Security Protocol Project

Digital Security

Radboud University Nijmegen

This course: form

- Lectures & some reading material
- Group project to design, build and document a JavaCard smartcard application
  in groups of 4 students
- Grade based on group project
- *Form groups of 4 persons asap !*

Learning objectives

- Experience the whole process from high-level design, given security requirements and assumptions, down to actual code on real hardware

- Appreciate complexity & interplay of
  - design considerations & constraints,
  - key management & distribution,
  - protocols,
  - low level implementation details,
  - silly hardware limitations, weird crypto padding, …
  - practicalities of *getting all of this working*

- How to document this whole process

Prerequisites

- Very basic knowledge of (a)symmetric crypto:
  - hashing
  - using (a)symmetric crypto for encrypting, signing, MACing
  - the use of certificates with asymmetric crypto

- Knowledge of basic security concepts such as CIA

Group project

You have been contracted to build a system

- electronic purse
- loyalty card
- petrol rationing
- car rental

that uses a smartcard

So you must

- design security protocols for this smartcard to interact with terminals,
- think about keys, certificates, PIN codes, etc. this requires,
- implement all this
  - with only bare-bones implementation of the terminals & back-end

Design constraints

1. you must use a *JavaCard smartcard*

   which can (securely) execute code and store data, incl. use of PIN codes & standard (a)symmetric crypto (eg AES and RSA)

2. you must store *some modifiable info on the card*

   - not just fixed crypto keys but also eg. card balance, credits, logs, counters, …

   so that some terminals can operate offline (maybe temporarily)

In our increasingly online world, a solution where cards only store keys for authentication and everything happens online a central back-end makes perfect sense, but for this assignment it is not allowed.

Smartcard basics (more details later)

A smartcard is simply a tiny, low-power computer

- few KB of RAM (aka volatile memory)
- a bit more EEPROM (aka persistent memory) that acts as SSD/hard drive
- very low-bandwidth communication, with messages usually just a few dozen bytes

It can execute arbitrary code and store arbitrary data,

e.g. a card number, customer ID, cryptographic keys, certificates, PIN codes, counters, JPEGs, …

Smartcard basics (more details later)

A smartcard is secure and tamper-resistant computer, i.e.

- Data & software on the card cannot be read or modified, so card ensures
  - integrity of the software
    - also confidentiality of the software
  - confidentiality & integrity of all data

- Installing code on the card is tightly controlled and usually disabled before the card is issued.

Smartcard attack basics (more details later)

Attackers can always do

- Man-in-the-Middle attacks on communication between the card & the terminal
- card tear attacks by removing the card from the terminal & its power supply
  - Special case of MitM attack: it abruptly aborts the program executing on the card *and* wipes the RAM memory content.

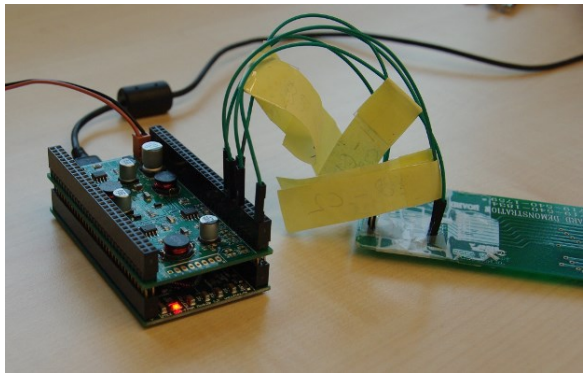
and may be able to do

- side-channel attacks
  - eg. analysing power consumption to retrieve cryptographic keys

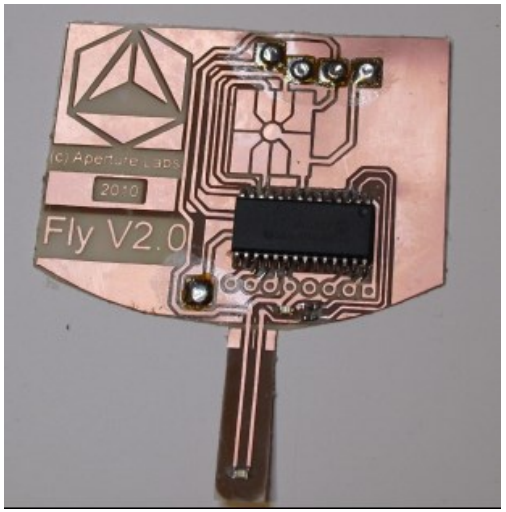
  So you should make sure that different cards use different keys.

