*Vorlesung*

# *Methodische Grundlagen des Software-Engineering*

im Sommersemester 2013

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

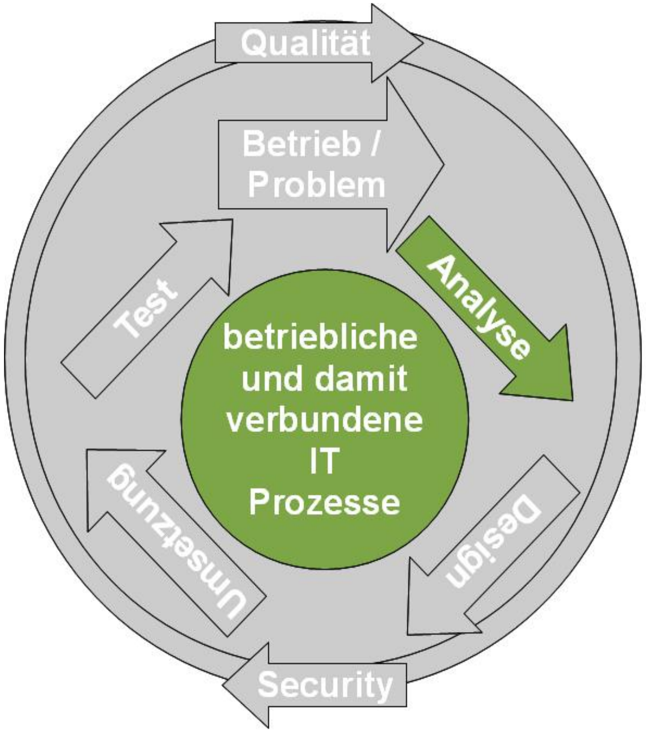
## 3.7: CEPS Purchase

v. 26.06.2013

# 3.7 CEPS Purchase

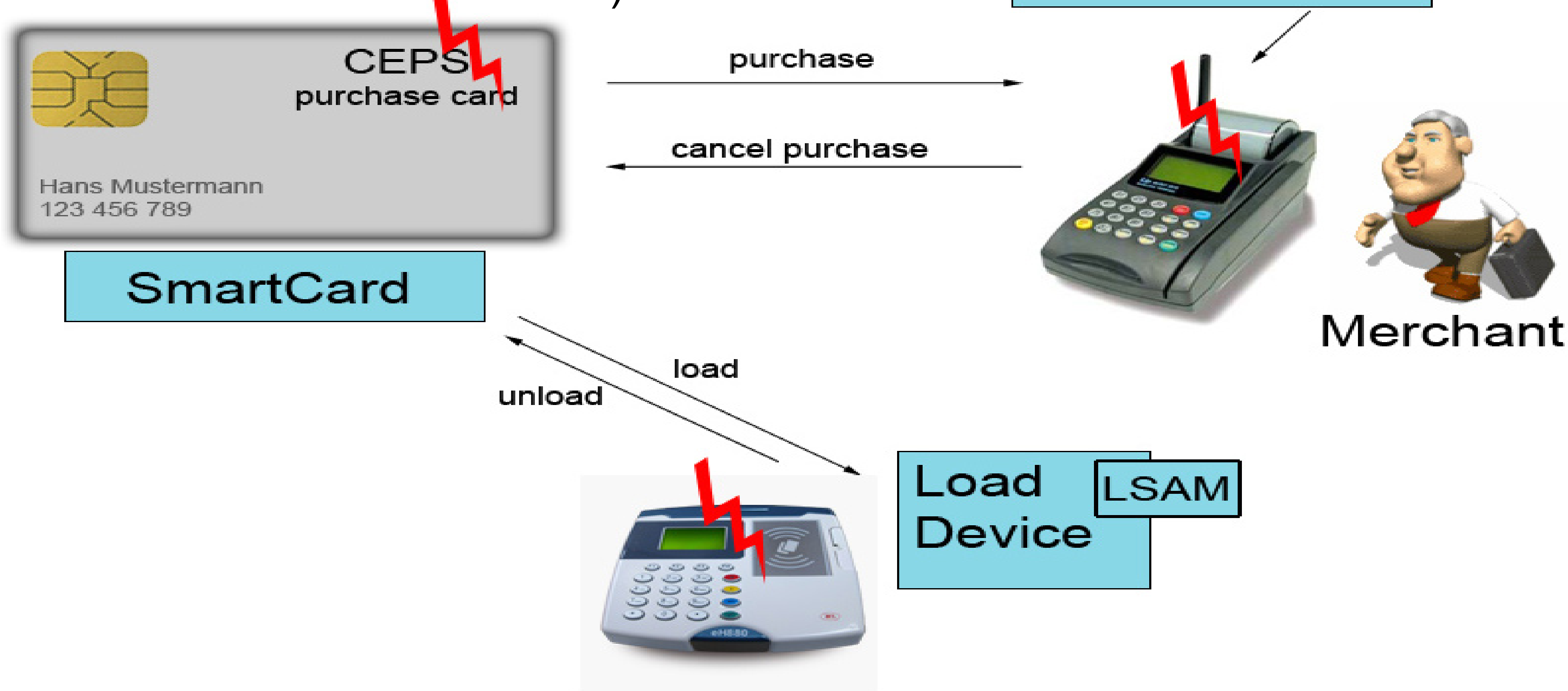
# Einordnung
## Sicheres Software Design

- Geschäfts-Prozesse
- Modelbasierte Softwareenwicklung
- Sicheres Software Design
  - Sicherheitsanforderungen
  - UMLsec
  - UML-Analysis
  - Design Principles
  - Examples
    - TLS Variant
    - CEPS Purchase



# Common Electronic Purse Specifications

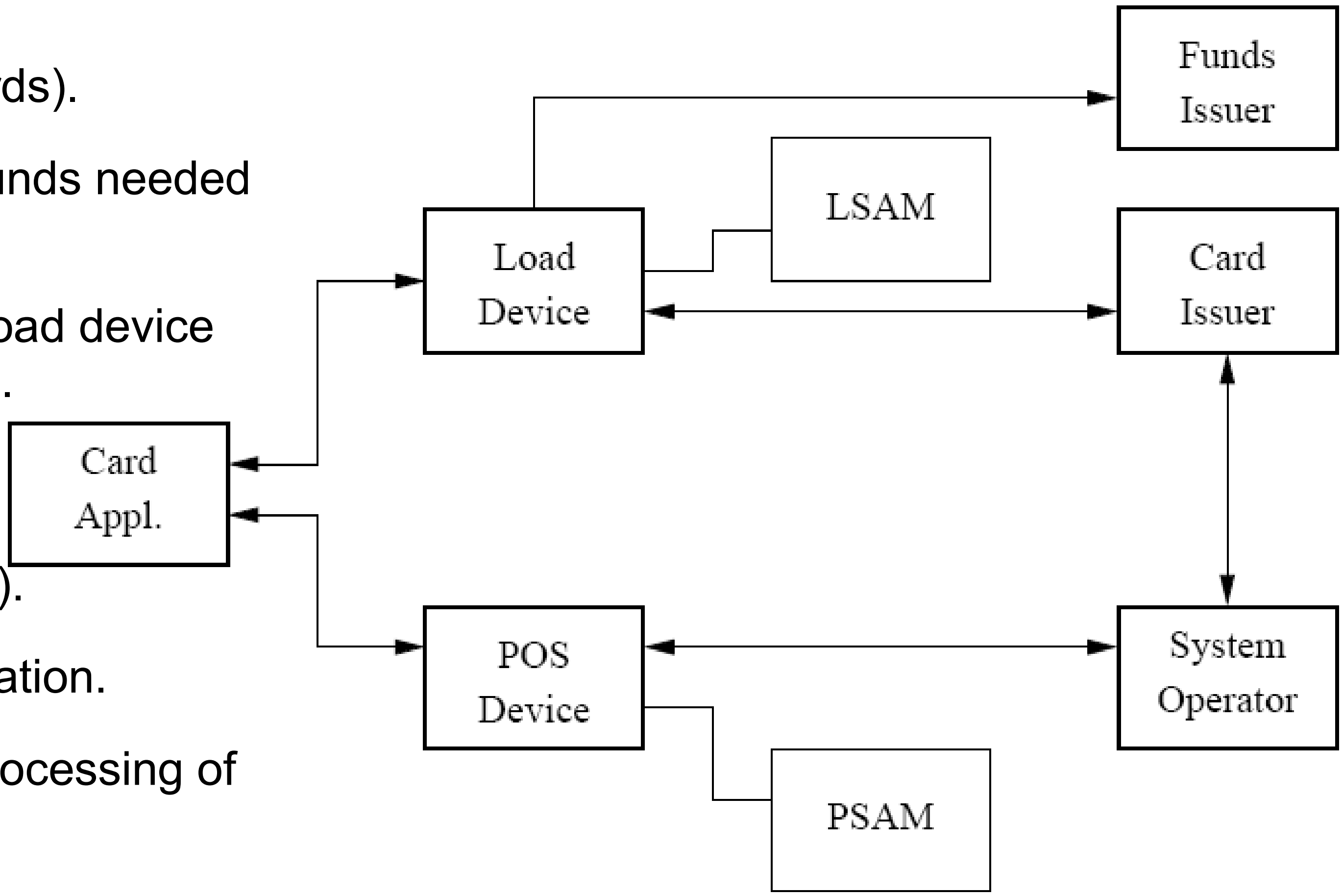
- Global electronic purse standard (90% of market).
- Smart card contains account balance. Chip secures transactions with crypto.
- More fraud protection than credit cards (transaction-bound authorization).



# CEPS Specifications Overview

Participants:

- Card issuer (issuing the cards).
- Funds issuer (processing funds needed for a linked card load).
- Load acquirer operating a load device (where card can be loaded).
- Merchant operating a POS device (where a card can be used to purchase goods).
- Card: running a card application.
- System operators for the processing of transaction data.



# CEPS Participants, Transactions

Transactions:

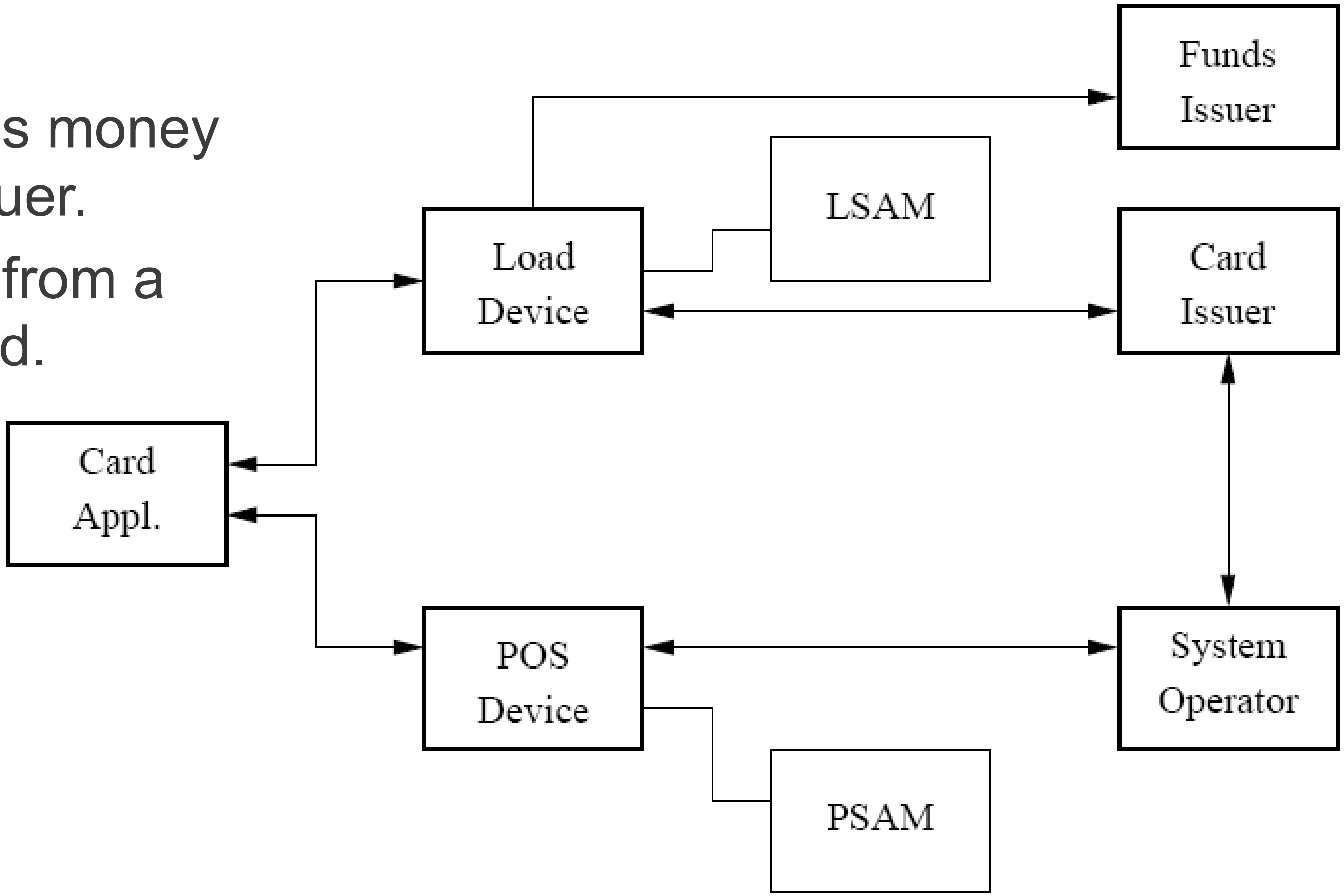
- Purchase (cardholder may purchase a good using the card).
- Purchase Reversal (merchant may reverse a purchase in case of a mistake).
- Incremental Purchase (incrementally performed purchases, e.g. phone-calls).
- Cancel Last Purchase (cardholder may cancel last purchase).
- Currency Exchange (the cardholder may exchange currencies on the card).
- Load (cardholder can load the card).
- Unload (card can be unloaded).

# CEPS Specification Resource Flow





- Cardholder loads card with money.
- Posttransaction settlement process:
  - Load acquirer sends money to relevant card issuer.
- Cardholder buys good from a merchant using his card.
- Settlement: Merchant receives correspon-ding amount of money from card issuer.



# CEPS Specification Resource Flow



LEHRSTUHL 14 SOFTWARE ENGINEERING

- CEPS designed to be aglobally interoperable standard
  - Overall transaction process may involve untrustworthy cardholders and corrupt merchants and load acquirers.
- Card issuers can take on roles of load acquirers
  - Transactions may involve competing card issuers, not trusting each other.
- Gobal situation: little hope to settle disputes using judicial means.
  - Vital: specifications requires minimal trust relations between transaction partners[1].

1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com.

# Two Central Parts of CEPS



- Purchase transaction:
  - Off-line protocol, allows the cardholder to use electronic value on a card to pay for goods.
- Load transaction:
  - On-line protocol, allows the cardholder to load electronic value on a card.

