Modelling audit security for smart-card payment schemes with UMLsec

Jan Jürjens

Computing Laboratory, University of Oxford

jan@comlab.ox.ac.uk

http://www.jurjens.de/jan

Secure Systems Development

Designing secure systems correctly is difficult:

Many flaws found in designs of security-critical systems, sometimes years after publication.

Use formal concepts and tools for systems design to ensure correctness of security design.

This Work

Use (formal core of) Unified Modeling Language (UML):
industry standard for specifying o-o systems.

Consider accountability and enforcement of audit policy
for Common Electronic Purse Specifications (CEPS).

Candidate for globally interoperable electronic purse standard.

Encapsulate knowledge on prudent security engineering,
make it available to developers.

UMLsec

Here:

Class diagrams: static structure of system. Classes with attributes and operations/signals; relationships between classes. Ensure secure communication between objects.

Statechart diagrams: dynamic behaviour of individual object. Input events cause state change and output actions. Ensure secure behaviour within an object.

Messages

Define set of messages $\mathbf{Exp}$ inductively by

$E ::=$	expression
d	data value ($d \in \mathcal{D}$)
K	key ($K \in \mathbf{Keys}$)
x	variable ($x \in \mathbf{Var}$)
$(E_1, \ldots, E_n)$	concatenation
$\mathsf{Enc}(K, E)$	encryption ($K \in \mathbf{Keys} \cup \mathbf{Var}$)
$\mathsf{Dec}(K, E)$	decryption ($K \in \mathbf{Keys} \cup \mathbf{Var}$)
$\mathsf{Mac}(K, E)$	MAC ($K \in \mathbf{Keys} \cup \mathbf{Var}$)
$\mathsf{Ver}(K, E)$	verify MAC ($K \in \mathbf{Keys} \cup \mathbf{Var}$)

Symmetric encryption.
Assume $\mathsf{Dec}(K, \mathsf{Enc}(K, E)) = E$ and $\mathsf{Ver}(K, \mathsf{Mac}(K, E)) = E$.

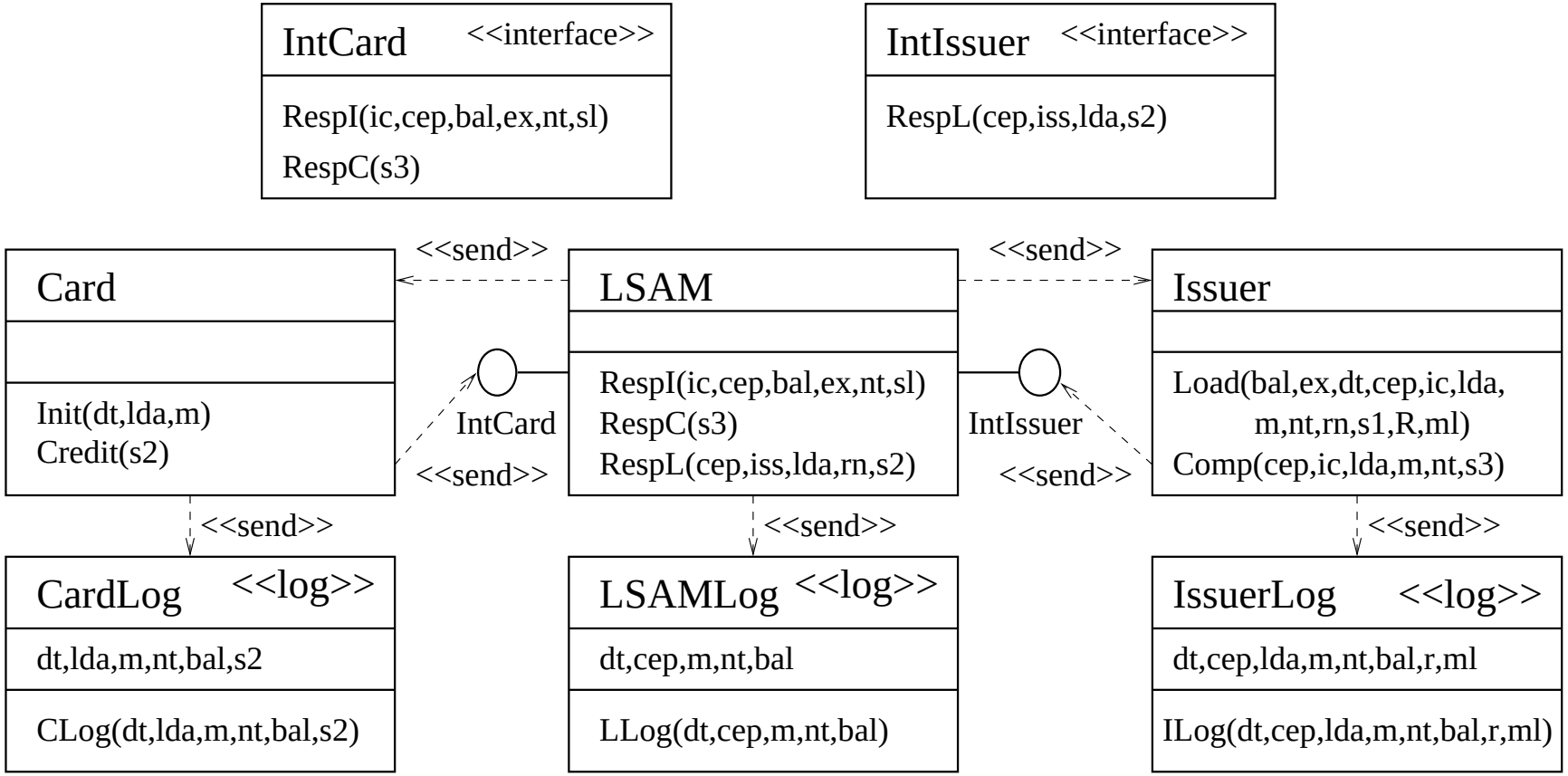
Common Electronic Purse Specifications (CEPS)