Lots more about side-channel attacks in 'Physical Attacks on Secure Systems' (NWI-IMC068) & 'Selected topics on hardware for security' (NWI-IMC065) by Lejla Batina & Ileana Buhan.

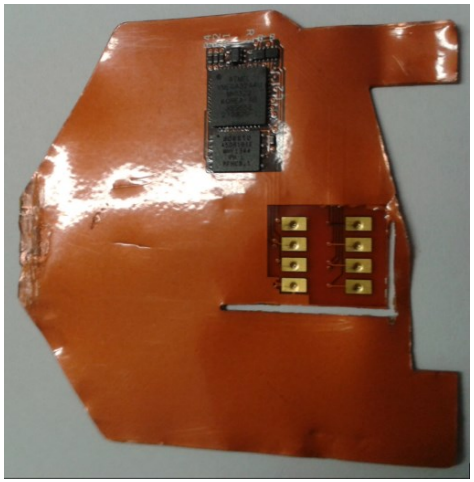
Man-in-the-Middle attacks using shims



Smartlogic tool by Gerhard de Koning Gans



Commercial shim



Shim found inside an ATM
https://krebsonsecurity.com/tag/atm-shimming/

Side-channel attacks

Using side-channel analysis attacker may be able to extract a key from the smartcard
so you should *not* have the same key in all cards

1st step: write a high level design document

Concise & clear document that outlines and motivates your design

- including *security requirements*, *threat / attacker model*, *trust assumptions*, *design decisions*
- down to details like
  - key & certificate distribution
  - abstract security protocols
    - as MSC or in Alice-> Bob style
    - with clearly stated security goals (eg. authentication, non-repudiation, …)
    - use of PIN codes or not,
    - which info gets logged, …
- 8 pages max, but try to use less

Target audience: security professional that has to assess the security of it this (so no silly marketing blurb)

More info in Brightspace & coming lectures.

Attacker model & trust assumptions

Your attacker model must include

- active Man-in-the-Middle attacks on all communications between cards & terminals
- card tear attacks
- side channel attacks to extract keys from individual card

W.r.t. your trust assumptions :

- the software on the smartcard will be in the TCB
- you may also need to trust terminals and employees (and maybe even customers?) for some specific properties.

NB even if you cannot prevent some attack by a component or actor, you may be able to detect it.

Use cases: *personalisation, issuance & end-of-life?*

- Cards need to be personalised
  - installing software, initialising keys, PIN codes, IDs, names, …

  before it is issued to the user (aka card holder)

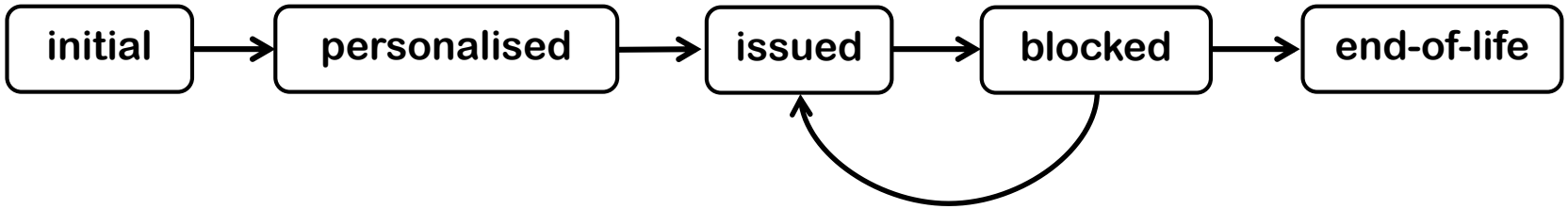
  This will typically require a separate (trusted) terminal.
  - In addition to say point-of-sale terminal.
  - Personalisation may happen in several stages.

- Cards may also need to be disabled, eg. at the end-of-life?
  - Or still be able to report data for fraud investigations?

Be explicit about the life-cycle of the card, eg with a state diagram

Persistent life cycle state

Card always has to record some life cycle state



This state has to be recorded & maintained in persistent memory (ie EEPROM)

Your report MUST include a state machine like this!

Getting started

- Next week: more discussion of the design document & any questions you may have

- Deadline for the initial design document: Feb 18.
  But the sooner you hand it in, the better.

- Lots more info in Brightspace.