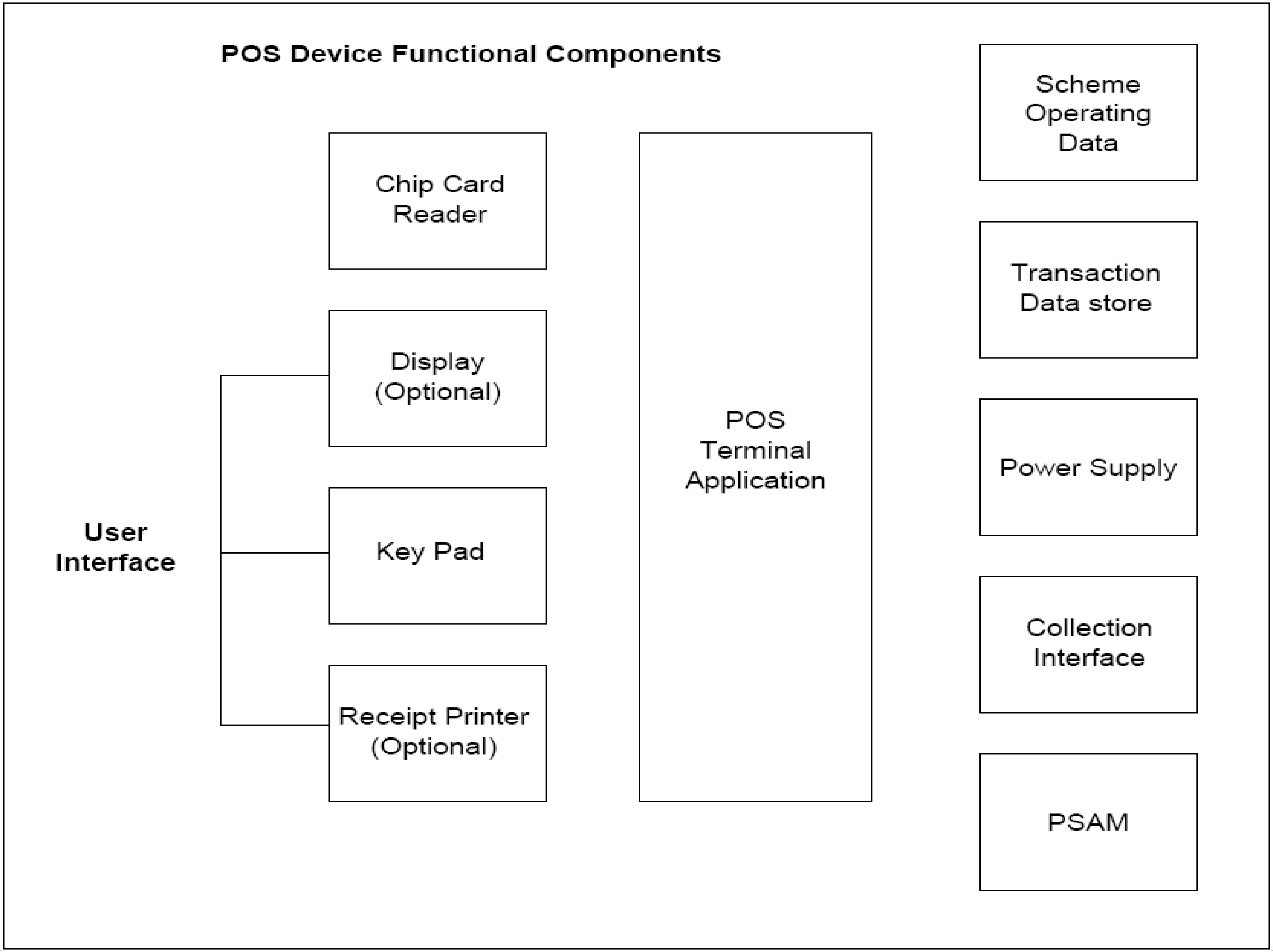
- We give a simplified account to keep presentation readable.
  - e.g. omit request messages to the smart card that are only included in the protocol, because:
  - Current smart cards communicate only by answering requests.

# Purchase Transaction

- Participants involved in off-line purchase transaction protocol:
  - Customer's card and merchant's POS device.
- POS device contains Purchase Security Application Module (PSAM)
  - To store and process data.
  - Required to be tamper-resistant.
  - Could also be implemented on a smart card.
- After protocol:
  - account balance in customer's card is decremented, and
  - balance in PSAM is incremented by corresponding amount.
- Card issuer later receives transaction logs.
- In addition to public terminals: Intended to use CEPS cards for transactions over Internet[1].

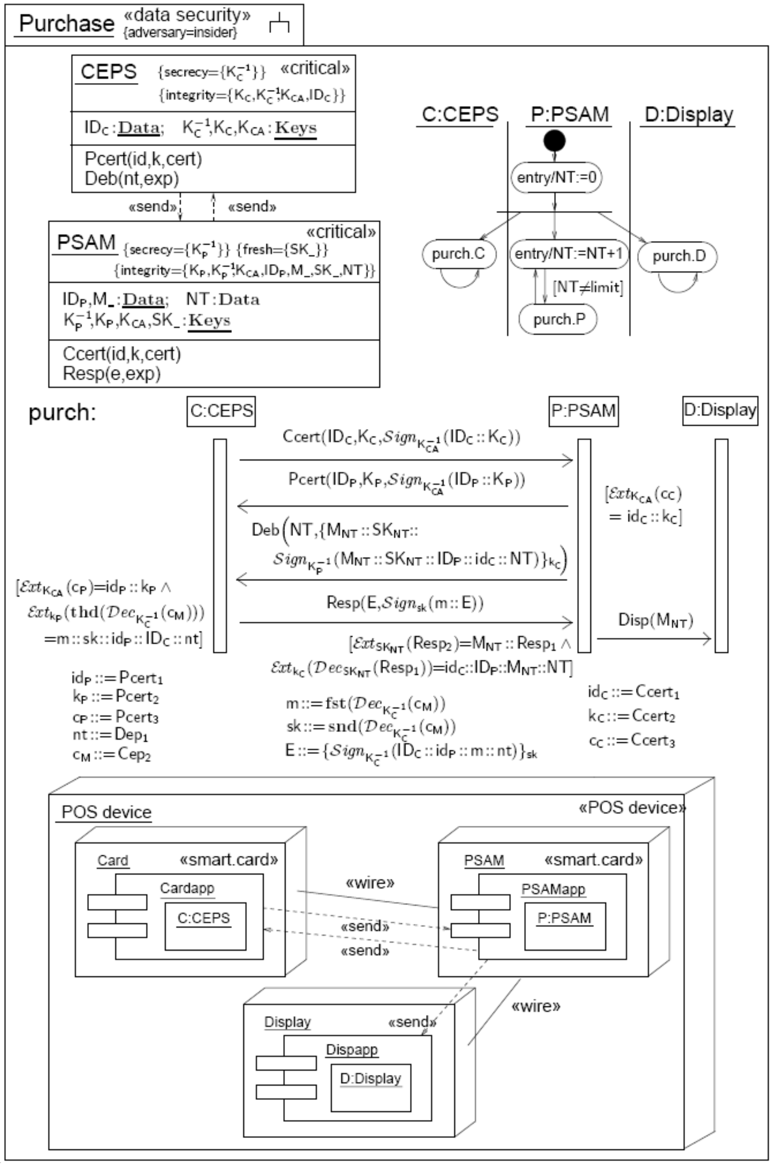
1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Bus. Req. ch. X

# POS Device Overview



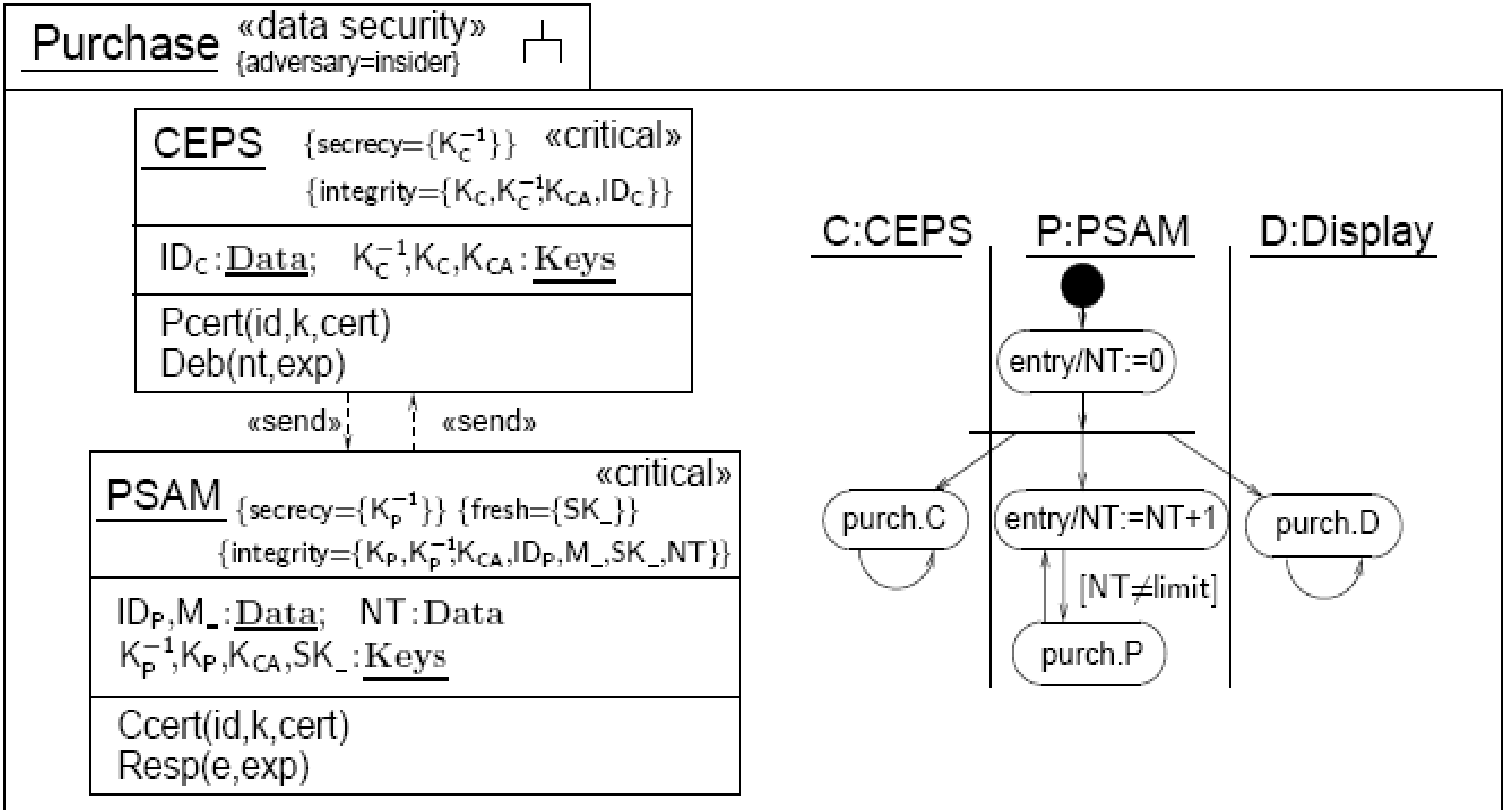

CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Tech. Spec. p. 77

# Specification: CEPS Purchase Transaction

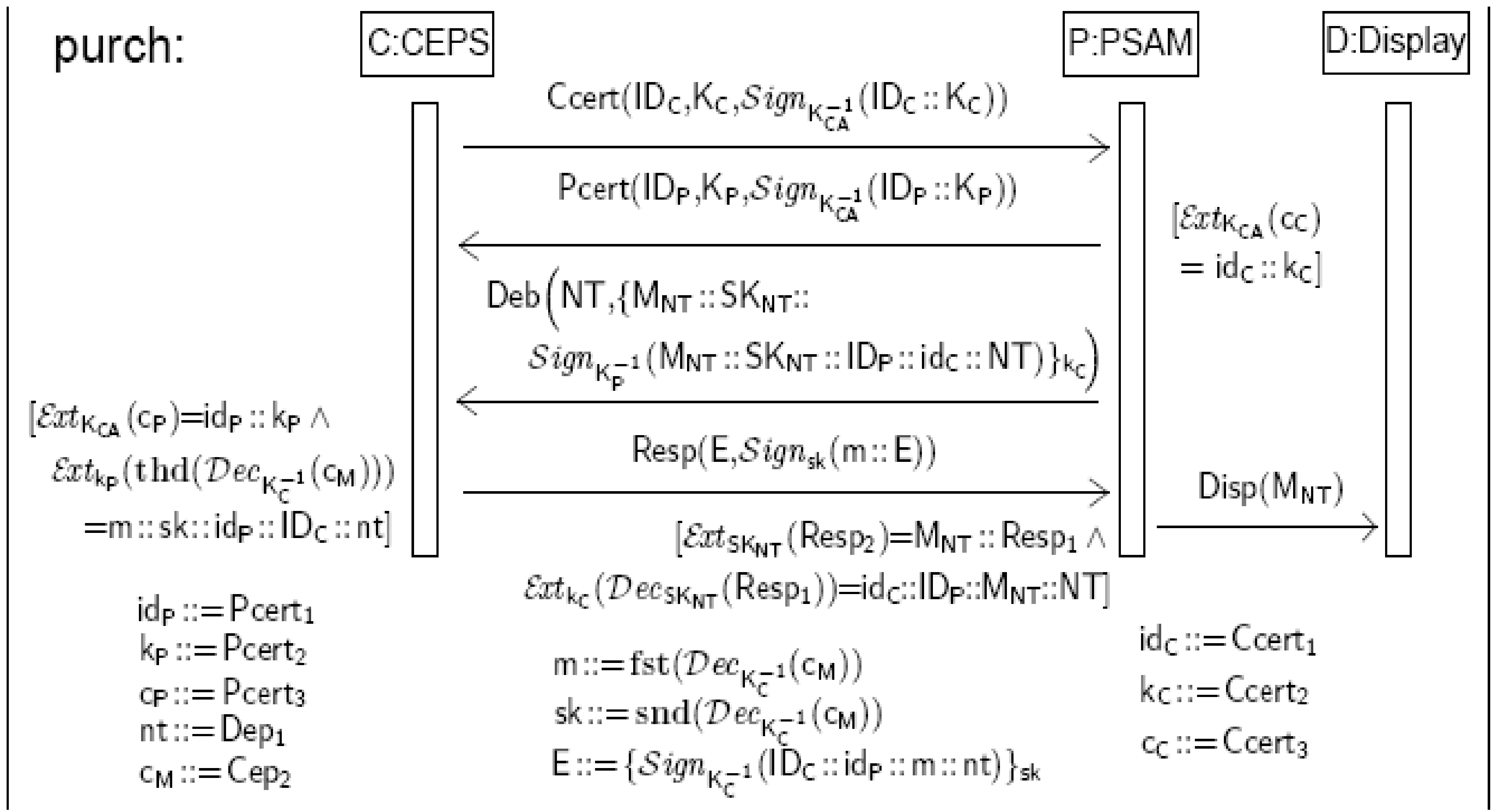


# Specification: CEPS Purchase Transaction



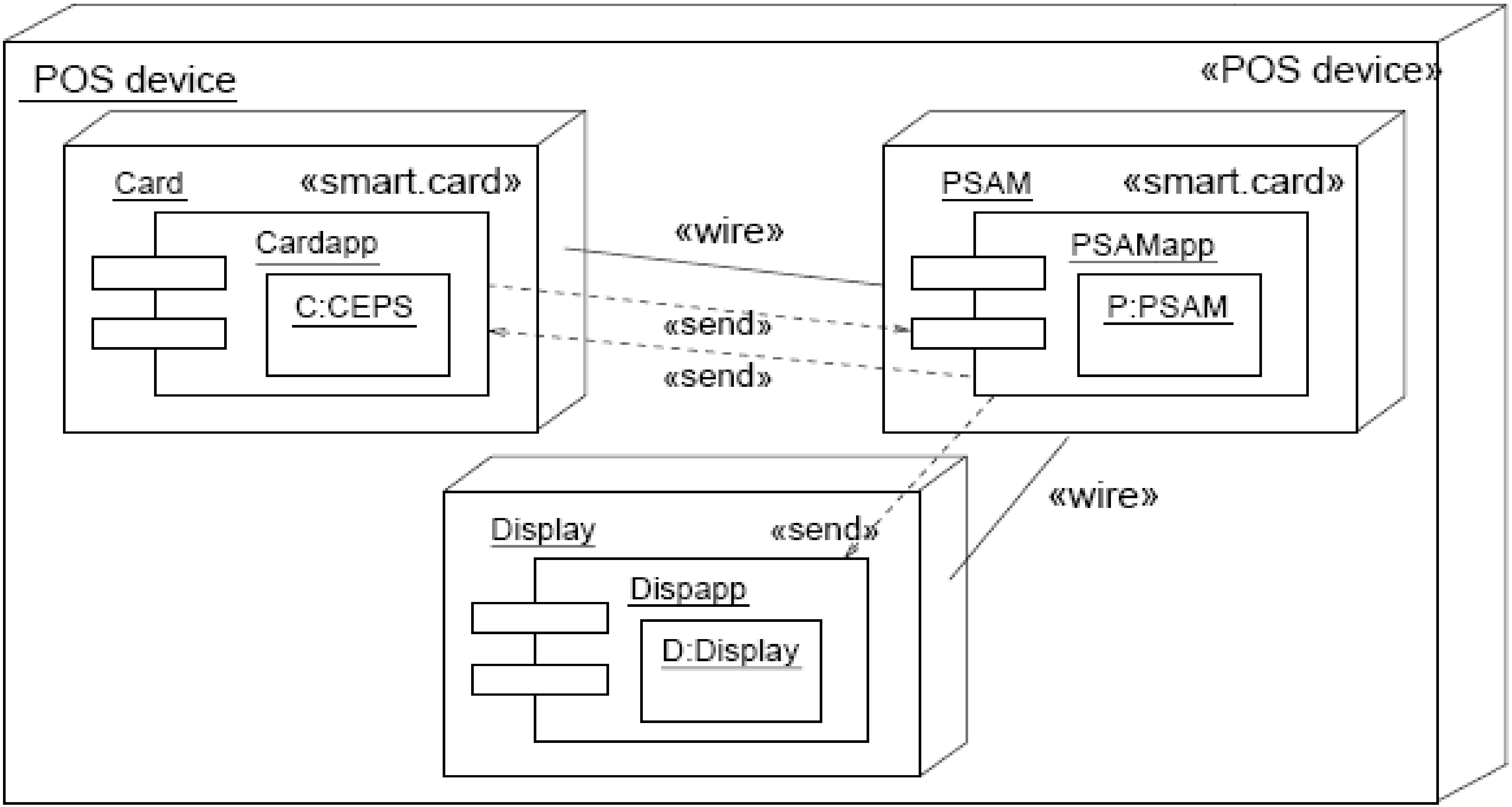


# Specification: CEPS Purchase Transaction



# Specification: CEPS Purchase Transaction





# Purchase Transaction Specification



- Specification of purchase transaction as a UML subsystem P.
- For simplicity, don't consider exception processing:
  - e.g. certificate verification fails => model simply stops further processing.
- Recall: for each method msg in diagram and each number n, $msg_n$ is the nth argument of operation call msg, most recently accepted according to sequence diagram.
- Continue to use notation var ::= exp
  - var is a shorthand for exp.

