Jacob Fenton
Fitzwilliam
jf574@cam.ac.uk

Computer Science Tripos - Part II - Progress Report

Efficient Asymmetric Cryptography for RFID Access Control

Project Supervisor: Dr Markus Kuhn
Director of Studies: Dr Robert Harle
Overseers: Dr Sean Holden and Dr Neel Krishnaswami

So far, I have completed the implementation of the card provisioning program, which tells the card to generate a public/private key pair and signs the public key, along with the card owner's CRSID and Group ID and an expiry date for the signature (all of which are provided as command line arguments to the provisioning application). Group ID has been implemented as a variable length byte-array, so its format is not fixed by my implementation. One possible idea is to have what would essentially be a bitstring, where each bit represents a location, and that bit is set to 1 if the card owner is allowed access to that location, else 0.

I chose to implement the OPACITY-FS (Forward Secrecy) protocol, which provides mutual authentication and privacy as built-ins. It manages this by having the reader send over its public key which is signed by the provisioning certificate (so cannot be faked). Once the card verifies the signature, it derives a new key using Diffie-Hellman (DH) with the reader's public key and an ephemeral private key that it generates on the spot. It then AES encrypts its own public key using this derived key and sends that to the reader, along with its ephemeral public key. Thus the reader can only decrypt the card's public key if it has the private key that matches the verified public key from earlier (and performs DH with its private key and the card's ephemeral public key). I haven't yet finished writing the card or reader applications which implement the protocol, and as such I'm behind my original schedule. However, I'm behind by only a week or two, and I reserved the first six weeks of Lent term for bug fixing/implementing extensions, so it's of no significant cost to use some of this time to finish writing the base system.

Unexpected difficulties were mainly due to unfamiliarity with the smart card platform or else with other parts of the technology stack. As an example, when I initially attempted to install "applets" (the word used in the smart card literature to describe an application that runs on a smart card), I had to fiddle with certain rather inconsequential values that are passed to the card during the applet installation process. For example, when installing an applet, one must first select the card manager applet by using the select command in GPShell, passing the card manager applet identifier (AID) as an argument. Similarly, when using the install command in GPShell, the security domain AID must be passed as an argument. Both the AID of the card manager and security domain are manufacturer specific and it was very unclear what these values needed to be. To further aggravate the problem, the documentation provided by the manufacturer of the smart card I chose to use for the project (NXP J3A040) was sparse.