Smart-card based payment system: more fraud protection than credit cards (transaction-bound authentication).

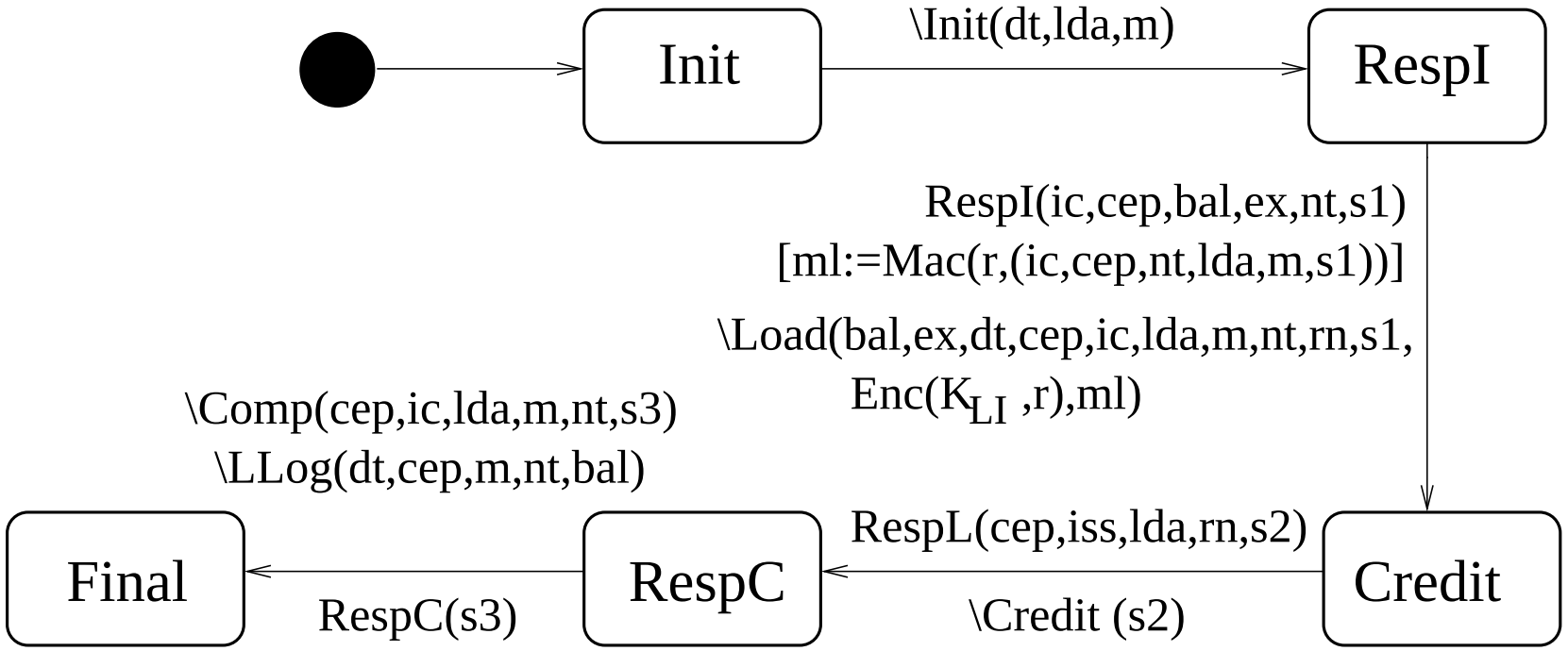
CEPS candidate for globally interoperable electronic purse scheme providing accountability and auditability.

