Jacob Fenton
Fitzwilliam
jf574

Computer Science Tripos – Part II – Draft Project Proposal

Efficient Asymmetric Cryptography for RFID Access Control

11th October 2017

Project Originator: Dr Markus Kuhn

Project Supervisor: Dr Markus Kuhn

Director of Studies: Dr Robert Harle

Overseers: Dr Sean Holden and Dr Neel Krishnaswami

Introduction

The current university access control system is based on the MIFARE Classic smart card, which conforms to ISO 14443 Type A, a standard for contactless integrated circuit cards to communicate with a "coupling device" (i.e. a smart card reader) over radio frequency. There are huge numbers of this particular card in existence – over 200 million are in use today.

The cryptography used in the card, a scheme named CRYPTO-1, was developed in-house by the manufacturer of the MIFARE Classic, NXP Semiconductors. NXP chose to keep the scheme secret, a practice known as security by obscurity. Such practice is eschewed by the security community because naturally, all cryptographic schemes are bound to have weaknesses and if researchers (or others) are not able to analyse a scheme, then they cannot provide advice as to how flaws within said scheme can be fixed. Furthermore, obscurity does not prevent others from deducing the scheme by observing it in operation and indeed this was the case for CRYPTO-1. In December 2007, a presentation at the Chaos Communication Congress (an annual security conference) by two German researchers, Nohl and Plötz, described a partial reverse engineering of CRYPTO-1, as well as some weaknesses. They managed to do this by reconstructing the card's electronic circuit from photos of the chip. They then verified their reconstruction by eavesdropping on the reader-card communication. Just a few months later, in March 2008, researchers in the Digital Security group at Radboud University Nijmegen revealed a complete reverse engineering of the scheme and were able to clone and manipulate the contents of a MIFARE Classic card. The most serious attack they detailed in their paper can recover the card's cryptographic key in under a second using only a laptop, without any pre-computation. NXP tried to obtain an injunction to prevent publication of the paper but were unsuccessful.

Ignoring the fact that the CRYPTO-1 scheme is inherently flawed, NXP's choice to use symmetric key cryptography for the MIFARE Classic was perhaps a misstep. Symmetric ciphers utilize what is called a shared secret, or secret key. Two parties wanting to communicate will first exchange this key over a secure channel and then use it to encrypt/decrypt messages sent between them. In the case of access control cards, this means that a card will store just one key – its own secret key, but that key will be stored in every door reader to which the card has access. This means that if a door reader is compromised, and the attacker is able to retrieve all the keys stored within, then they're able to clone any card which had access to that door. If this door is not in a very specific department, then many people will have access to it,

and thus the attacker will have access to a very diverse set of doors – essentially, the entire system is compromised. Such a weakness does not exist when using a scheme based on asymmetric key cryptography, in which each card has not one but two keys – one public, one private. The private key is known only to the owner and is never sent over any channel, whilst the public key is known to everyone. If two parties wish to communicate, then they encrypt their messages with each other's public keys. The message can now only be decrypted with the recipient's private key, which only the recipient knows. In this case, the door reader contains only a long list of public keys corresponding to all the cards that can access the door. Thus, an attacker who's able to compromise a reader doesn't learn any secret information except for the door's private key. This only allows them to clone that specific door reader and doesn't compromise any other cards or readers in the system.

The aim of the project is to produce an access control system that uses asymmetric key cryptography to authenticate smart cards. The system should act as a replacement for the existing MIFARE Classic system.

Starting point

The project will make significant use of the material from Security I and Security II - I have already studied the lecture notes for Security II, although I have set aside some time in my plan for recap. Further, material from Object-Oriented Programming and Further Java will be utilised when writing the smart card application, which runs on the JavaCard platform and supports a subset of the Java language.

A previous attempt was made at this project by Denys Natykan, and so his dissertation must be mentioned as a starting point. However, I already anticipate significant differences between our end products as I intend to use a rather different authentication protocol.

Substance and structure of the project

The objective of the project is to produce an access control system that implements an authentication protocol based on asymmetric key cryptography. As well as producing a card application and reader application that implement the authentication protocol, I intend to write a card-provisioning application that can be used to issue new cards or reprogram existing cards.

I intend to compare at least two different digital signature algorithms (DSAs) for speed in my evaluation, and it's possible that one or more of these algorithms won't be implemented by the JavaCard SDK, in which case I will have to implement them myself.

Given that smart cards are low power, I expect that I may have to spend time optimising the protocol, so that authentication happens within the required time.

Success criteria and evaluation

- An authentication protocol must be chosen.
- The protocol must be implemented in two separate applications – one to run on the card, the other on the reader.
- A command-line application must be written for provisioning and reprogramming cards.
- The system must be able to authenticate a card in less than one second.
- The system should be tested to ensure it operates as the protocol dictates it should.
- The dissertation must be planned and written.

Possible extensions

- Mutual authentication of both card and controller so that only authorised readers (i.e. university door controllers) are able to communicate with cards.
- Provide user untraceability as a feature of the authentication protocol.
- Ensure the system is resilient to cloning.
- Implement a GUI for the card-issuing application.

Timetable

Weeks 1 to 2

Initial research period. I will familiarise myself with existing authentication protocols and either select one of them to use, either in full or as a guideline, else I will design one myself. Familiarisation with the JavaCard SDK and the GlobalPlatform API.

Weeks 3 to 4

Implement a very basic challenge-response application on the smart card. Gain a deeper understanding of the J3A040 smart card, specifically the memory structure and the implications of this for fast authentication.

Weeks 5 to 10

Implement the chosen authentication protocol on the card and reader. Implement the card-provisioning application to run on computer.

Weeks 11 to 12

Time reserved for testing the system and sorting out any leftover bugs in the system.

Weeks 13 to 16

Reserved for dealing with bugs. If the base system is in good working order, then this time can be used to implement extensions. I'm currently undecided as to which extensions will be prioritised – my choice will depend on available time.

Weeks 17 to 20

Perform evaluation of the system. Begin writing dissertation.

Weeks 21 to 23

Time reserved for handling any bugs that escaped notice earlier in the process. This time can be used for writing the dissertation if there's nothing to be fixed.

Weeks 24 to 25

Finish writing initial draft of the dissertation.

Weeks 26 to 28

Time left to send dissertation draft to supervisor for review (this may happen a couple of times) and make changes. Finalise dissertation and submit electronically.

Resources declaration

- NXP J3A040 - a programmable smart card supporting JavaCard SDK and GlobalPlatform API.
- SCL3711 – a USB smart card reader.
- JavaCard SDK and GPShell for programming smart cards.
- My own MacBook Pro and Lenovo Yoga 2 Pro for writing applications, documentation and dissertation. I plan to use Git for source control, and will regularly push to a remote Bitbucket repository to avoid significant loss in the event that I experience a hard drive failure.