Daniel Chatfield
Robinson College
`dc584`

Part II Project Proposal

Elliptic Curve Digital Signature Protection for Mifare Classic RFID Cards

October 25, 2016

supervised by

Dr. Markus Kuhn

Introduction

Many organizations use contactless smart cards for authentication, including the University of Cambridge. The Mifare Classic card is one of the most popular smart cards and is currently the card used by the University of Cambridge for everything from access to buildings to paying for meals.

The communication between the card and the reader is based on the ISO-14443-A standard with a proprietary authentication scheme that provides mutual authentication between the reader and the card. The proprietary authentication scheme has been the subject of academic research for several years. Initial research focussed on reverse engineering the protocol and then later research exploited weaknesses to fully compromise the cards [**?**].

The storage on the card is split into sectors, each sector has two keys associated with it (A and B) each of which can be configured to allow read/write access permissions. A typical deployment (like Cambridge University cards) uses one key as a read/write key that is kept secret by the card issuing office and the other key for reading that is distributed to the various organzations that need to read the cards.

Today it is possible to extract all the secret keys from a Mifare Classic card within 10 minutes if just one of the sectors uses a default key[*]. Since it is unusual for all the sectors to have unique keys it is usually much quicker. Once the keys have been extracted, the attacker can read/write to the card and any other cards that have the same keys.

The publication of these vulnerabilities prompted some organizations to move to more secure contactless smart cards but many are still using Mifare Classic cards due to the cost or difficulty involved in replacing all issued cards or upgrading readers.

I plan to improve the security of these cards by digitally signing the data along with the card UID. The reader would then verify that the signature is valid for the data on the card, thus preventing unauthorized modification. By including the card UID in the payload to be signed it substantially raises the bar for cloning a card as it would require the possession of either a Mifare Classic compatible card that allowed the UID to be set or a device that could emulate one such as a Proxmark 3. Elliptic curve digital signatures are appropriate as they offer much shorter signatures than traditional digital signature algorithms, this is desirable with the memory constraints of the card.

It is common to have the need to disable a card before it expires (for example, when an employee leaves or when a card is lost). Some door-access readers are "online"[†] and thus all that is required to revoke access to these is to remove the card from the access list, however a large proportion are "offline"[‡] and require a manual update to the reader to block access to a specific card. I will design and implement a protocol

[*]The Mifare specification requires this for the first sector as it should contain data for identifying who issued the card.

[†]They communicate with some central server to establish whether they should grant access.

for using the cards themselves as a medium for propagating the list of revoked cards from online readers to offline readers.

Starting Point

I will draw upon the following during my project:

- **Computer Science Tripos**
  My project will use knowledge from parts 1A and 1B of the tripos. The digital signature implementation will draw from content from the Security I course. Theory from the 1B Networking course will be useful when designing the offline revocation list distribution. The part II Computer Systems Modeling course may prove useful for modeling the efficiency and reliability of my distribution protocol in different environments.
- **Experience**
  I have some experience with Mifare Classic cards and NFC more broadly and extensive experience with the Go programming language.
- **Go crypto libraries**
  Go has comprehensive cryptography libraries that implement much of the required elliptic curve cryptography required for this project.

Project Structure

Core library in Go

As well as supporting authenticating to, reading from and writing to Mifare Classic cards, the core library will support the writing and verification of elliptic curve digital signatures allowing a reader to perform offline verification of a card without holding the private key used to sign it.

I have chosen Go as the language to implement the core library as it compiles to statically linked binaries that don't require a runtime, has excellent support for cryptographic operations and its channels are a natural fit for the asynchronous nature of NFC communication.

Go can be easily cross compiled for different platforms including Android and iOS, making the codebase very portable.

The development of the library will follow software engineering best practices. I will develop a comprehensive test suite comprising of unit tests to test individual methods, and acceptance tests that test high-level functions like writing an entire

[‡] They either have a preprogrammed access list or the card has a signed token.

card and then reading it. This will help ensure correctness and reduce the likelihood of regressions being introduced as features are added.

The library will provide abstractions to make it easy to perform common tasks in third party projects, without being familiar with how NFC works.