Here: (unlinked, cash-based) load transaction.
Load value onto card using cash at load device.
Load device contains Load Security Application Module (LSAM):
    secure data processing and storage.
Card account balance adjusted; transaction data logged
    and sent to issuer for financial settlement.

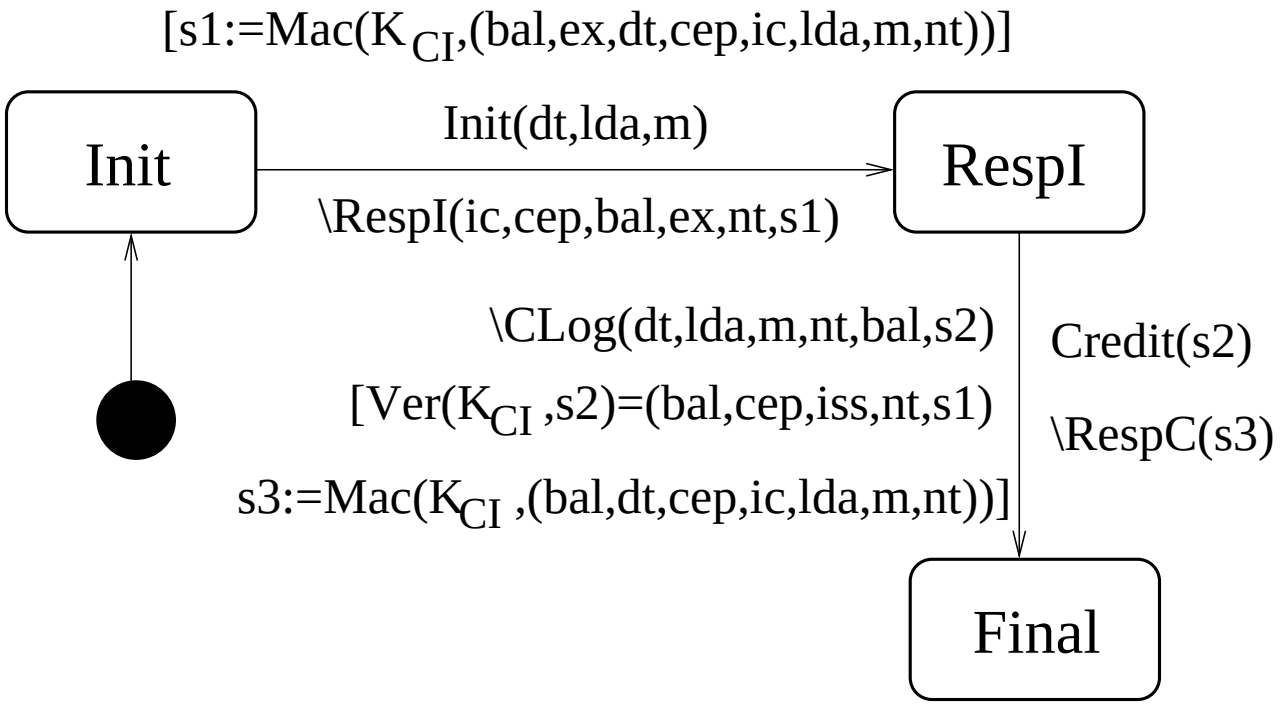
Load Transaction: Class diagram



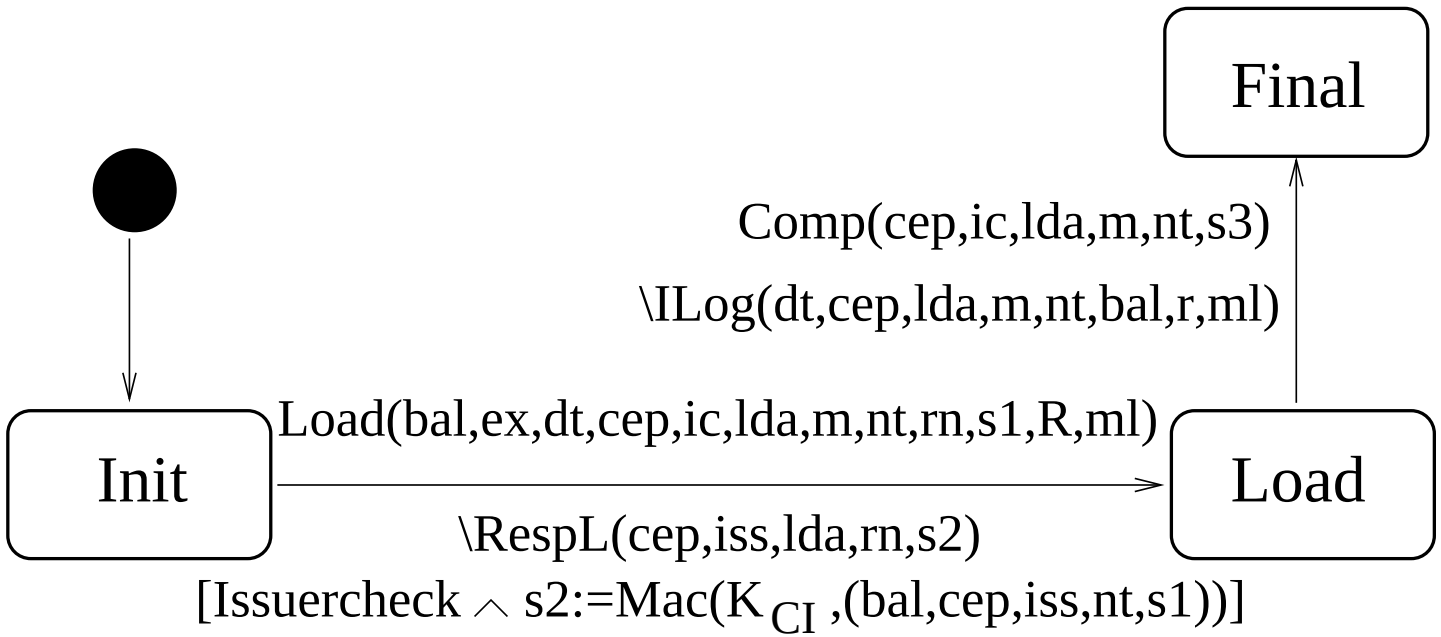
LSAM Statechart



Card Statechart



Issuer Statechart



Security Threat Model

Card, LSAM, issuer security module assumed tamper-resistant.

Threat scenario: intercept (modify etc.) communication
between card, LSAM, issuer.

Attack motivations:
Cardholder: charge without pay
Load acquirer: keep cardholder's money
Card issuer: demand money from load acquirer
Third party: disrupt service

May coincide or collude.

Audit security

Tamper-resistance could be defeated ? Produce own cards ?

No direct communication between card and cardholder.
May manipulate load device display.

Use post-transaction settlement scheme.

Relies on secure auditing.

Verify this here (only runs completed without exception).

Security conditions (1)

Audit security condition on CardLog in final state:

Correct amount: $s2$ and $s1$ verify correctly:

- $\mathsf{Ver}(K_{CI}, \mathsf{CardLog}.s2) = (bal', cep', iss', nt', s1')$
- $\mathsf{Ver}(K_{CI}, s1') = (bal'', ex'', dt'', cep'', ic'', lda'', m'', nt'')$

and correct amount is logged: $\mathsf{CardLog}.m = m''$.

Load acquirer authenticated correctly:

No masquerade: We have $\mathsf{Shared}(K_{LI}, \mathsf{IssuerLog}.lda)$.

Security conditions (2)

ml supposed to prove that load acquirer owes transaction amount to issuer:

