- Parse footer
- Locate Salt and Encrypted Master Key
- Run a password guess through PBKDF2 with salt, use resulting key and IV to decrypt master key, use resulting master key to decrypt first sector of encrypted image.
- If password is correct, plain text will be revealed
- Cracking PINs takes seconds. Passwords are usually short or follow patterns due to being the same as the lock screen password

Remote Access

Reverse Shell

- App with no permissions can create a reverse shell, giving remote access to attacker.

Secure Communication

HTTP over SSL/TLS

- Bad support/implementation in older phones
- Criticisms towards the Certificate Authority Trust Model (PKI)
- Cost of the certificates

Solution

- Secure Remote Password Protocol (SRP): allows mutual authentication and secure key exchange, while being resistant to on-line brute-force and Man-in-the-Middle attacks (MITM).
- SRP might not ideal solution to integrate with Oauth protocol.

- **Pinning** is an emerging concept of associating a given client with a list of known public keys or certificates.
- Uses SRP to distribute public keys and certificates securely

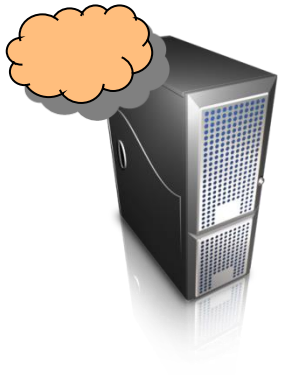
Server Deployment

Open source server app installed on private machine:

- Maintainance
- Configuration

Commercial server app installed on third party machine and offered as a service:

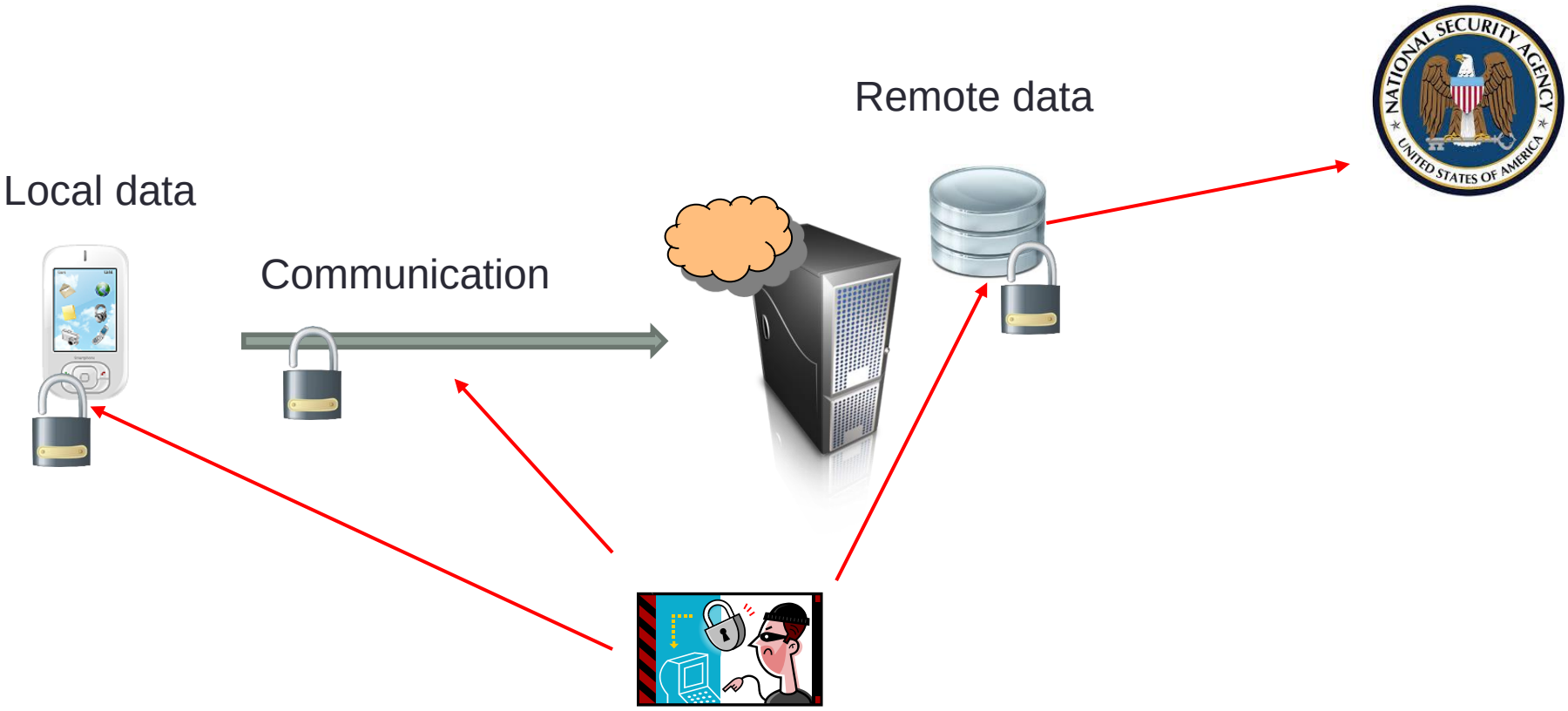
- Open an account on the server
- Pay per collector/ Data traffic