# Purchase Transaction Specification: Critical Parts

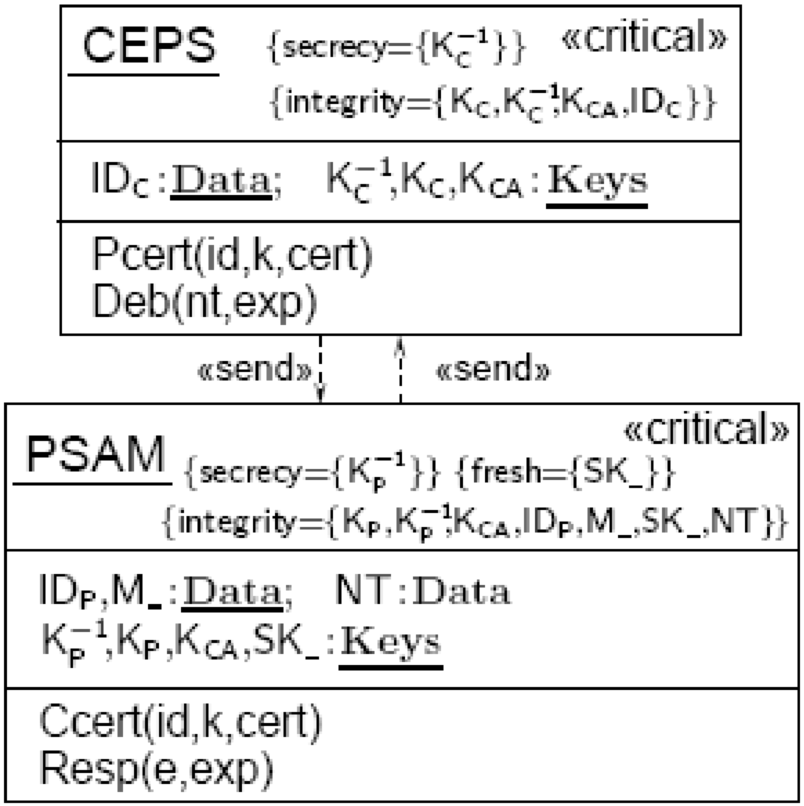
Security functionality:

- Incremental transactions (not considered here)
- Provided only by PSAM, and not the rest of PC

Protocol participants

- CEP card C, with identity $ID_C$ public/private keys $K_C$ / $K^{-1}_C$, and
- PSAM P, with identity $ID_P$ and public/private keys $K_P$ / $K^{-1}_P$.

Both have stored public key $K_{CA}$ of certification authority before transaction.




- We model the display which is security-relevant.
  - Relevant as far as cardholder can't communicate with his card directly.

# Purchase Transaction Assumptions



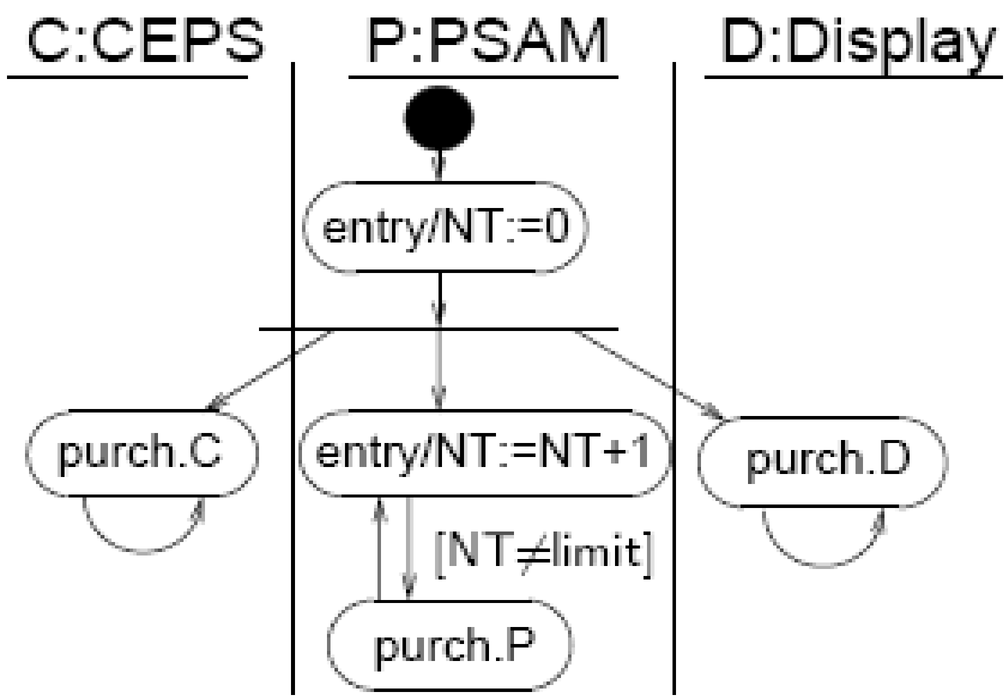
- For simplicity: Omit that protocol used with different cards during lifetime of PSAM.
    - Card revocation is not considered here.
- Given:
    - sequence of transaction amounts $M_{NT}$ indexed by transaction number NT
    - sequence of session keys $SK_{NT}$ .

      Required to be fresh at PSAM object (indicated by {fresh})[1].
- Specification: Expressions of form $SK_x$, for any subexpression x, appear only at PSAM object and the associated view of sequence diagram.
- Keys (different constant symbols in Keys) are mutually distinct => mutually independent.
- M_ denotes an array whose fields $M_x$ have type Data.
- Constant attributes have their initial values as attribute names.
    - Corresponding attribute types are underlined.

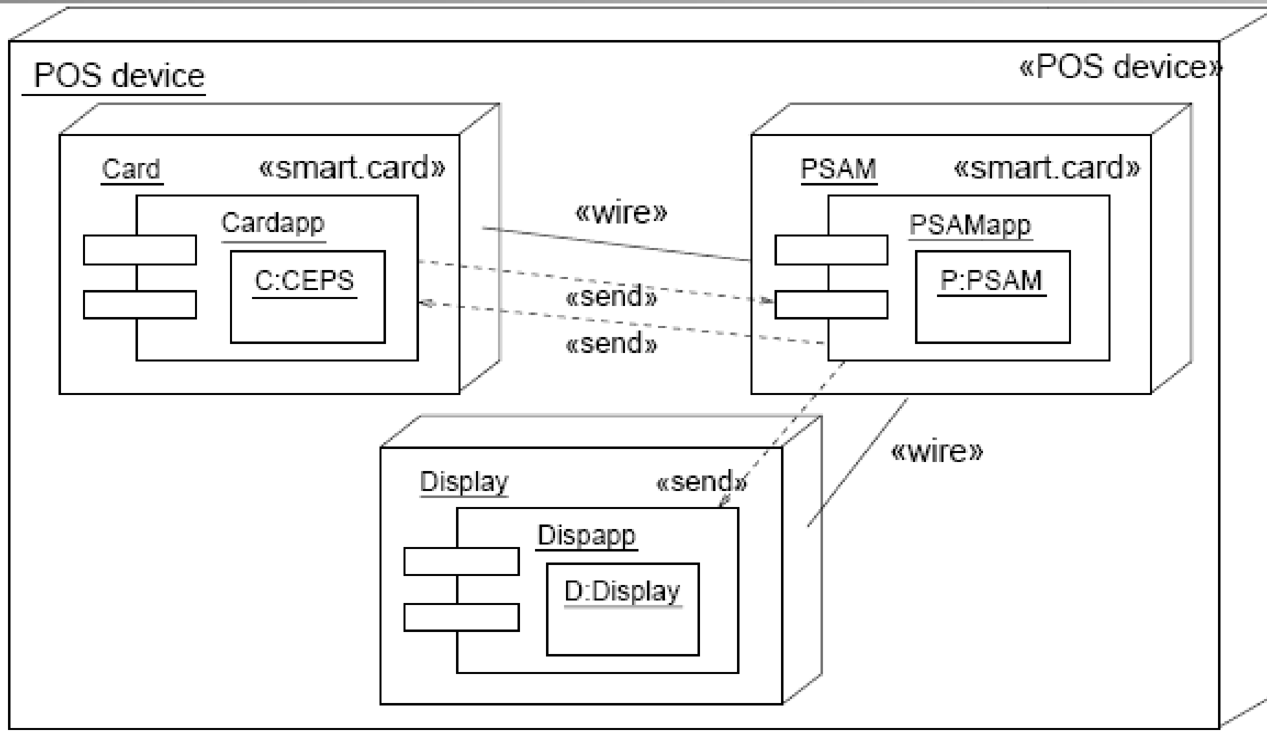
1 Jan Jürjens, Secure Systems Development with UML, Springer 2004. Sect. 4.1.2

# Purchase Transaction Protocol Execution I

- Beginning of execution in POS device,
  - PSAM creates transaction number NT with value 0.
- Before each protocol run, NT is incremented.
- If limit is exceeded, PSAM stops functioning:
  - to avoid rolling over of NT to 0.
- Note: additional operation, +:
  - to build up expressions.



# Purchase Transaction Protocol Execution II

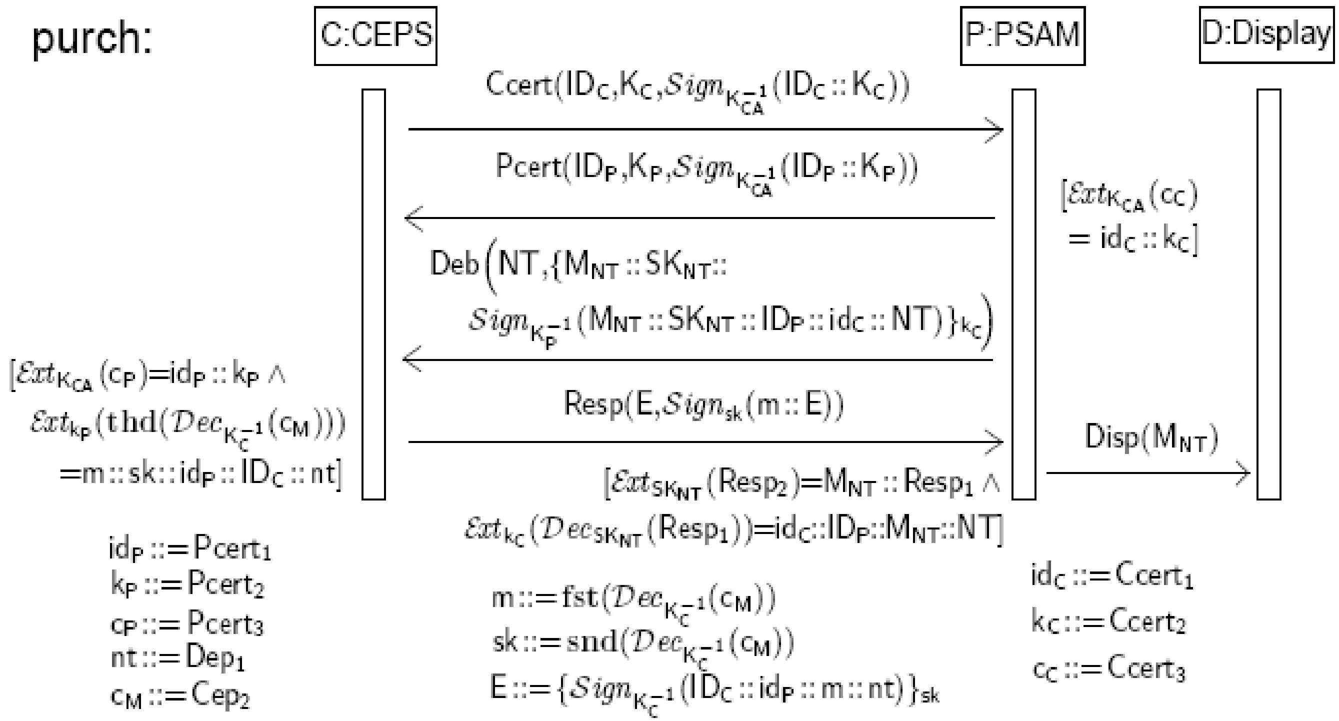



- Protocol between card C, PSAM P, and display D
  - Supposed to start after:
    - C inserted into POS device (containing P and D), and
    - amount M is communicated to PSAM.
      - by typing into a terminal (assumed to be secure).

# Purchase Transaction Protocol Execution III

- Each Protocol run consists of parallel execution of card's and PSAM's part of protocol.
- C and P begin protocol by exchanging certificates
- $ID_C$, $K_C$, $Sign_{K^{-1}_{CA}}(ID_C :: K_C)$

  $(ID_P, K_P, Sign_{K^{-1}_{CA}}(ID_P :: K_P))$
  - Containing identifier $ID_C$ ($ID_P$) and public key $K_C$ (resp. $K_P$),
  - With same information signed with $K^{-1}_{CA}$.
- Both check validity of received certificate.
  - Check: signature consists of received identifier and public key.
    - Signed with $K^{-1}_{CA}$ , by verifying signature with key $K_{CA}$.



# Consideration: Knowledge of C



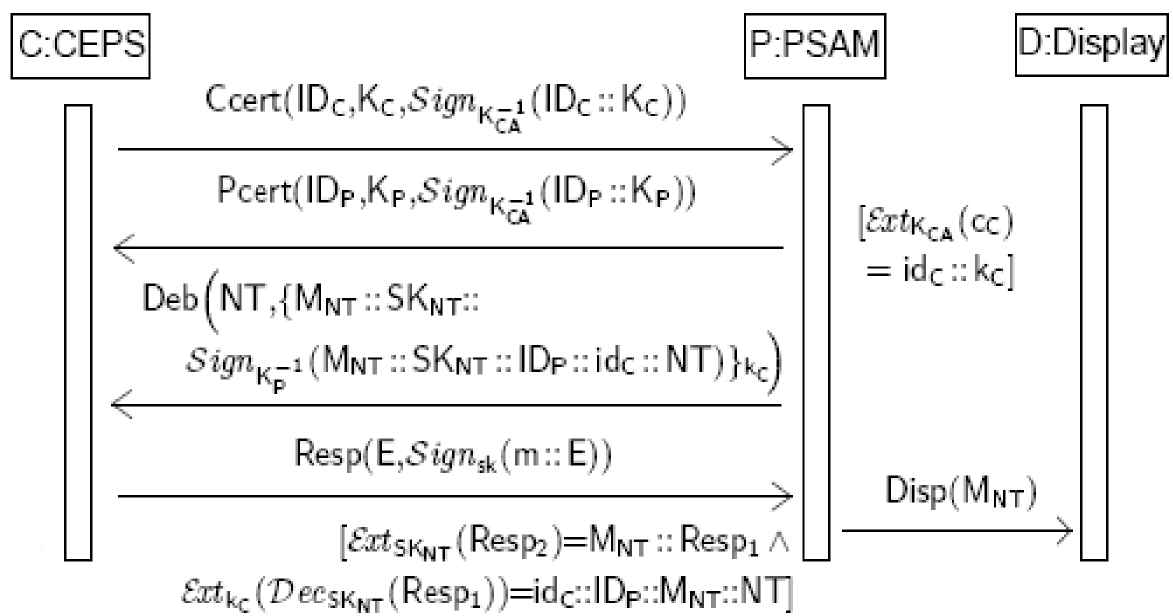


Note

- C "knows" that it has received a valid certificate,
- C does not know whether it has received certificate for P at present physical location,
  - because C has no information regarding identity of P that $ID_P$ itself could be verified against.