Acquirer guarantee functionality: If

$$\mathsf{IssuerLog}.ml = \mathbf{Mac}(\mathsf{IssuerLog}.r, (ic', cep', nt', lda', m', s1', hl'))$$

then the LSAM lda' has received m'.

Check that ml issued correctly:

Acquirer guarantee security:
If card and issuer are in final state and $\mathsf{CardLog}.m = m'$ then

$$\mathsf{IssuerLog}.ml = \mathbf{Mac}(\mathsf{IssuerLog}.r, (ic', cep', nt', lda', m', s1', hl')).$$

Results (1)

Theorem. Acquirer guarantee functionality not provided.

ml protected by random value r protected by secret key K_{LI} shared between load acquirer and card issuer.

Issuer could manufacture ml and r.

Cannot prove that load acquirer manufactured ml.

Results (2)

Theorem. Correct amount, **No Masquerade**, and **Acquirer guarantee security** hold.

Correct amount: Show that K_{CI} shared between card and issuer establishes end-to-end security.

No masquerade: Show that load device identifier corresponds to device with which issuer shares K_{LI}.

Acquirer guarantee security: Show that integrity of information between card and issuer preserved.

Related Work

Verification of smart-card payment systems
(Anderson 1999), (Stepney, Cooper, Woodcock 2000)

CEPS purchase protocol using AutoFocus
(Jürjens, Wimmel 2001)

UML for security (Jürjens FASE'01, IWSecP'01, VIS'01)

Conclusion, Future Work

Audit security of CEPS load protocol using object-oriented modelling language UML.

Benefits of approach:

- security in context of system development
- widely used notation

CEPS weakness: no proof of transaction for issuer from load acquirer.

Future work: further parts, other security properties (fail-safety); tool support.