Open source server installed on cloud PaaS (GAE):

- Configuration
- Possibly free
- Easy Client Authentication (Oauth)

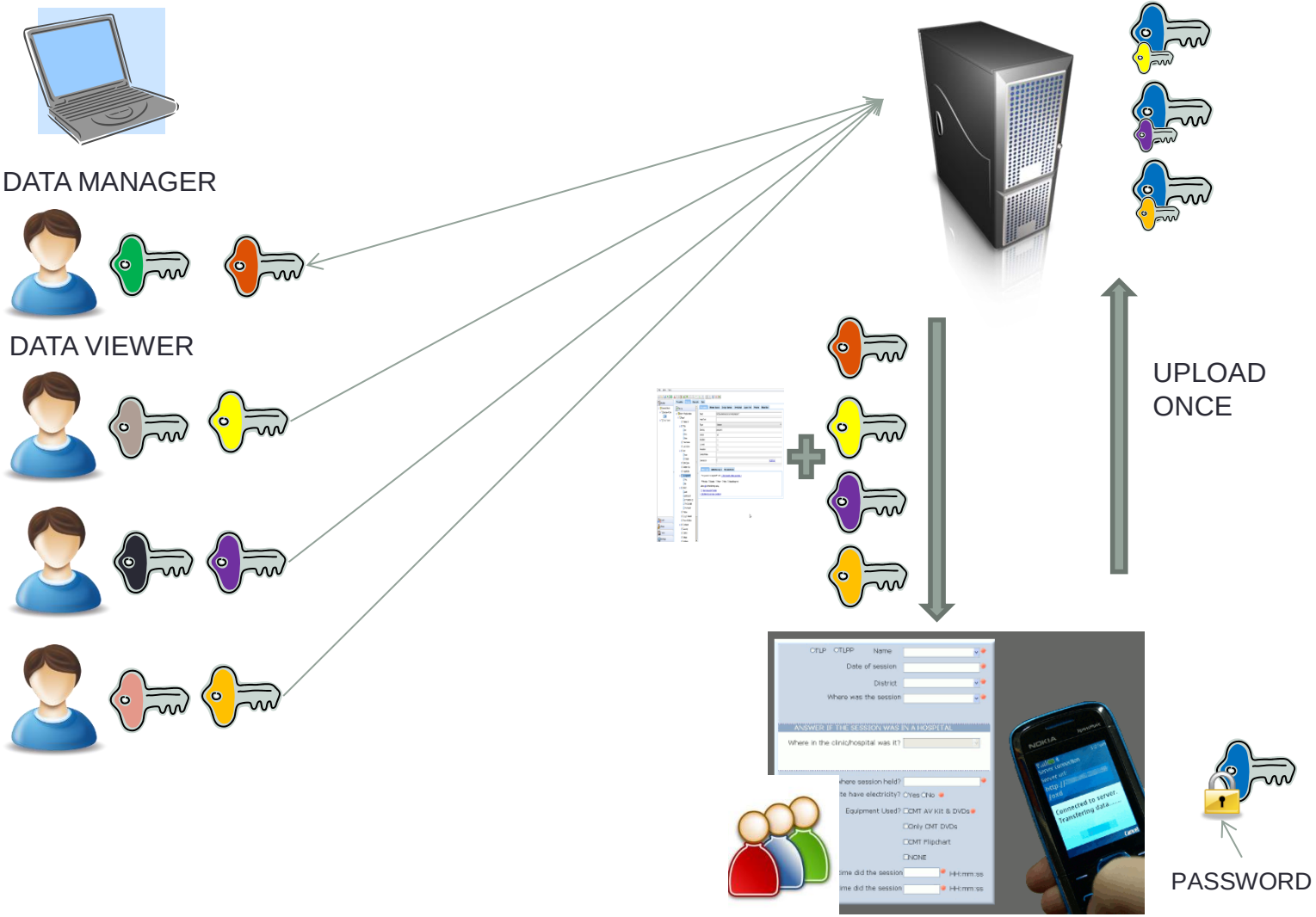
Possible Threats



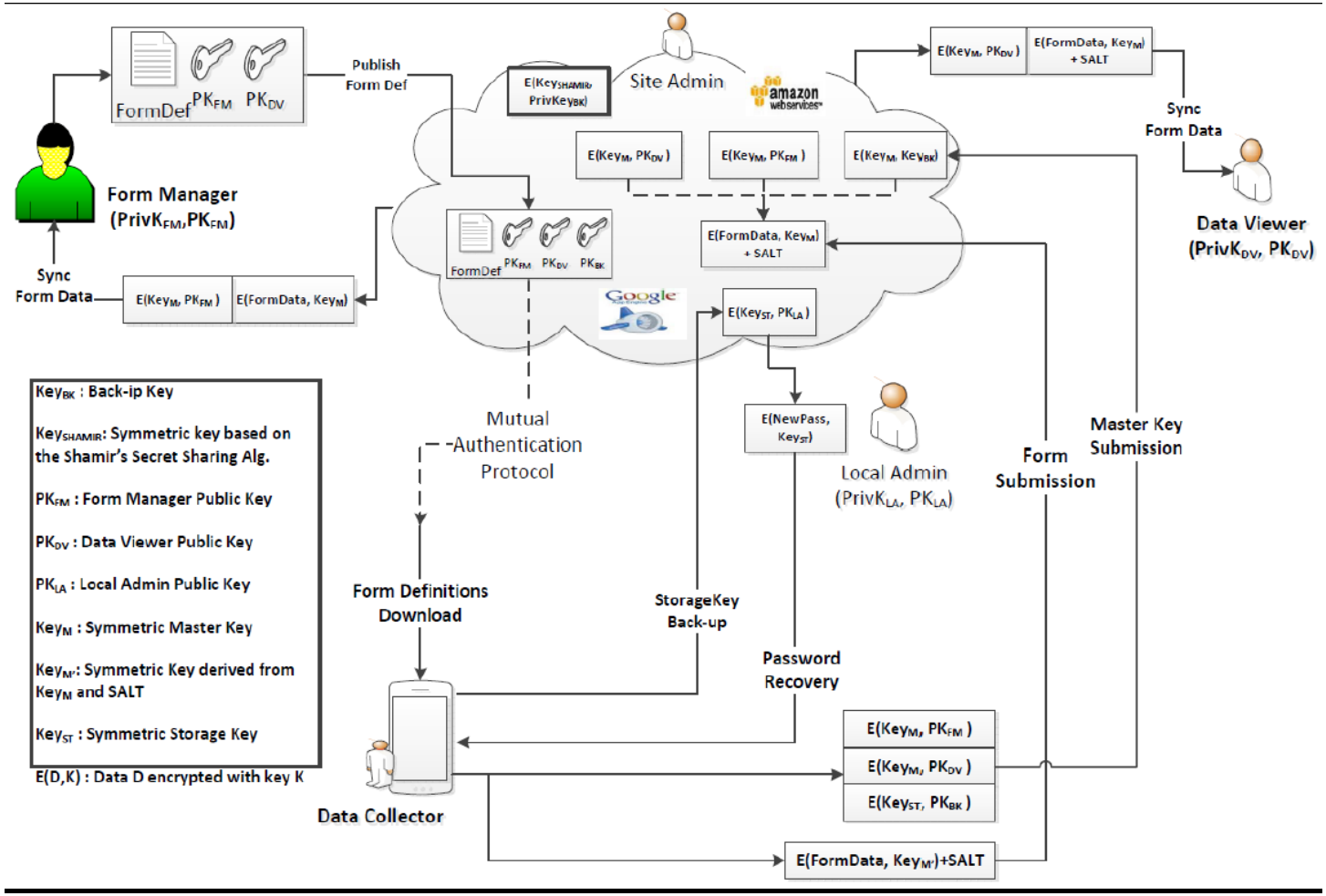
How?

