Security Protocol Project

Generic Feedback

Cristian Daniele & Erik Poll
Digital Security
Radboud University Nijmegen

Use case: lost or stolen cards

What happens if cards get stolen or lost?

Reporting a card as stolen or lost would be a separate use case

Decision not to have procedure for this deserves to be explicitly stated & motivated.

Attacker/threat model

Don’t forget to explicitly state the security requirement that ‘breaking’ a single card (e.g. retrieving key material by side channel analysis) should not break the entire system

Most groups have thought about this, but almost no group stated it as requirement

Blocking cards

‘Blocking a card’ is an overloaded term, as it can mean

a) blocking a card itself

b) blocking a card in the back-end

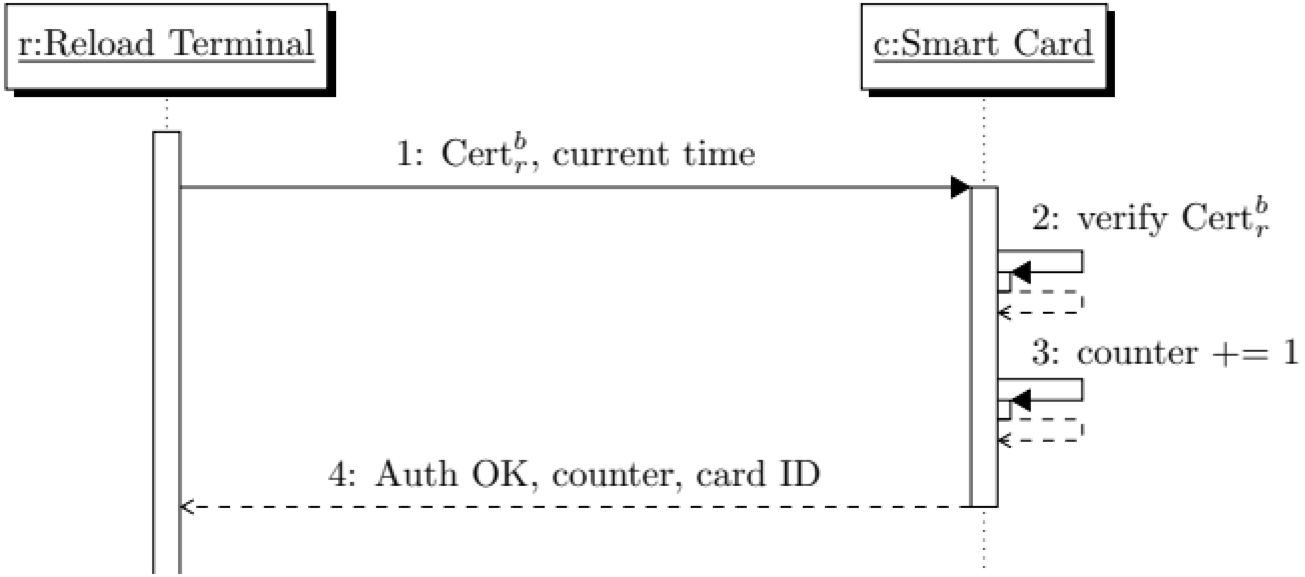
ie. setting a EEPROM `state` flag on the card to `BLOCKED`

vs

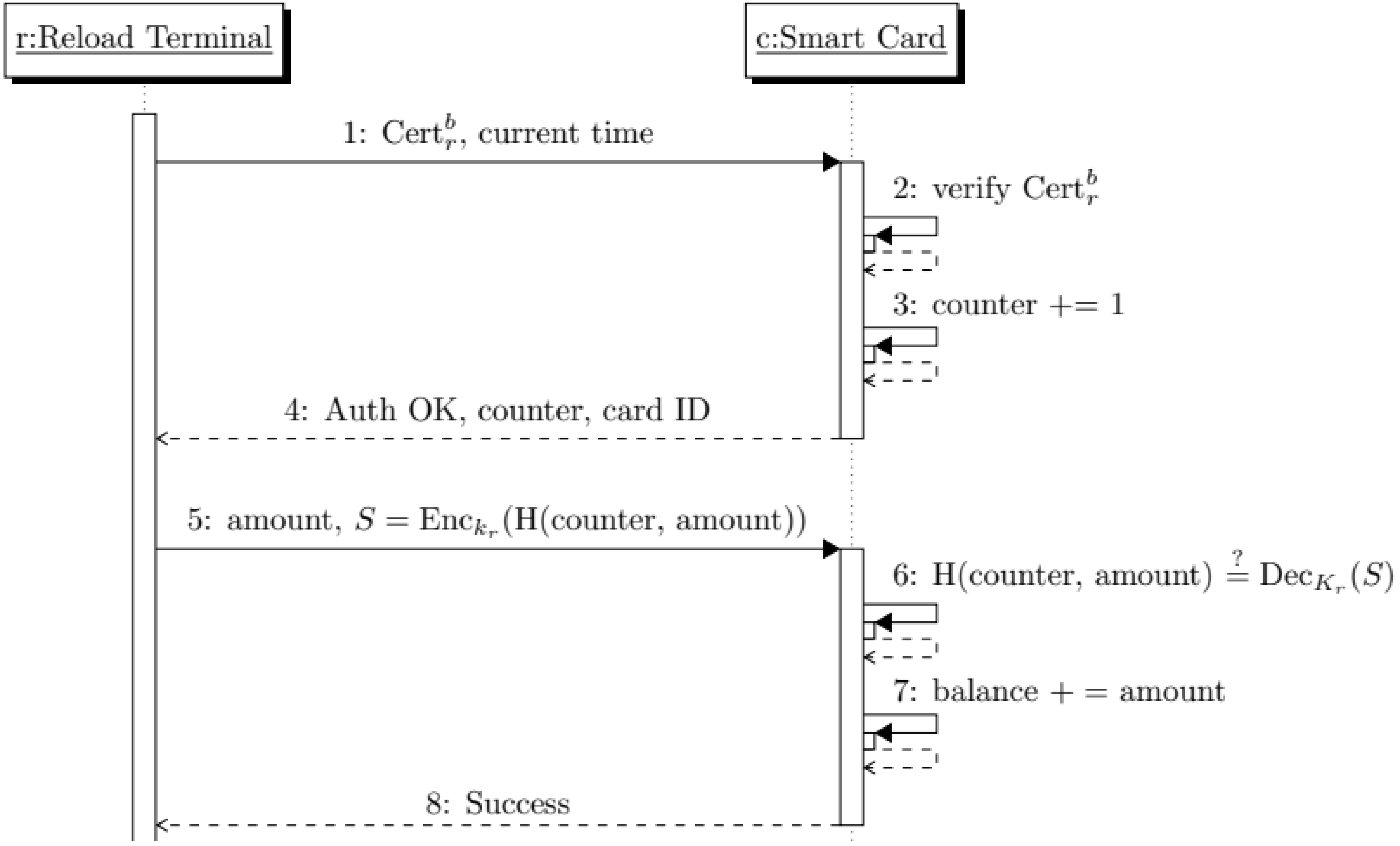
setting some flag in the back-end database