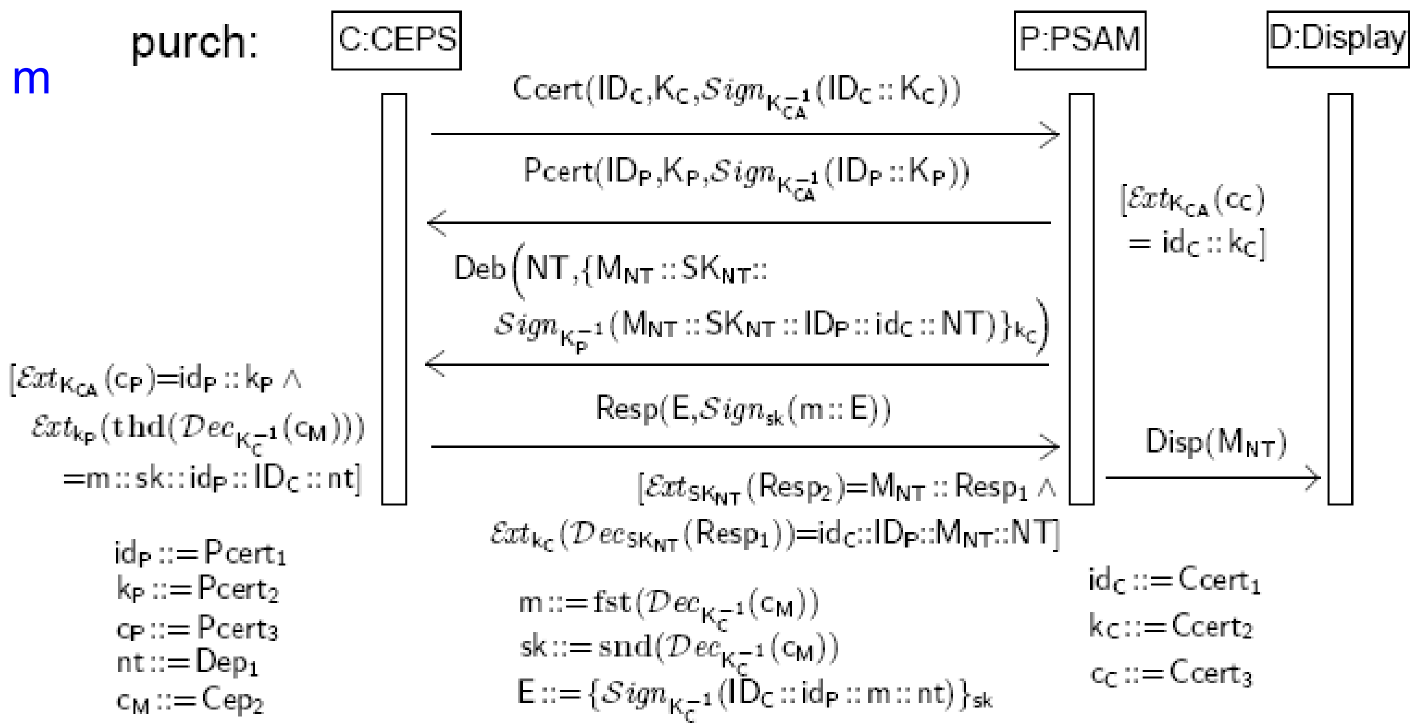
# Purchase Transaction Protocol Execution IV

- Then P sends the Debit-for-Purchase message containing:
  - transaction number NT.
  - Encryption of following data under $k_C$ received in C's certificate:
    - Concatenation of price $M_{NT}$ of good to be purchased,
    - Symmetric session key $SK_{NT}$,
  - Following data signed with $K^{-1}_P$ :
    - Amount $M_{NT}$,
    - Key $SK_{NT}$,
    - P's identifier $ID_P$,
    - data $id_C$ earlier received as C's identifier,
    - transaction number NT.



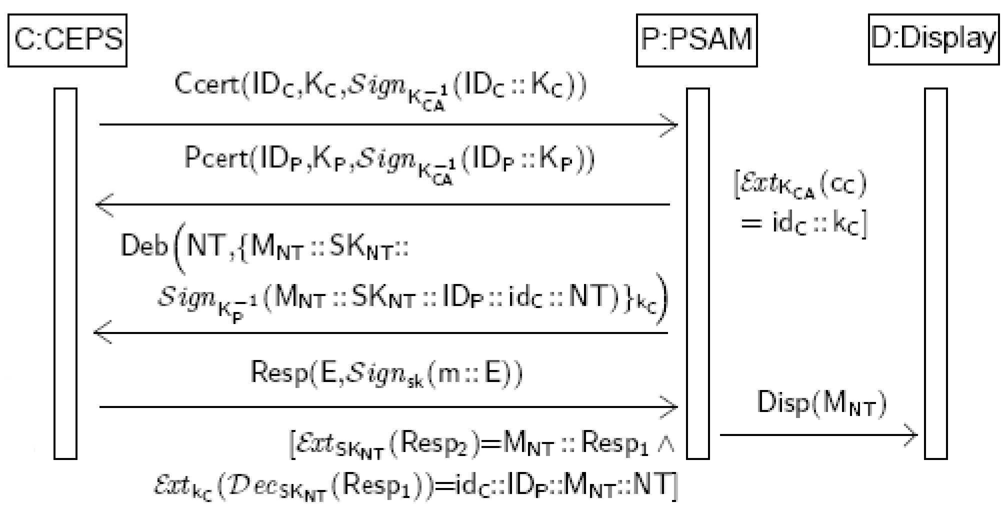
# Purchase Transaction Protocol Execution V

- C checks validity of signature with $k_P$ (earlier received) against
  - received data amount m
  - received key sk
  - received identifier $id_P$
  - own identifier $ID_C$,
  - received transaction number nt.
- Then C returns,
  - E which consists of $ID_C$, $id_P$,m,and nt, signed with $K^{-1}_C$ and encrypted under sk
  - secondly, m and E signed with sk.



# Purchase Transaction Protocol Execution VI

- P verifies:
  - Second part of received message is concatenation of $M_{NT}$ sent out previously and first part of message, signed with $SK_{NT}$,
  - first part of message, after decryption with $SK_{NT}$, gives signature of concatenation of $id_C$, $ID_P$, $M_{NT}$, and NT.
  - If all verifications succeed: protocol finishes.
    - Otherwise execution stops at failed verification.



# Security Threat Model

- CEPS:
  - Require smart card and PSAM to be tamper-proof.
  - But **not** the POS device[1].
- Purchase transaction: supposed to provide mutual authentication between terminal and card using
  - Certificate issued by a certification authority
  - Card's or PSAM's public key.

# Security Threat Model
# Cardholder Security

- Smart card inserted into POS device
  - Can communicate with PSAM.
- No direct communication between cardholder and card.
- Info displayed by POS device has to be trusted at point of transaction.
  - Security against fraud by merchant supposed to be provided by:
    - checking card balance after transaction.
    - complaining to merchant, and if necessary to card issuer.
      - in case of incorrect processing.

# Security Threat Model
# Merchant Security

- Security against customer:
  - supposed to be provided by exchanging purchased good only for a signed message from card containing transaction details:
    - for which merchant will receive corresponding monetary amount from the issuer in settlement process afterwards.
  - More precisely:
    - merchant possessing PSAM with identifier $ID_P$
      - when presenting signature $E = Sign_{K^{-1}_{A}}(ID_C::ID_P::M_{NT}::NT)$.

      receive monetary amount $M_{NT}$ from account of cardholder with identifier $ID_C$, once for each $NT$.
      - $K_C$ is key for $ID_C$.

# Security Threat Model
# Main Idea





- Keep risk of fraud is small since
  - Fraud should be either prevented or at least later detected in settlement.
  - Certificates of cards or PSAMs actively involved in fraud can be revoked using revocation lists (treatment omitted here).
- Kinds of fraud can only be detected after transaction.
  - e.g. cardholder unable to communicate with card directly to authorize transaction.
    - POS device could charge a higher amount than shown.

# Security Threat Model
# Three Security Goals





- Cardholder security:
  - Merchant can only claim amount registered on card after transaction
    - can be checked with cardholder's cardreader.
- Merchant security:
  - Merchant receives valid signature in exchange for sold good.
- Card issuer security:
  - Sum of balances of all valid cards and all valid PSAMs remains unchanged by transaction.

- Beware:
  - Protocol also expected to be used over Internet.
  - POS device,
    - Providing communication link between card and PSAM

    not considered to be within security perimeter.

# Security Threat Model Formalized: Cardholder Security



- Call $K_X$ valid for a card or PSAM with identifier $ID_X$

  if $Sign_{K^{-1}_{CA}}(ID_X :: K_X)$ in participant's knowledge.

- Cardholder security:
  - For all $ID_C$, $ID_P$, $M_{NT}$, $NT$, $K^{-1}_C$
    - such that $K_C$ valid for $ID_C$,

    if $P$ is in possession of $Sign_{K^{-1}_{CA}}(ID_C::ID_P::M_{NT}::NT)$
  - Then $C$ is in possession of $Sign_{K^{-1}_P}(M_{NT} :: SK_{NT} :: ID_P :: ID_C :: NT)$,
  - For some $SK_{NT}$ and $K^{-1}_P$
  - Such that corresponding $K_P$ valid for $ID_P$.

# Security Threat Model Formalized: Merchant Security

Merchant Security:

- Each time D receives $M_{NT}$, P is in possession of
  - $Sign_{K^{-1}_{CA}}(ID_C :: K_C)$ and
  - $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT)$
- for some $ID_C$, $K^{-1}_C$ , and new value NT.

# Security Threat Model Formalized: Card Issuer Security

Card Issuer Security

- After completed purchase transaction.
  - Let $S$ be sum of all $M_{NT}$ in sequence, of processed elements of form $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT)$ over all expressions $ID_C$, $ID_P$, and $K^{-1}_{C}$ .
    - such that corresponding $K_C$ valid for $ID_C$
    - and where $NT$ are mutually distinct for fixed $C$.
  - Let $S'$ be sum of all $M'_{NT'}$ in sequence of processed $Sign_{K^{-1}_{P'}}(M'_{NT'}::SK'_{NT'}::ID_{C'}::ID_{P'}::NT')$over all expressions $ID_{C'}$ ,$ID_{P'}$ ,$K^{-1}_{P'}$ .
    - such that corresponding $K_{P'}$ is valid for $ID_{P'}$
    - and where $NT'$ are mutually distinct for fixed $C'$.
- Then $S$ is no greater than $S'$.

# Results

- According to assumptions in CEPS, we consider attacker able to:
  - access POS device links.
  - access other PSAMs over Internet,
  - but is **not** able to tamper with smart cards.

That is, what we consider the insider attacker.

| Stereotype | $\mathsf{Threats}_{insider}()$ |
|---|---|
| Internet | {delete, read, insert} |
| encrypted | {delete, read, insert} |
| LAN | {delete, read, insert} |
| wire | {delete, read, insert} |
| smart card | $\emptyset$ |
| POS device | $\emptyset$ |
| issuer node | {access} |

# Vulnerability

- Current threat scenario:
  - Weakness with regards to goal of merchant securit arising from facts that:
    - POS device is not secured against potential attacker that may try to betray merchant.
    - CEPS to be used over Internet.
    - Attacker could be employee. (realistic scenario).

- First sketch idea of attack informally
- Then exhibit attacker within formal model.

# Informal Description of the Attack

- Attacker redirects messages between card C and the PSAM P to another PSAM P'
  - e.g. Buy electronic content and let cardholder pay for it.
- Assume: Attacker manages to have amount payable to P' equal the amount payable to P.
- Attacker also sends required message to display.
  - Display will reassure merchant that required amount has been received.

# Informal Description of the Attack



- Attack has a good chance of going undetected:
  - Cardholder won't notice anything suspicious
    - deducted amount is correct.
  - C registers identifier $id_{P'}$ rather than $id_P$,
    - Identifiers are non-self-explanatory data.
    - Cardholder cannot be assumed to verify
    - C has no information about what identity of P should be.
    - Identifier $id_C$ in Deb message is as expected
      - P' correctly assumes to be in transaction with C.
  - Merchant who owns P will notice later lacking amount of $M_{NT}$.
- Note: P not involved in this attack.

# Attack
# Message Flow Diagram

$E := \{Sign_{K_C^{-1}}(ID_C::ID_P::M_{NT}::NT)\}_{sk}$.

$$C \xrightarrow{\mathsf{Ccert}(ID_C, K_C, \mathcal{S}ign_{K_{CA}^{-1}}(ID_C::K_C))} A \xrightarrow{\mathsf{Ccert}(ID_C, K_C, \mathcal{S}ign_{K_{CA}^{-1}}(ID_C::K_C))} P'$$

$$C \xleftarrow{\mathsf{Pcert}(ID_{P'}, K_{P'}, \mathcal{S}ign_{K_{CA}^{-1}}(ID_{P'}::K_{P'}))} A \xleftarrow{\mathsf{Pcert}(ID_{P'}, K_{P'}, \mathcal{S}ign_{K_{CA}^{-1}}(ID_{P'}::K_{P'}))} P'$$

$$A \xleftarrow{\mathsf{Deb}(NT, \{M_{NT}::SK_{NT}::\mathcal{S}ign_{K_{P'}^{-1}}(M_{NT}::SK_{NT}::ID_{P'}::id_C::NT)\}_{k_C})} P'$$

$$C \xleftarrow{\mathsf{Deb}(NT, \{M_{NT}::SK_{NT}::\mathcal{S}ign_{K_{P'}^{-1}}(M_{NT}::SK_{NT}::ID_{P'}::id_C::NT)\}_{k_C})} A$$

$$C \xrightarrow{\mathsf{Resp}(E, \mathcal{S}ign_{sk}(m::E))} A \xrightarrow{\mathsf{Resp}(E, \mathcal{S}ign_{sk}(m::E))} P'$$

$$A \xrightarrow{\mathsf{Disp}(M_{NT})} D$$

# Simplified Attack



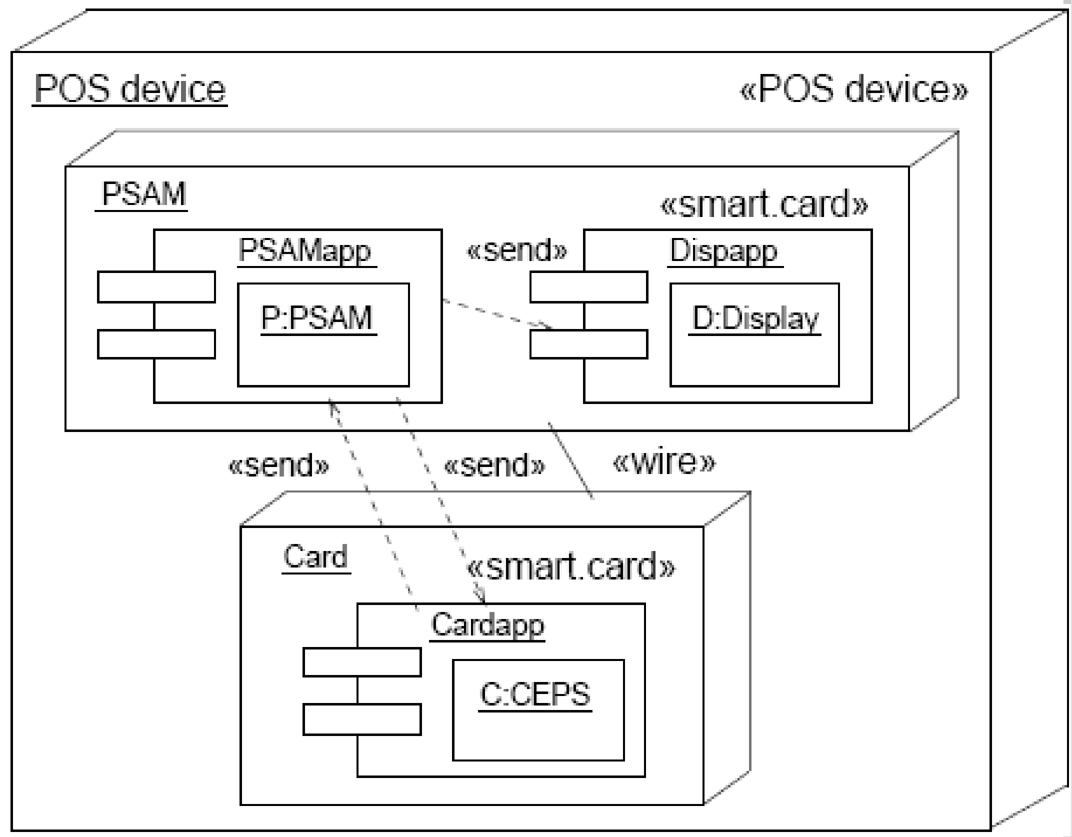
Note:

- Simpified attack if attacker can eavesdrop on connection between terminal (where $M_{NT}$ is entered) and PSAM P.
  - Attacker only has to intercept $M_{NT}$.
  - Redirect all messages from C to P' and back.
  - Finally send Disp($M_{NT}$) to display.
- If in addition assume: Cardholder coincides or collaborates with attacker
  - Attacker could remove $M_{NT}$ and send Disp($M_{NT}$) to the display,
  - Cardholder receives good without having to pay for it.