- Encrypt data both locally and on the server
- Keep keys away from the server and protect on the mobile device
- Authenticate always both users and server
- Minimize the damages if a user account is compromised
- Guarantee a back-up plan for disaster recovery (If we have data from hundreds of collectors, losing a password should not compromise the collected data)

Secure Cloud Storage



The complete picture

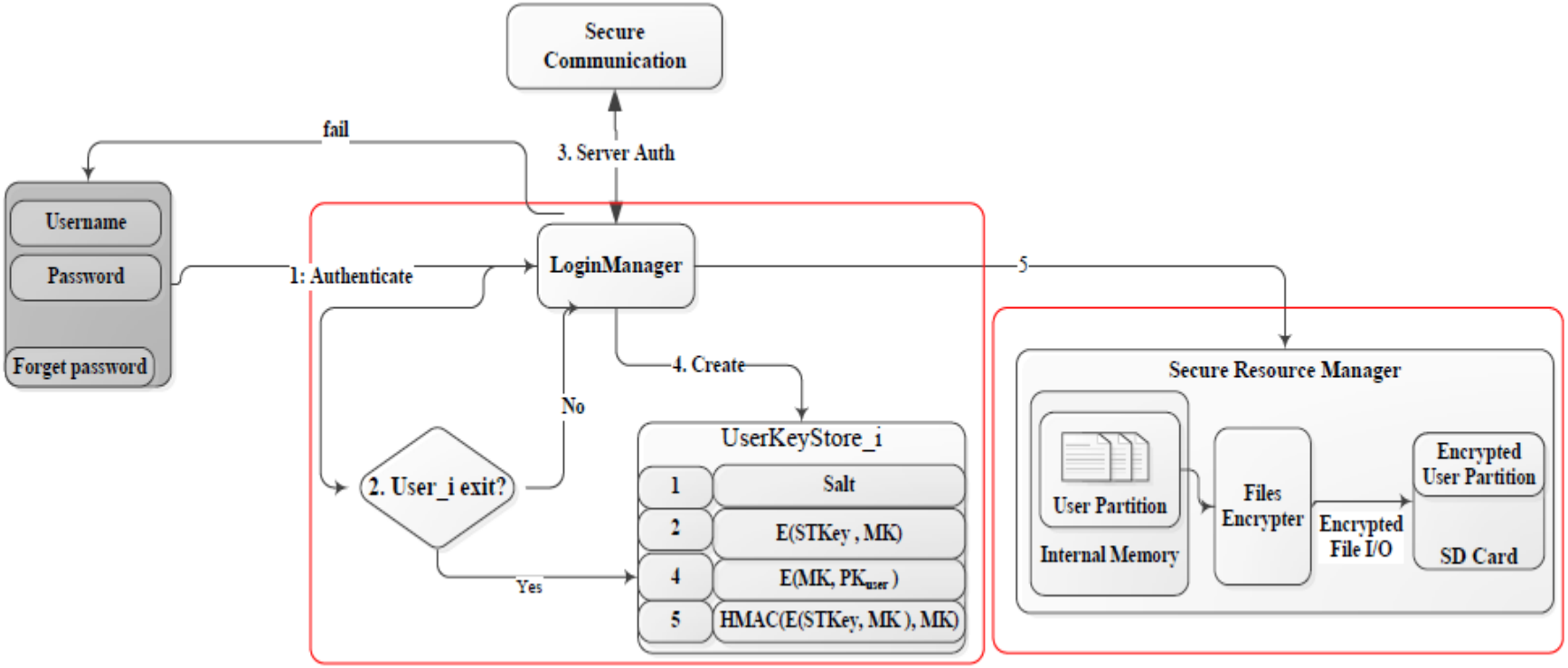


Solution

App Distribution Channel:

1. Manual installation is the ideal choice of secure app distribution.
2. Distributing apps through the vendor websites.
3. Vendors become specialised appstores, where they guarantee for all customised versions of their systems.

Secure Storage



PBKDF2 with higher iteration or scrypt

Conclusions

- Fairly secure solution, but probably little scalable.
- Suitable for small project, where not too much data needs to be downloaded to be analyzed
- Implementation is in progress on ODK, needs to be transparent for the user and integrate with Oauth (no dedicated user accounts)

Thank you!