Command Line Interface

The library will be wrapped in a command line interface that exposes its high-level functionality. The CLI will have two modes — one suitable for consumption by a human using a terminal, and one that returns machine readable JSON to make it easy for projects not written in Go to use the library without attempting to both parse the text output and rely on it being backwards compatible in future versions.

Desktop Application

A desktop application will sit on top of the CLI providing a graphical interface for managing the lifecycle of the cards including the writing, reading and verification. This desktop application will be developed using Github's Electron [?], as this makes it easy to make cross-platform applications with rich UIs.

Offline Revocation List Propagation Protocol

The development of the offline propagation protocol will be iterative. I will first make a simple and nave protocol and create a simulation suite that benchmarks this protocol in different environments*. I will then evolve the design to overcome weaknesses, using the simulation suite results to justify and drive the changes made.

Possible Extensions

I'm sure that further possible extensions will become apparent whilst carrying out the project but two possibilities are:

Stronger protection of the private key

The security of the system relies on the private key remaining private. To help ensure this I could extend the application to support the use of a hardware security

*I will vary the number of cards and readers (including offline/online split). I will also simulate different card access patterns in an attempt to both simulate likely real world usage and also identify pathological scenarios.

module that would prevent the key from being covertly extracted from one of the machines that sign the cards.

Alternatively, I could allow the application to connect to a central server to do the signing. This would allow an organization to distribute the application to trusted employees without entrusting them with the key as they would have to sign in to the central server[*] which could then log what was signed by which employee and access could be revoked immediately.

Offline access violation reporting

As I have already stated, many readers are offline and cannot report access violations. Typically the only way to discover such violations is for a technician to physically download the access log[†] from the reader. I could extend my project to allow these doors to use the cards as a medium to report these violations back to an online reader.

Success Criteria

At the end of the project I should be able to:

- Generate new public/private keypairs for use with a new card deployment (or upgrading an existing deployment to support signatures).
- Load an existing private key or public key.
- Provision a new card with a digital signature.
- Upgrade an existing card to include a digital signature.
- Read a card with a digital signature and verify that the signature matches.
- Reject cloned cards where the UID hasn't been spoofed.
- Revoke a card and use the cards to propagate this to offline readers.

Project Timetable

Michaelmas term

24th Oct — 8th Nov

1. Obtain an NFC reader.
2. Obtain some blank Mifare Classic cards.

[*]Cambridge University could use raven for this purpose
[†]Not all readers store a log

3. Research elliptic curve cryptography.
4. Familiarize myself with libnfc.

9th Nov — 22nd Nov

1. Research published weaknesses in Mifare Classic cards.
2. Develop a threat model against the existing system to use as a benchmark for comparison.
3. Begin development of the core library.

Milestone 1 By 22nd November I should be able to read and write to a Mifare card using my library.

23rd Nov — 6th Dec

1. Develop the command line wrapper around the library.
2. Test the signature creation and verification using Mifare cards.

Christmas vacation

6th Dec — 10th Jan
Start development of the desktop application.

Lent term

11th Jan — 17th Jan
Write progress report and complete desktop application.

Milestone 2 By 17th January I should be able to use the desktop application to read and write to the cards and verify the digital signatures.

18th Jan — 31st Jan

1. Design a protocol for offline propagation of the card revocation list.
2. Use modeling to verify that the design works at scale.

1st Feb — 14th Feb
Implement offline revocation list propagation.

15th Feb — 13th Mar
Main chapters of dissertation

Easter Vacation

14th Mar — 17th Apr
Proof read dissertation.

Milestone 3 By the end of the Easter vacation I should have delivered the full draft proposal to my supervisor and DoS.

Resources Required

To complete the project I will need an ISO-14443-A compatible USB NFC reader and some blank Mifare Classic cards. I don't envisage any problems sourcing these as they are both widely available at relatively low cost.

Development and testing will be done on my 2013 MacBook Pro, I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

Backup Policy

I will use git for all source code (including latex), using either github or bitbucket as a remote. In addition to this, I will use Apple time machine to periodically backup all files on my laptop. In the event of hardware failure, I will only lose any changes made since the most recent git push or time machine backup which should be no more than a days work.

Any computer that supports compiling Go is suitable for development and therefore should my laptop become permanently broken it will be easy to source a replacement.