# Proposed Solution

Problem can be solved by:

- Securing communication link between PSAM and display.
  - e.g. using smart card with integrated display as PSAM.
- Ensure this PSAM cannot be replaced without being noticed.
- Leads to specification P' with modified deployment diagram and otherwise unchanged protocol specification.



# Security of Improved Protocol Security Properties

- Argue that specification provieds security properties against insider adversaries.
- Proposition: P' provides secrecy of $K^{-1}_{C}$ , $K^{-1}_{P}$ and integrity of $K^{-1}_{C}$ , $K_C$ , $K_C$ , $ID_C$ , $K^{-1}_{P}$ , $K_P$ , $M_{NT}$ , $SK_{NT}$ , NT
  - Meaning: Adversary should not be able to make atttributes take on values previously known only to him.

against insider adversaries with $\mathbf{K}^{A}_{P} \cap \{K^{-1}_{C}, K^{-1}_{P}\} = \emptyset$.

# Security of Improved Protocol Security Properties: Proof I





- For adversary adv to gain knowledge of $K^{-1}_C$ , $K^{-1}_P$ .
  - adv would have to read these from one of the two communication links.
- Consider: Is at any point any of the expressions communicated over any of the two communication links.
- According to specification none of the values is output by any protocol participants at any time.
- Therefore secrecy of $K^{-1}_C$ , $K^{-1}_P$ is provided
  - since values never sent outside smart cards (assumed to be impenetrable).

# Security of Improved Protocol Security Properties: Proof II

- For adv to violate integrity of any attribute
  - $K^{-1}_C$ , $K_C$, $K_{CA}$, $ID_C$, $K^{-1}_P$ , $K_P$, $M_{NT}$, $SK_{NT}$.

  adv would have to cause their values take on atomic value in $Data^a$, during interaction with protocol participants.
  - Their values would have to change.
- Protocol specification: Value of none of these attributes changed during protocol execution.
- Thus integrity preserved.

# Security of Improved Protocol Security Properties: Proof III



- For adv to violate integrity of NT,
  - adv would have to cause value take on atomic value in $Data^a$, during interaction with protocol participants.
- From protocol specification: Value of NT changed only to take on values of form
  0, 0 + 1, 0 + 1 + 1, etc., all of which are not in $Data^a$.
- Thus integrity of NT is preserved.

# Further Vulnerability

Note:

- Proposition doesn't imply that C and P terminate protocol with same value for $M_{NT}$.
  - Cannot be guaranteed, since a „redirection attack" similar to above still applies.
- Display can no longer be manipulated.
  - Would be noticed if PSAM received less money than expected.
- Money could still come from different card than inserted into POS device.
- Kinds of integrity property relevant here considered as:
  - „Cardholder security".
  - „Merchant security".

# Notes

Note:

- Secure definition of $M_{NT}$,
  - outside current specification.

  relies on secure connection between terminal (amount entered) and PSAM.
- Creation of session keys $SK_{NT}$ is outside current scope.
  - Values assumed to be given.

- Theorem: Consider adv of type A = insider with
  - $\mathbf{K}^{p}_{A} \cap (\{K^{-1}_{C}, K^{-1}_{P}, K^{-1}_{CA}\} \cup \{SK_{NT} : NT \in \mathbb{N}\}$
  - $\cup \{Sign_{K^{-1}_{P}}(E) : E \in Exp\} \cup \{Sign_{K^{-1}_{C}}(E) : E \in Exp\}$
  - $\cup \{Sign_{SK_{NT}}(E) : E \in Exp \wedge NT \in N\}) = \emptyset$.

  and such that for each $X \in Exp$ with $Sign_{K^{-1}_{CA}}(X::K) \in \mathbf{K}^{p}_{A}$, $X = ID_C$ implies $K = K_C$ and $X = ID_P$ implies $K = K_P$.
- Following security guarantees provided by P' in presence of adv of type A:
  - Cardholder security.
  - Merchant security.
  - Card issuer security.

# Security Threat Model Formalized: Cardholder Security



Cardholder security:

- For all $ID_C$, $ID_P$, $M_{NT}$, $NT$, $K^{-1}_C$
  - such that $K_C$ valid for $ID_C$,

  if $P$ is in possession of $Sign_{K^{-1}_{CA}}(ID_C::ID_P::M_{NT}::NT)$
- Then $C$ is in possession of $Sign_{K^{-1}_P}(M_{NT} :: SK_{NT} :: ID_P :: ID_C :: NT)$,
- For some $SK_{NT}$ and $K^{-1}_P$
- Such that corresponding $K_P$ valid for $ID_P$.

# Security Threat Model Formalized: Merchant Security

Merchant Security:

- Each time $D$ receives $M_{NT}$, $P$ is in possession of
  - $Sign_{K^{-1}_{CA}}(ID_C :: K_C)$ and
  - $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT)$
- for some $ID_C$, $K^{-1}_C$ , and new value $NT$.

# Security Threat Model Formalized: Card Issuer Security

Card Issuer Security:

- After completed purchase transaction.
  - Let $S$ be sum of all $M_{NT}$ in sequence, of processed elements of form $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT)$ over all expressions $ID_C$, $ID_P$, and $K^{-1}_{C}$ .
    - such that corresponding $K_C$ valid for $ID_C$
    - and where $NT$ are mutually distinct for fixed $C$.
  - Let $S'$ be sum of all $M'_{NT'}$ in sequence of processed $Sign_{K^{-1}_{P'}}(M'_{NT'}::SK'_{NT'}::ID_{C'}::ID_{P'}::NT')$over all expressions $ID_{C'}$ ,$ID_{P'}$ ,$K^{-1}_{P'}$ .
    - such that corresponding $K_{P'}$ is valid for $ID_{P'}$
    - and where $NT'$ are mutually distinct for fixed $C'$.
- Then $S$ is no greater than $S'$.

# Security of Improved Protocol
# Proof: Cardholder Security

Cardholder Security: Proof by contraposition.

- Suppose
  - $\forall\ SK_{NT}, K^{-1}_{P}$ such that corresponding $K_P$ valid for $ID_P$

  $C$ not in possession of $Sign_{K^{-1}_{P}}(M_{NT} :: SK_{NT} :: ID_P :: ID_C :: NT)$.
- Like to show that
  - $\forall\ K^{-1}_{C}$ such that corresponding $K_C$ is valid for $ID_C$,

  $P$ is not in possession of $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT)$.
- Fix such $ID_C$, $K_C$, and $K^{-1}_{C}$ .

# Security of Improved Protocol Proof: Cardholder Security

- Consider:
  - Joint knowledge set **K**, all participants except C.
    - objects P, D, and any given adv, which w.r.t. scenario are not able to penetrate smart card on which C resides) and
  - Knowledge set $\mathbf{K}_C$ of C.

**Claim**. K is contained in every subalgebra X of Exp containing

- $Keys \setminus \{K^{-1}_C\} \cup \mathbf{K}^p_A \cup Data \cup$

$\{\{Sign_{K^{-1}_C}(ID_C :: id_P ::m:: nt)\}_{sk},$

$Sign_{sk}(m :: \{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk}) :$

$id_P, k_P, m, sk, nt, E \in \mathbf{K}_C \wedge Sign_{K^{-1}_{CA}}(id_P :: k_P) \in \mathbf{K}_C$

$\wedge\ Ext_{k_P}(E) = m:: sk :: id_P :: ID_C :: n\}.$

# Security of Improved Protocol Proof: Cardholder Security



Note:

- $Sign_{sk}(m :: \{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk})$ redundant.
  - But included for explicitness.
- Not claimed that **K** is intersection of such algebras.
  - e.g. any of above algebras (and thus their intersection) contains key $K^{-1}_{CA}$, although **K** does not.
    Latter fact is nevertheless used in proof (later when using the claim).
    - similar remark applies to terms of form $Sign_{K^{-1}_{CA}}(ID :: K)$.
  - **K** contains $SK_{NT}$, but not $K^{-1}_C$(shown later).

## Proof of Claim

- Claim holds because knowledge set **K** by definition subalgebra of the algebra of Exp built up from initial knowledge by protocol participants except C and any adversary in interaction with C.
- Have to consider:
  - What knowledge other participants can gain from interaction with C.
- Expressions learned from first message from C contained in X
  - Because X assumed to contain all
    - keys $K \in Keys \setminus \{K^{-1}_C\}$,
    - and all data in Data.

# Security of Improved Protocol Proof: Cardholder Security

- Proof of Claim
- Expressions learned from second message from C are contained in X
  - because X assumed to contain
    - $\{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk}$ and $Sign_{sk}(m :: \{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk})$

    for all $id_P, k_P \in \mathbf{K}_C$ with
    - $Sign_{K^{-1}_{CA}}(id_P :: k_P) \in \mathbf{K}_C$ and $m, sk, nt, E \in \mathbf{K}_C$ with
      - $Ext_{k_P}(E) = m :: sk :: id_P :: ID_C :: nt$.
  - and because C must receive values
    - $id_P$, $k_P$, $Sign_{K^{-1}_{CA}}(id_P :: k_P)$, m, sk, nt, E

    before sending out messages
    - $\{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk}$ and $Sign_{sk}(m :: \{Sign_{K^{-1}_C}(ID_C :: id_P :: m :: nt)\}_{sk})$.

# Security of Improved Protocol Proof: Cardholder Security

- In particular: $K^{-1}_C \notin \mathbf{K}$ because:
  - Initial knowledge of P, D.
  - and adversary does not include $K^{-1}_C$.
    - and it (or anything it could be derived from) is not transmitted.
- Under assumption: $Sign_{K^{-1}_P}(M_{NT} :: SK_{NT} :: ID_P :: ID_C :: NT) \notin \mathbf{K}_C$
  - for any $SK_{NT}$, $K^{-1}_P$ such that corresponding $K_P$ is valid for $ID_P$.

  we prove subalgebra X with $Sign_{K^{-1}_C}(ID_C :: ID_P :: M_{NT} :: NT) \notin X$ exists.
- Let X be Exp subalgebra generated by
  - G $:= Keys \setminus \{K^{-1}_C\} \cup Data \cup$

    $\{\{Sign_{K^{-1}_C}(id_C :: id_P :: m :: nt)\}_{sk},$

    $Sign_{sk}(m :: \{Sign_{K^{-1}_C}(id_C :: id_P :: m :: nt)\}_{sk}) :$

    $(id_C, id_P, m, nt) \neq (ID_C, ID_P, M_{NT}, NT)\}.$

# Security of Improved Protocol Proof: Cardholder Security

- By construction, X fulfills above conditions,
  - Adversary does not have access to $Sign_{K^{-1}_{CA}}$ ,
    - not adverary's initial knowledge and
    - it (or anything it could be derived from) is never transmitted.

  thus doesn't have access to terms of form $Sign_{K^{-1}_{CA}}(id_P :: k_P)$ unless $k_P$ valid for $id_P$.
- Also, we have $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT) \notin X$.
- Thus we have $Sign_{K^{-1}_{C}}(ID_C :: ID_P :: M_{NT} :: NT) \notin \mathbf{K}$.

# Security of Improved Protocol Proof: Merchant Security

Merchant Security proof:

- Each time $D$ receives $M_{NT}$, $P$ is in possession of
  - $Sign_{K^{-1}_{CA}}(ID_C :: K_C)$,
  - $Sign_{K^{-1}_C}(ID_C :: ID_P :: M_{NT} :: NT)$ for some $ID_C$, $K^{-1}_C$ ,
  - a new value $NT$.
- By specification of $P$,
  - and assumption of secure communication link between $P$ and $D$.

  $D$ receives $M_{NT}$ only after $P$ has checked conditions in its part of protocol:
  - $P$ is in possession of $Sign_{K^{-1}_{CA}}(id_C :: k_C)$ and
  - $Sign_{K^{-1}_C}(id_C :: ID_P :: M_{NT} :: NT)$ for some $id_C$.
- Newness of $NT$ guaranteed
  - $P$ creates value itself. (Incrementing between different runs of protocol),
  - and value is prevented from rolling over.

- Card issuer security: Follows from cardholder security proof.

# Protocol Improvement: Discussion I

Note

- Card C can't verify: Identity $ID_P$ corresponds to PSAM with which it communicates.
- Certificate proves $K_P$ is valid public key, linked to some identity $ID_P$.
- No information in $ID_P$ linking to physical POS device containing PSAM owning $ID_P$.
  - Such as shop name, or location.
  - Information exists only at card issuer
    - Not obtained during transaction.
  - Thus: C „knows“ it owes money to PSAM P with which it communicates.
    - C doesn't know whether P registered as being in physical location where C currently is.
    - and C doesn't know what this physical location is.
    - Including this information would probably improve the security of the protocol.

# Protocol Improvement: Discussion II





  - Attack described could be detected by cardholder immediately after transaction with a portable cardreader.
    - Even if POS device display not within security perimeter.
    - Probably incur higher organizational expenses.
- Validity of $ID_P$ not relevant to cardholder in case of successful purchase.
- If $ID_P$ invalid identity, cardholder will have purchased good
  - May not have to pay, in settlement process no legitimate claimer of money.
- However, validity of $ID_P$ gives cardholder better prospect of claiming back amount (illegitimately charged to C by POS device),
- Therefore certificate for POS not redundant.

# Load Protocol General Overview

- Unlinked, cash-based load transaction (on-line).
- 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.
- Uses symmetric cryptography.

# Load Transaction

- Load transactions in CEPS:
  - Are on-line transactions
  - Using symmetric cryptography for authentication.
- Only consider unlinked load, where cardholder pays cash into,
  - possibly unattended,

  loading machine and receives corresponding credit on card.
- Linked load,
  - where funds transferred e.g. from bank account (so-called funds issuer)

  is viewed as offering fewer possibilities for fraud,
  - because funds moved only within one financial institution[1].