You may have both operations and they may then be related.

Spot the security flaw (1)


r:Reload Terminal
c:Smart Card
1: Cert_r^b, current time
2: verify Cert_r^b
3: counter += 1
4: Auth OK, counter, card ID

Spot the security flaw (2 & 3)


r:Reload Terminal
c:Smart Card
1: Cert_r^b, current time
2: verify Cert_r^b
3: counter += 1
4: Auth OK, counter, card ID
5: amount, $S = \text{Enc}_{k_r}(\text{H}(\text{counter, amount}))$
6: $\text{H}(\text{counter, amount}) \stackrel{?}{=} \text{Dec}_{K_r}(S)$
7: balance + = amount
8: Success

Spotting protocol flaws/improvements

- Authentication MUST use some form of challenge-response
  - Just exchanging & certificates is not enough!
  - The challenge has to be a nonce,
which can be a random number OR a counter

- Double-check that message that triggers the actual transaction cannot be replayed

- Beware of unauthenticated responses
  eg a card or terminal saying `OK`

Spotting protocol flaws/improvements

- If you have a session key, it's dangerous to let only one party decide the session key
  - better - or necessary – to let both parties contribute randomness

- MACing or signing data with *long-term* (private) key provides a stronger guarantee than MACing with a *session* key

- If you use encryption in your protocol, double-check if there's a corresponding security requirement about confidentiality
  - unless it's encryption of a nonce for authentication, of course

IDs

- If a card (or terminal) has its own keypair,
  then you can use public keys to identify that card.

- But it's much cleaner to give cards and terminals unique identifier *cid* and *tid* as well as own keypairs
  - You might want to have customer-id's *and* card-id's

Certificates

- A certificate is not a just a signed public key,
  it is a signed blob of information that *includes* a public key

A typical certificate will be

$(\text{id} \mid\mid \text{PubKey}_{\text{id}} \mid\mid \text{expiry-date} \mid\mid \text{type-info} \mid\mid \ldots)$

signed by a public master key

Or more formally

$\text{Cert}_{\text{id}} = \text{Signed}_{\text{PubKeyM}}(\text{id} \mid\mid \text{PbK}_{\text{id}} \mid\mid \text{expiry-date} \mid\mid \ldots)$

where

$\text{Signed}_{\text{PK}}(m) = m \mid\mid \text{Enc}_{\text{PK}}(\text{hash}(m))$

Notation

- Be aware of the difference between
  - constants, e.g. `BLOCKED` and `OK`
  - programs variables, e.g. `state`
  - meta-variables, which e.g. stand for *values* used in protocols such as *amount* , *card_id* , *terminal_id* , or *PIN_guess*

  Some meta-variables also appear as program variables; different fonts can help to distinguish them

- Be aware of different meanings of =, which include
  - mathematical definitions

    $$EncSign_{K1,K2}(m) =_{def} Enc_{K1}(m) \,||\, Enc_{K2}(hash(m))$$

  - assignments in code

```
state := PERSONALISED
```

Notation

- Introduce convenient mathematical functions & notation

  Eg

  $m = \mathit{Encrypt}_K(\mathit{amount} \,||\, \mathit{card\_id} \,||\, \mathit{nonce})$

  $(\mathit{amount}, \mathit{cid}, \mathit{time}) = \mathit{Decrypt}_K(\mathit{payload})$

  $m_2 = \mathit{DecryptAndCheckSignature}_{K1,K2}(m_1)$ or abort if signature incorrect

- Numbering steps in protocols can be useful
  - also when you start coding

Avoid duplication

Duplication is bad in code, but also in text,

so avoid it in your report

when describing protocols, giving definitions, discussing attacker models, listing security requirements, ...

- It's better to have fewer SRs than many SRs
  - so avoid duplicating or overlapping security requirements

Defense in Depth

What if…? one of your security assumptions is broken

Would you be able to detect it if

- *a malicious insider issues loads of cards?*
- *a malicious POS operator gives away free points or redeems non-existent points?*
- *a malicious shop owner claims too much money, eg by duplicating transactions?*
- *a card is cloned?*
- *key material from a terminal leaks?*
- *a terminal is hacked to compromise its behaviour?*
- *…*

Logging & procedures to inspect logs can help