1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Funct. Req. p. 12

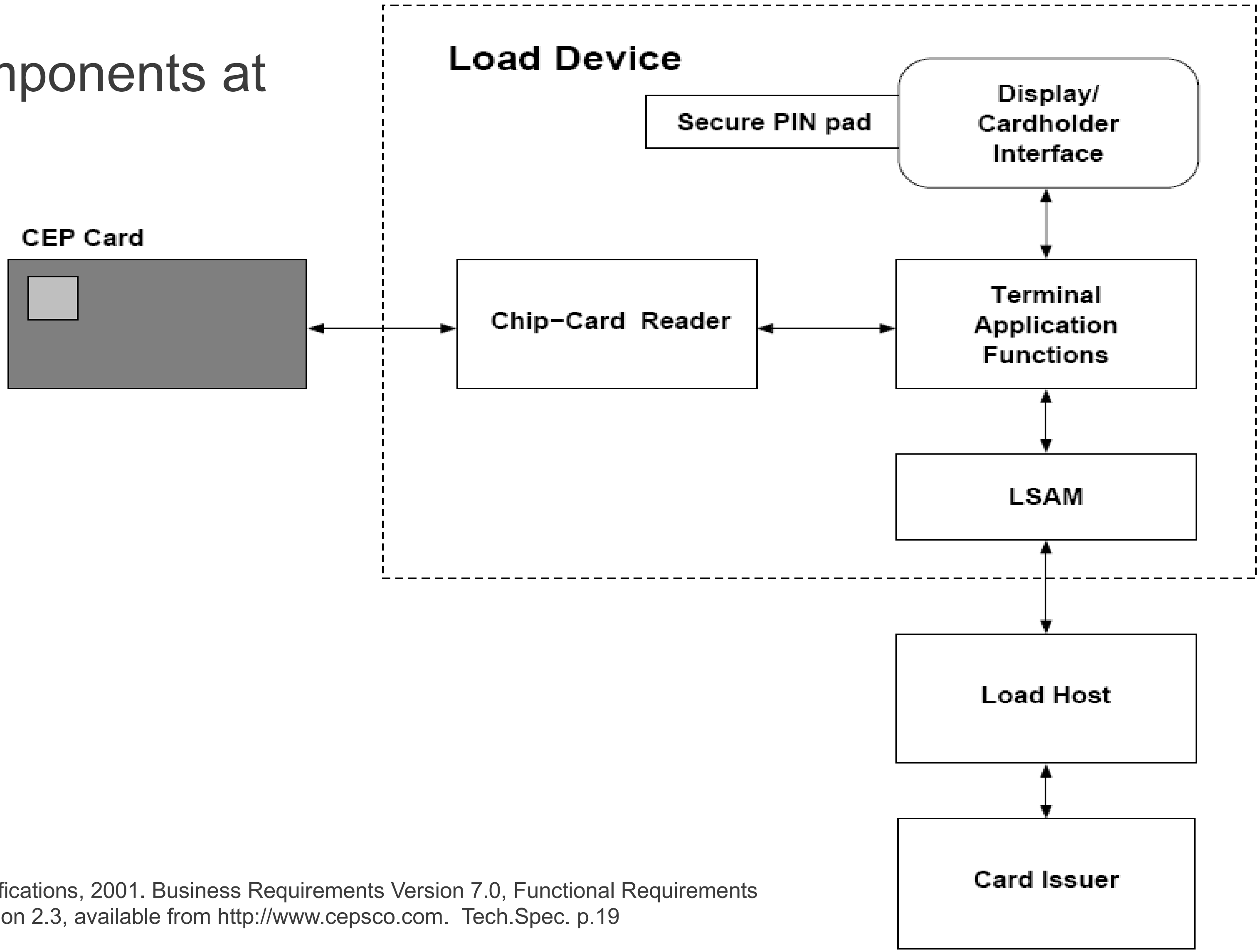
# Load Transaction Informal Description



- To perform cash-based load transaction Cardholder,
  - Inserts card into card reader and
  - Inserts money into cash slot of load device.
  - To load cash on card, enter PIN.
- Remember: Cardholder not able to communicate with card directly,
  - Only through display of load device.
- Load Secure Application Module (LSAM) used to provide necessary cryptographic and control processing.
- LSAM reside within load device or at load acquirer host.
- Load acquirer keeps log of all transactions processed.
- Through load host application, LSAM communicates with card issuer.

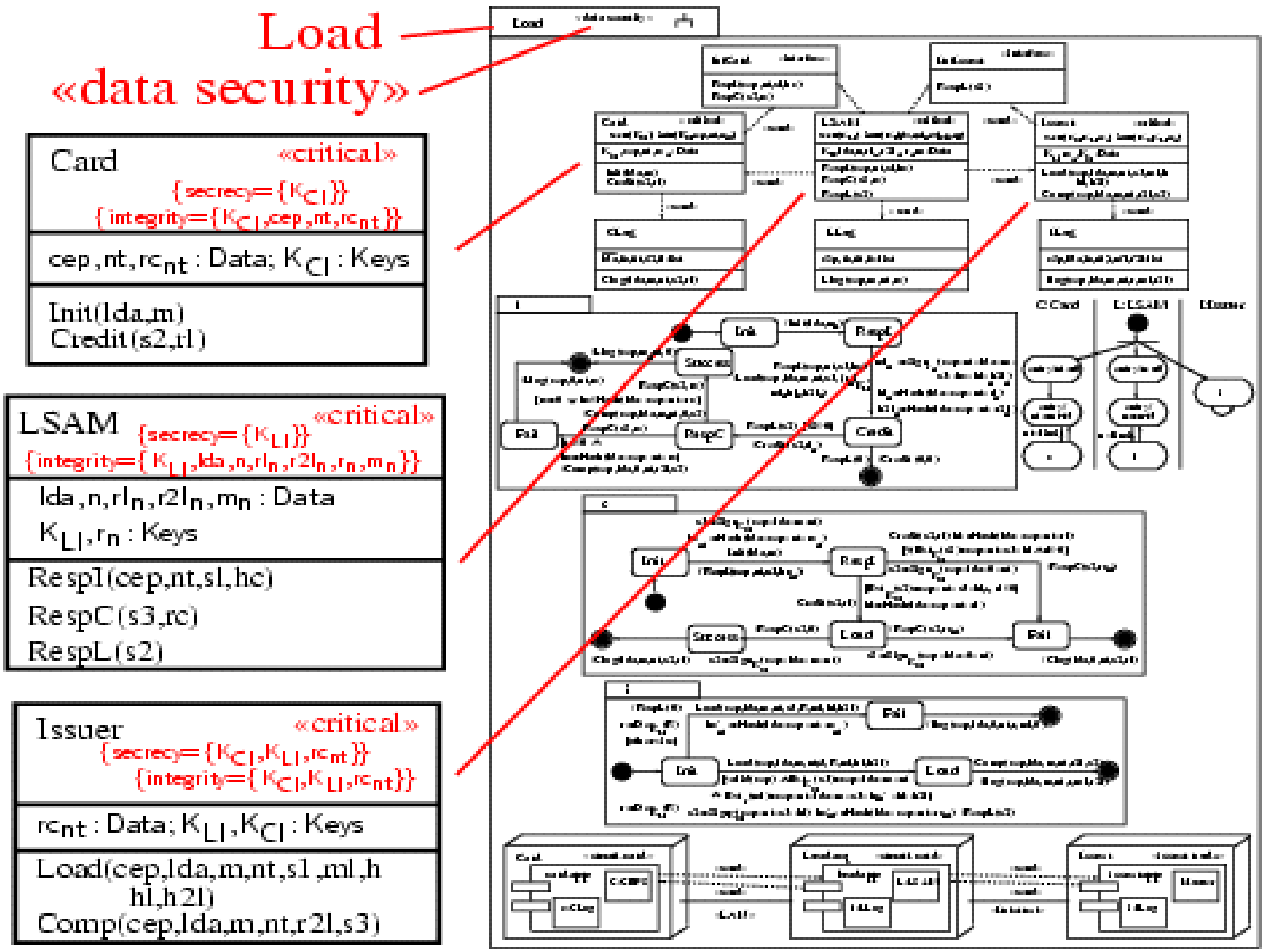
# Load Acquirer Components

Overview over components at load acquirer.[1]

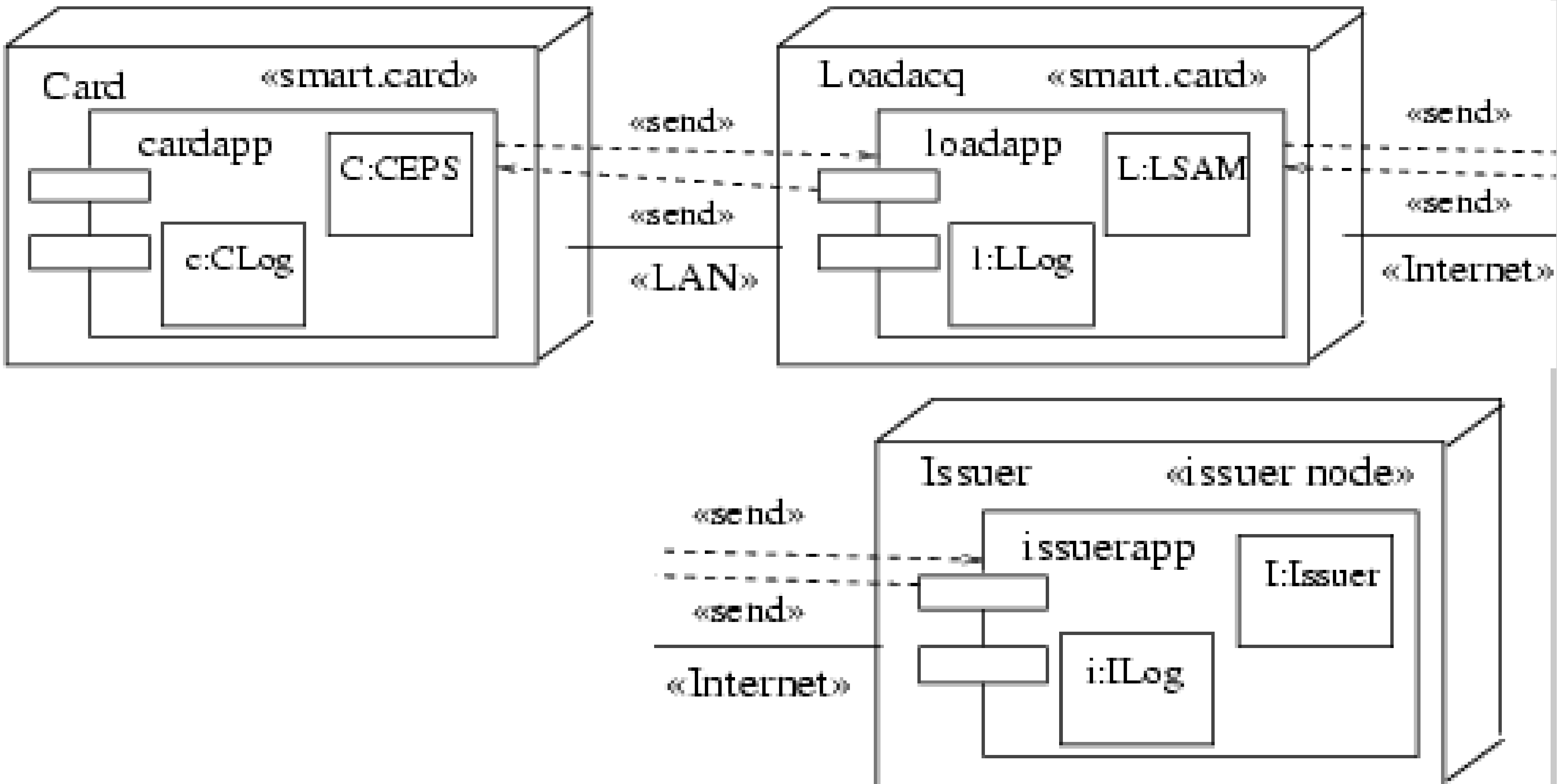



1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Tech.Spec. p.19

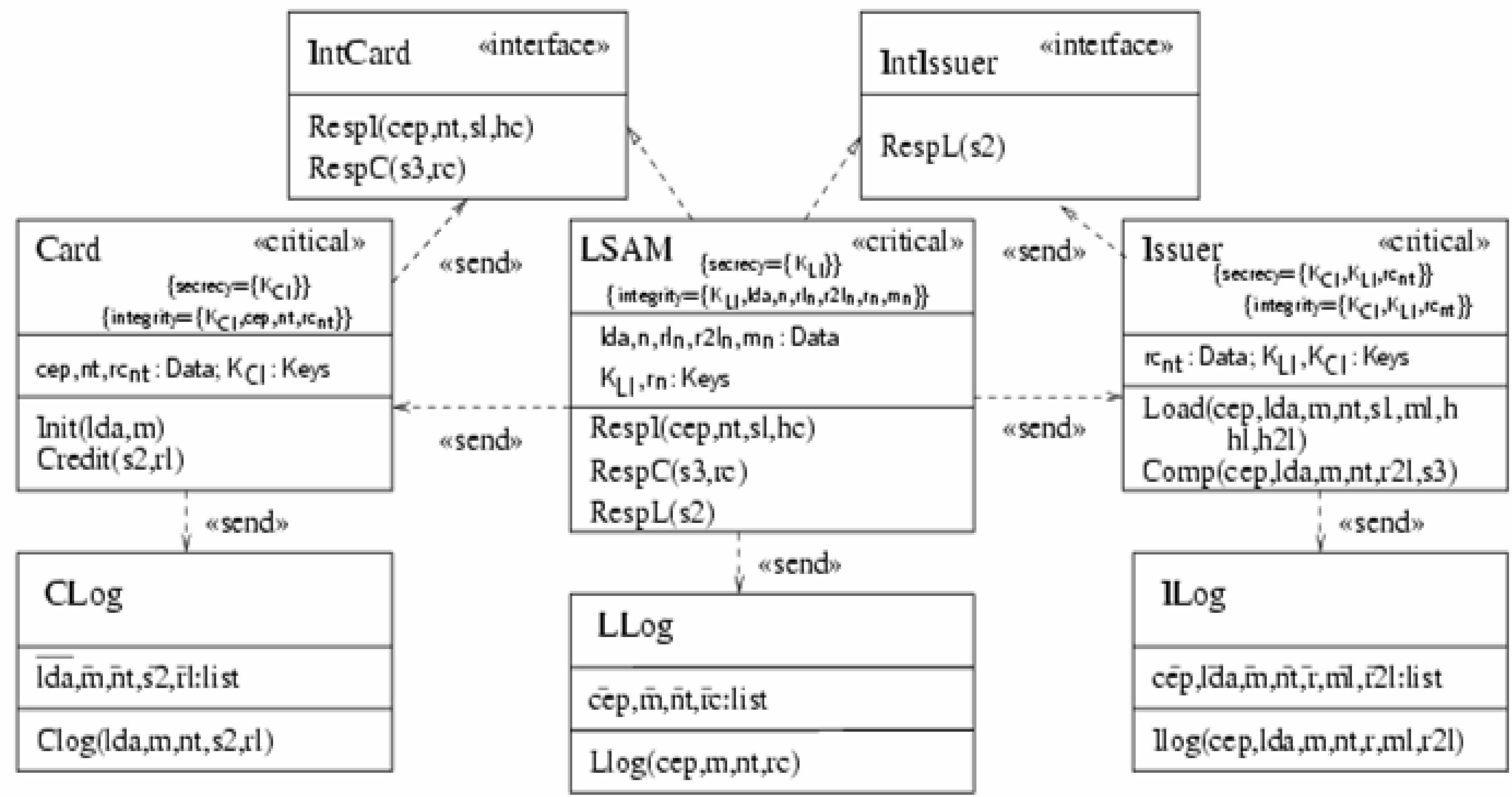
# Load Protocol: Overview



# Load Protocol: Physical View

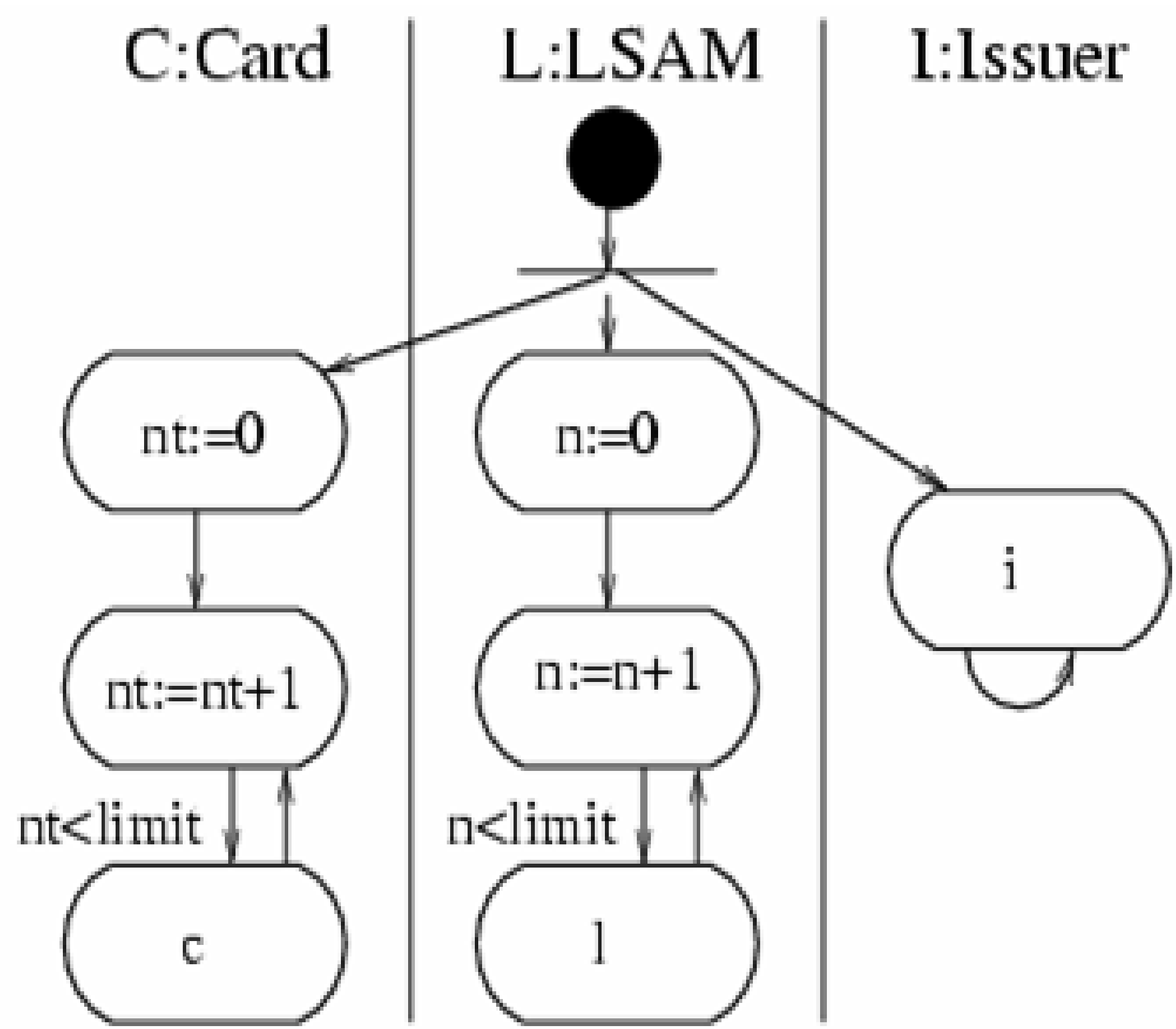


# Load Protocol: Structural View

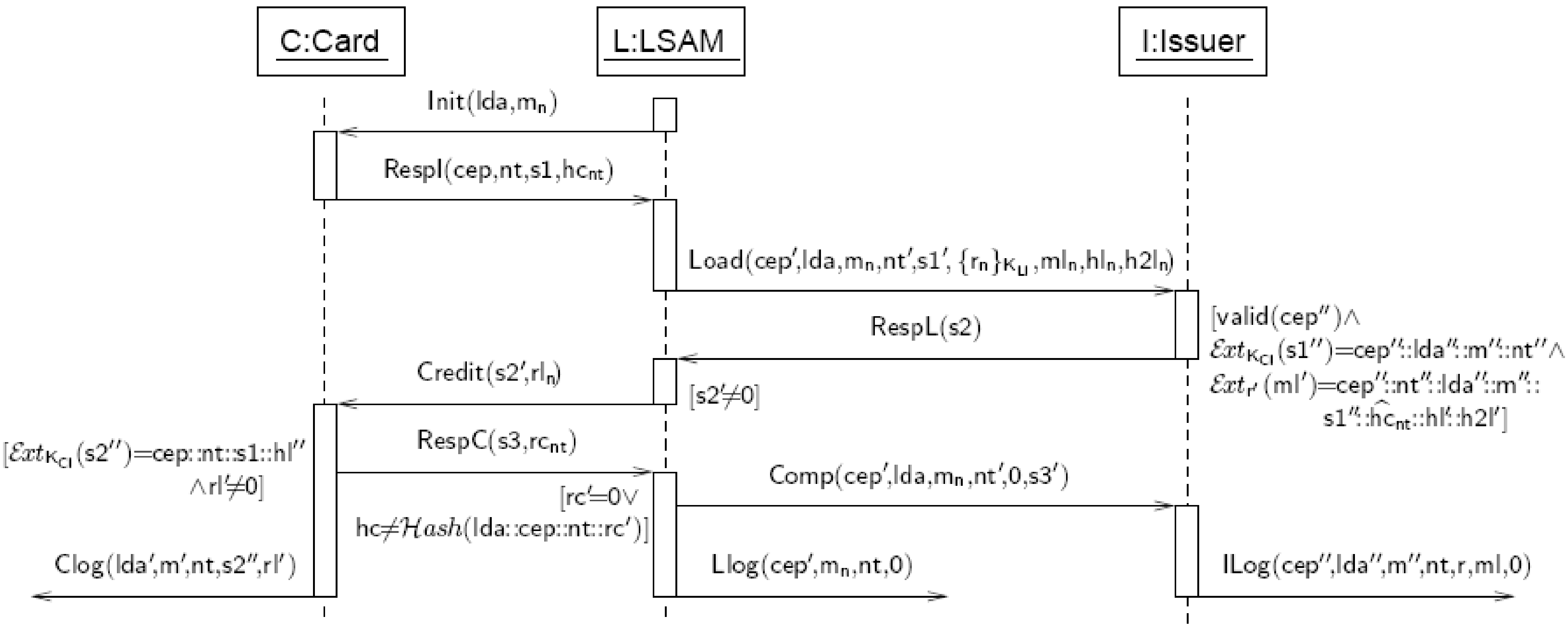


# Load Protocol: Coordination View





# Load Protocol: Interaction View




$(lda', m')$ ::= Init
$s1 ::= \mathcal{S}ign_{K_{CI}}(cep::lda'::m'::nt)$
$hc_{nt} ::= \mathcal{H}ash(lda'::cep::nt::rc_{nt})$
$s3 ::= \mathcal{S}ign_{K_{CI}}(cep::lda::m'::nt)$
$(s2'', rl')$ ::= Credit
$hl'' ::= \mathcal{H}ash(lda'::cep::nt::rl')$

$s2'$ ::= $RespL_1$
$(s3', rc')$ ::= RespC
$(cep', nt', s1', hc')$ ::= RespI
$hl_n ::= \mathcal{H}ash(lda::cep'::nt'::rl)$
$h2l_n ::= \mathcal{H}ash(lda::cep':: nt'::r2l_n)$
$ml_n ::= Sign_{r_n}(cep'::nt'::lda::m_n::s1'::hc'::hl_n::h2l_n)$

$(cep'', lda'', m'', nt'', s1'', R, ml', hl', h2l')$ ::= Load
$r' ::= \mathcal{D}ec_{K_{LI}}(R)$
$s2 ::= \mathcal{S}ign_{K_{CI}}(cep''::nt''::s1''::hl')$
$\widehat{hc}_{nt} ::= \mathcal{H}ash(lda''::cep''::nt''::rc_{nt})$

# Load Transaction
# Used Notations





- Specification of CEPS load transaction:
  - Slightly simplified, leaving out security-irrelevant details.
  - Including exception processing.
  - Specification given in form of UML subsystem L.
- Use notation var ::= exp as syntactic short-cut.
  - Local variable var not used for any other purpose.
  - Expression exp may not contain var.
- Before assigning semantics to diagram, var should be replaced by exp at each occurrence.
- Increase readability, use pattern matching:
  - e.g. (lda',m') ::= Init means
    - when deriving formal semantics of sequence diagram,

    one would have to replace lda' with $Init_1$ and m' with $Init_2$ in each case.

# Load Transaction General Notes



- As with purchase protocol,
  - Link between LSAM and loading device, and loading device itself, need to be secured.
  - Otherwise attacker could initiate protocol without having inserted cash into machine.
- For simplicity, leave out communication between LSAM and loading device to determine amount to be loaded,
- But assume amount is communicated to LSAM in secure way.
- CEP card name cep called valid if:
  - Name registered at card issuer.
  - Name not on list of revoked cards.

# Load Transaction Protocol Participants

- Participants of protocol,
  - Classes Card,
  - LSAM, and
  - Issuer.
  - Each has associated class used for logging transaction data named CLog, LLog, and ILog, respectivly.
- Logging objects:
  - Simply take arguments of their operations and update attributes accordingly.
  - Behavior for readability omitted in figures.

# Load Protocol Assumption I





- Assume sequence of random values $rc_{nt}$ given
  - Shared between card C and its card issuer I.
  - Values required to be fresh within Load subsystem.
    - Indicated by {fresh}, attached to Load[1].
- Viewing Load subsystem in isolation, associated condition is vacuous:
  - Requires any appearance of expression $rc_x$ in Load must be in Load.
  - Using {fresh} at a top-level subsystem still meaningful.
    - Because, including subsystem in another subsystem, stereotyped <<data security>>, would extend scope of freshness constraint to larger subsystem.

1 Jan Jürjens, Secure Systems Development with UML, Springer 2004. Sect. 4.1.2

# Load Protocol Assumption II

- In this example: wouldn't make sense to attach {fresh} with value rc_ to any object in Load.
  - Because random values supposed to be shared among C and I.
- Write rc_: Data to denote array with fields in Data.
- Given:
  - Random numbers $rl_n$, $r2l_n$ and symmetric keys $r_n$ of LSAM.
- Values supposed to be generated freshly by LSAM.
- Expressions of form $rl_x$, $r2l_x$, $r_x$,
  - for any subexpression x,

  only appear in object and statechart associated with LSAM.

# Load Protocol Assumption III



- Remember:
  - Keys and random values are independent of each other and of other expressions in diagram.
  - Constant attributes have initial values as attribute names and corresponding attribute types underlined.
- Finally: Given transaction amounts $m_n$.
- Before first protocol run:
  - Card and LSAM initialize card transaction number nt and acquirer-generated identification number n, repectively.
- Before each protocol run.
  - Card and LSAM increment nt and acquirer-generated n,
    - as long as given limit not reached (avoid rolling over numbers).

# Load Protocol Variables

| Variable | Explanation |
|---|---|
| C | card |
| L | LSAM |
| I | card issuer |
| $rc_{nt}$ | secret random values shared between card and issuer |
| $rl_n$, $r2l_n$ | random numbers of LSAM |
| $r_n$ | symmetric keys of LSAM |
| $m_n$ | transaction amounts |
| m, rl, hl | $m_n$, $rl_n$, $hl_n$ as received at card issuer |
| nt | card transaction number |
| n | acquirer-generated identification number |
| lda | load device identifier |
| cep | card identifier |
| s1 | card signature: $\mathcal{S}ign_{K_{CI}}(cep :: lda :: m :: nt)$ |
| $hc_{nt}$ | card hash value: $\mathcal{H}ash(lda :: cep :: nt :: rc_{nt})$ |
| $\widehat{hc}_{nt}$ | $hc_{nt}$ as created at issuer |
| rc, hc | $rc_{nt}$, $hc_{nt}$ as received at load acquirer |
| $K_{CI}$ | key shared between card and issuer |
| $K_{LI}$ | key shared between LSAM and issuer |
| $ml_n$ | $\mathcal{S}ign_{r_n}(cep :: nt :: lda :: m_n :: s1 :: hc :: hl_n :: h2l_n)$ (signed by LSAM) |
| $hl_n$ | hash of transaction data: $\mathcal{H}ash(lda :: cep :: nt :: rl)$ |
| $h2l_n$ | hash of transaction data: $\mathcal{H}ash(lda :: cep :: nt :: r2l)$ |
| s2 | issuer signature: $\mathcal{S}ign_{K_{CI}}(cep :: nt :: s1 :: hl)$ |
| s3 | card signature of the form $\mathcal{S}ign_{K_{CI}}(cep :: lda :: m :: nt)$ |

# Textual Explanation of Interaction initialization I



- Protocol between card C, LSAM L, and card issuer I supposed to start after:
  - C issued by I inserted into loading device containing L and cardholder inserts amount $m_n$ of cash into loading device.

- L initiates transaction after CEP card inserted into load device.
  - By sending "Initialize for load" message Init with arguments.
    - Load device identifier lda and
    - Transaction amount $m_n$.
      - Cash paid into load device by cardholder supposed to be loaded onto C.

# Textual Explanation of Interaction initialization II



- Whenever C receives Init after being inserted into load device, it sends back „Initialize for load response“ message RespI to L, arguments:
  - Card identifier cep,
  - Card's transaction number nt,
  - Card signature s1, and
  - Hash value $hc_{nt}$.
- s1 consists of values cep, received load acquirer identifier lda' and amount m', and nt, all of which are signed with $K_{CI}$ shared between C and corresponding I.
- $hc_{nt}$ is hash of values lda, cep, nt, and $rc_{nt}$.
- $rc_{nt}$ secret shared between C and I

# Textual Explanation of Interaction LSAM Sends „load request“ Message

- L sends "load request" message Load to I, arguments:
  - Received card identifier cep', lda, $m_n$, received transaction number nt' and card signature s1', and values $Enc(K_{LI}, r_n)$, $ml_n$, $hl_n$, and $h2l_n$.
    - $Enc(K_{LI}, r_n)$: encryption of key $r_n$ under key $K_{LI}$ shared between L and I.
- $ml_n = Sign_{r_n}(cep' :: nt' :: lda :: m_n :: s1' :: hc' :: hl_n :: h2l_n)$:
  - Signature of cep', nt', lda, $m_n$, s1', hc', $hl_n$, and $h2l_n$ using $r_n$,
  - hc': message part $hc_{nt}$ as received by L.
  - $hl_n$: hash of lda, cep', nt', and $rl_n$,
  - $h2l_n$: hash of lda, cep', nt', and $r2l_n$.

# Textual Explanation of Interaction Issuer Validates Message I

- I checks if received card identifier cep" is valid and verifies if
  - Received signature s1" is valid signature generated from values:
    - cep",
    - received load device identifier lda",
    - received amount m", and
    - received transaction number nt" with $K_{CI}$.
  - Technically:
    - Whether $Ext_{K_{CI}}$(s1") = cep" :: lda" :: m" :: nt" holds.

# Textual Explanation of Interaction Issuer Validates Message II

- I retrieves r' from received ciphertext R.
  - Supposed to evaluate to $Enc(K_{LI}, r)$

  using $K_{LI}$ shared between L and the I.
- That is, we have r' ::= $Dec_{K_{LI}}(R)$.
- I then checks if received signature ml' is valid signature.
  - of values cep", nt", lda", m", s1", $\hat{hc}_{nt}$, hl, and h2l

  using key r,that is if $Ext_r(ml) = cep :: nt :: lda :: m :: s1 :: \hat{hc}_{nt} :: hl :: h2l$.
- $\hat{hc}_{nt}$: Hash of values lda", cep", nt", and $rc_{nt}$.

# Textual Explanation of Interaction Issuer Reaction



- If checks succeed, I sends „respond to load“ message RespL with argument s2 to L.
  - s2 consists of values cep'', nt'', s1'', and hl', signed with $K_{CI}$.
- Otherwise, I sends RespL with argument 0 to L.
  - Then sends Ilog to logging object.
    - Arguments: cep'', lda'',amount 0 (since load unsuccessful), nt'', r', ml', and 0.
      - (no r2l received from L)
  - And finishes protocol run.

# Textual Explanation of Interaction LSAM Receive RespL





- If L receives s2' ≠ 0 as argument of RespL,
  - Sends „credit for load“ message Credit to C.
    - Arguments: Received signature s2' and value rl
- If L receives zero as argument of RespL,
  - Sends „credit for load“ message Credit.
    - Arguments: 0, 0.

    to C and
  - Finishes protocol by returning cash to cardholder.

# Textual Explanation of Interaction C Receives Message „Credit“ I



- If C receives message Credit, it checks whether
  - First argument s2' is signature of values cep, nt, s1, and hl''.
    - hl'' defined to be hash of Ida', cep, nt, and rl'.
  - Second argument rl' ≠ 0.
  - If either check fails,
    - C sends „response to credit for load“ message RespC with arguments s3 and $rc_{nt}$ to L,
      - s3 consists of cep, Ida', amount 0, and nt, signed with $K_{CI}$.
    - Also sends logging message Clog to object CLog, with arguments Ida', amount 0, nt, s2', and rl'.

# Textual Explanation of Interaction C Receives Message „Credit“ II

- If both checks succeed, C attempts to load itself with amount m'.
  - If C succeeds,
    - Sends message RespC with arguments s3 and 0,
      - s3 defined to be signature of cep, lda', m', and nt using $K_{CI}$.
  - If C fails, sends message RespC with arguments s3 and $rc_{nt}$,
    - s3 defined to be signature of cep, lda', amount 0, and nt using $K_{CI}$.

# Textual Explanation of Interaction LSAM receives message „RespC“

- If L receives message RespC with arguments s3' and rc',
  - Assuming: Not finished already,

  checks whether rc' ≠0 and hc' (in first message from C) is hash of lda, cep', nt', rc'.
  - If yes (load unsuccessful)
    - L sends „transaction completion message“ Comp to I, with arguments
      - cep', lda, amount 0, nt', r2l, and s3'
    - Also, sends logging message Llog to logging object LLog, with arguments
      - cep', amount 0, nt', and rc
    - Then finishes by returning cash to cardholder.
  - If no, L sends message Comp to I, with arguments
    - cep', lda, $m_n$, nt', 0 (no r2l), and s3'.
    - Also, sends Llog with arguments cep', m, nt', and 0 to LLog.
    - Then finishes without returning cash to cardholder.

# Textual Explanation of Interaction Issuer Device Receives message „Comp“

- If issuer device receives message Comp with arguments
  - cep'', lda'', m'', nt'', r2l, and s3''

  from L, (assuming not finished already).
  - Sends message Ilog with arguments
    - cep'', lda'', m'', nt'', r', ml', and r2l

    to object ILog and finishes.
  - In this case, either
    - m'' is supposed to be transaction amount and r2l = 0, or
    - m'' = 0 and r2l ≠ 0.

# Security Threat Model

- Again, assumption card C, LSAM L, and security module of card issuer tamper-resistant w.r.t. adversary under consideration.
  - Contained secret keys can't be retrieved physically.
- For example:
  - Protocol attacked by attacking communication links between protocol participants.
  - Participant
    - Cardholder Ch, load acquirer, or card issuer I

    could exchange respective device with one exhibiting different behavior.

# Security Threat Model

- No direct communication between Ch and C.
  - Security for customer against fraud by load acquirer supposed to be provided by:
    - checking card balance after transaction and
    - complaining to load acquirer, and if necessary to I,

    In case of incorrect processing.

Possible attack motivations:

- **Cardholder**: charge without pay
- **Load acquirer**: keep cardholder's money
- **Card issuer**: demand money from load acquirer

# Security Threat Model
# Load Aquirer Security

- Security for load acquirer against customer
  - Partly relies on fact that signed message from load acquirer acknowledging receipt of payment sent to C
    - Only after cash is inserted into loading device.
- However, since load acquirer obliged to return cash
  - in case of failure in loading process,

  one needs to make sure:
  - Cash returned only in exchange for valid certificate from C.
    - Stating loading process has been aborted.
  - Otherwise Ch could claim not to have received cash-back.

# Security Threat Model
# Load Aquirer Security Details I

- More precisely, value $ml_n$
  - „provides a guarantee that the load acquirer owes the transaction amount to the card issuer“

  for each new $n$, as required[1].
- Guarantee is negated if load acquirer in possession of $rc_{nt}$ .
  - $rc_{nt}$ sent from $C$ to $L$ in case $C$ wants to abort loading protocol after $L$ has released $ml_n$.
- Failed load signaled by $L$ to $I$ by sending $r2l_n$,
  - Can be verified by $I$ by computing hash of $lda :: cep :: nt :: r2l_n$ and comparing it to $h2l_n$ received earlier from $L$.

1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com.  Tech. Spec. 6.6.1.6

# Security Threat Model
# Load Aquirer Security Details II



- Load acquirer can verify $rc_{nt}$ genuine by comparing hash of lda :: cep :: nt :: $rc_{nt}$ with value $hc_{nt}$ (in first message from C).
  - $hc_{nt}$ checked to be genuine by I (receives it in $ml_n$).
- $rl_n$ gives guarantee by L to C that load can be completed and load acquirer will pay transaction amount to I.
- C can verify validity of $rl_n$ by computing hash $hl_n$ of lda ::cep:: nt :: rln and verifying that signature s2 forwarded by L from I was constructed from cep :: nt :: s1 :: hln.
- Signatures s1 and s3 from C indicate C's intention to load contained amount and C's notification to have loaded contained amount.

# Security Threat Model Competing Card Issuer



- May be reasonable that Ch trusts I,
- May not be reasonable to expect load acquirer trusts I.
- Aim of CEPS is to provide globally interoperable system.
- Many Is also operate as load acquirer within their regional boundaries,
  - Means if Ch load cards elsewhere, load acquirers operated by competing Is.
  - Competing Is may not trust each other;
    - Especially when jointly operating relatively complex system that may provide temptation for fraud even at corporate level.

# Security Threat Model Real Life Example



- Realistic threat scenario. E.g.:
  - Urban train operators in major English metropolis [1]
    - Attempted to cheat each other about passenger numbers on respective parts of urban train system.
    - To increase own revenue at expense of their competitors.
- CEPS plainly contend „electronic purse system participants must be assured that load/unload devices must not link to the system without security that protects all participants from fraud“[2] .
- However, Ch and load acquirer may not trust each other, and I may not trust either Ch or load acquirer.
- In particular, I needs to have valid proof in case Ch or load acquirer disputes transaction in post-transaction settlement process.
- Thus security of system relies crucially on validity of audit data.

1. R. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, New York, 2001.
2. CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Bus. req. p. 19

# Security Threat Model Security Conditions



Derive following security conditions:

- Cardholder security:
  - If card appears to have been loaded with certain amount according to its logs,
    - Cardholder can prove to card issuer:
      - There is load acquirer who owes amount to card issuer.
- Load acquirer security:
  - Load acquirer has to pay amount to card issuer only if load acquirer has received amount in cash from cardholder.
- Card issuer security:
  - Sum of balances of cardholder and load acquirer remains unchanged by transaction.

# Security Threat Model Notes

- Protocol doesn't ensure card will be loaded if cardholder inserts cash into loading device,
  - Usual risk, machine simply retains money without further action or
  - Loads card with a smaller amount than inserted.
- Cardholder can only make complaint,
  - If necessary through card issuer in post-transaction settlement scheme.
- Correct functioning of settlement scheme relies on fact that cardholder should only be led to believe that certain amount has been correctly loaded
  - e.g. when checking card with portable cardreader

  if cardholder able to prove this using the card.
- Otherwise load acquirer could first credit the card with correct amount, but later in settlement process claim that cardholder tried to fake transaction.

# Result
# Load Acquirer Security Formalization



- According to CEPS, value $ml_n$, together with the value $rl_n$ sent in CreditforLoad message to card,
  - Taken as guarantee that amount m specified in $ml_n$ has to be paid by specified load acquirer to issuer of specified card,
    - Unless it is negated with value $rc_{nt}$ [1].
- Load acquirer security:
  - Suppose that card issuer I possesses value $ml_n = Sign_{r_n}(cep :: nt :: lda :: mn :: s1 :: hcnt :: hln :: h2l_n)$ and C possesses $rl_n$, where $hn = Hash(lda :: cep :: nt :: rl_n)$.

1 CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from http://www.cepsco.com. Tech. Spec. 6.6.1.6

# Result: Load Acquirer Security Formalization



- I possesses $ml_n$ = $Sign_{r_n}(cep :: nt :: lda :: m_n :: s1 :: hcnt :: hln :: h2l_n)$
- C possesses $rl_n$
- After execution either of following conditions hold:
  - Message $Llog(cep, lda, m_n, nt)$ has been sent to I : Llog
    - Implies that L has received and retains $m_n$ in cash
  - Or message $Llog(cep, lda, 0, nt)$ has been sent to I : Llog
    - load acquirer assumes that load failed and returns amount $m_n$ to cardholder and
    - load acquirer L has received $rc_{nt}$ with
      - $hc_{nt} = Hash(lda :: cep :: nt :: rc_{nt})$ (thus negating $ml_n$).

# Flaw

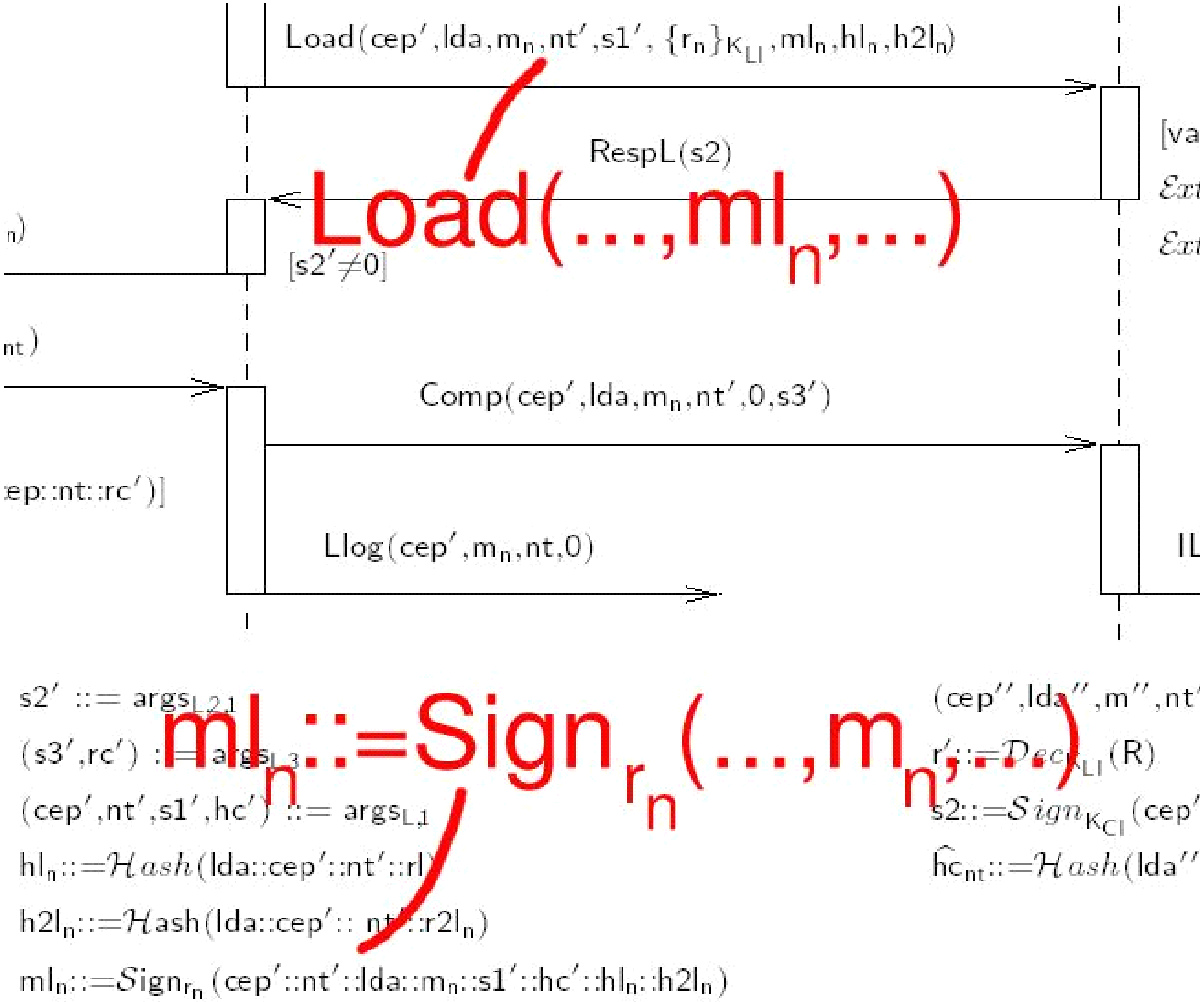
*L* does not provide load acquirer security against adversaries of type insider.

**Why ?**

# Flaw

$ml_n$: „Proof“ for bank that load machine received money.

But: $r_n$ shared between bank and load machine.



# Load Acquirer Security Vulnerability First Weakness Intuitively I

- Weaknesses break both conditions.
- Firstly, $ml_n$ only protected with key $r_n$
  - Which is only protected with key $K_{LI}$
    - Shared between load acquirer and card issuer $I$.
- Further, hash value $hl_n$ doesn't depend on amount $m$.
  - Thus card issuer can modify amount $m_n$ (in $ml_n$) to greater amount $\tilde{m}$.

- In detail:
  - Having received $\{r_n\}_{K_{LI}}$ from load acquirer, $I$ can replace

    $$ml_n = Sign_{r_n}(cep :: nt :: lda :: m_n :: s1 :: hc_{nt} :: hl_n :: h2l_n)$$

    received from load acquirer by value

    $$\tilde{m}l = Sign_{r_n}(cep :: nt :: lda :: \tilde{m} :: s1 :: hc_{nt} :: hl_n :: h2l_n).$$
  - Consequently, load acquirer only receives $m_n$ in cash, but has to pay $\tilde{m}$ to card issuer.

# Load Acquirer Security Vulnerability Legislative Situation



- Assume card issuer in judicially stronger position.
  - e.g. load acquirer may have signed contract to pay whichever amount $m$ contained in $ml_n$.
- In different judicial situation.
  - Load acquirer might instead betray card issuer.
    - By claiming card issuer modified $ml_n$ to contain greater amount $m$, and
    - Pay only allegedly correct smaller amount $m'$.
- Example of observation:
  - Security analysis of practical systems has to take into account legislative situation[1].

1. R. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, New York, 2001.

# Load Acquirer Security Vulnerability Second Weakness Intuitively I

- Vulnerability against load acquirer when:
  - Card sends $rc_{nt}$ to load acquirer in RespC message.
- Only way load acquirer can verify validity of this value is against hash $hc_{nt}$ sent from card to load acquirer in RespI message.
- Since neither:
  - Secret $rc_{nt}$ shared between card and issuer nor Hash $hc_{nt}$

  protected by any signature.

# Load Acquirer Security Vulnerability Second Weakness Intuitively II





- Neither $rc_{nt}$ nor $hc_{nt}$ protected by any signature.
  - Load acquirer has no way to prove in post-transaction settlement process that $rc_{nt}$ is genuine, and that thus cash has been returned to cardholder:
    - Card issuer can simply claim,
      - Card didn't send value $rc_{nt}$ to load acquirer.
      - Load acquirer invented $rc_{nt}$ and computed $hc_{nt}$ .
    - Since card issuer controls settlement process, load acquirer would have to pay.
      - Or go to court, with unclear prospects of success.

# Load Acquirer Security Vulnerability Formalized

Theorem. L doesn't provide load acquirer security against adversaries of type insider with $\{cep, lda, m_n\} \subseteq \mathbf{K}^p_A$.

- Vulnerability has been reported[1].
- CEPS security has been informed and acknowledged observation[2].
- Note: Signatures s1 and s3 considered part of guarantee that load acquirer has to pay contained amount,
  - Doesn't remove weakness entirely,
  - Only requires card issuer to also modify issued cards.
  - Load acquirer not able to verify, s1 and s3 created with $K_{CI}$
    - $K_{CI}$ shared between card and issuer

  contain correct amount m.

1. J. Jürjens. Modelling audit security for smart-card payment schemes with UMLsec. In M. Dupuy and P. Paradinas, editors, Trusted Information: The New Decade Challenge, pages 93-108. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2001.
2. R. Hite. Oral communication, May 2001.

# Proposed Solution

Modifications to protocol:

- $ml_n$ should be protected by asymmetric key:
  - $ml_n = Sign_{K^{-1}_L}$ (cep' :: nt' :: lda :: m :: s1' :: hc' :: $hl_n$ :: $h2l_n$)

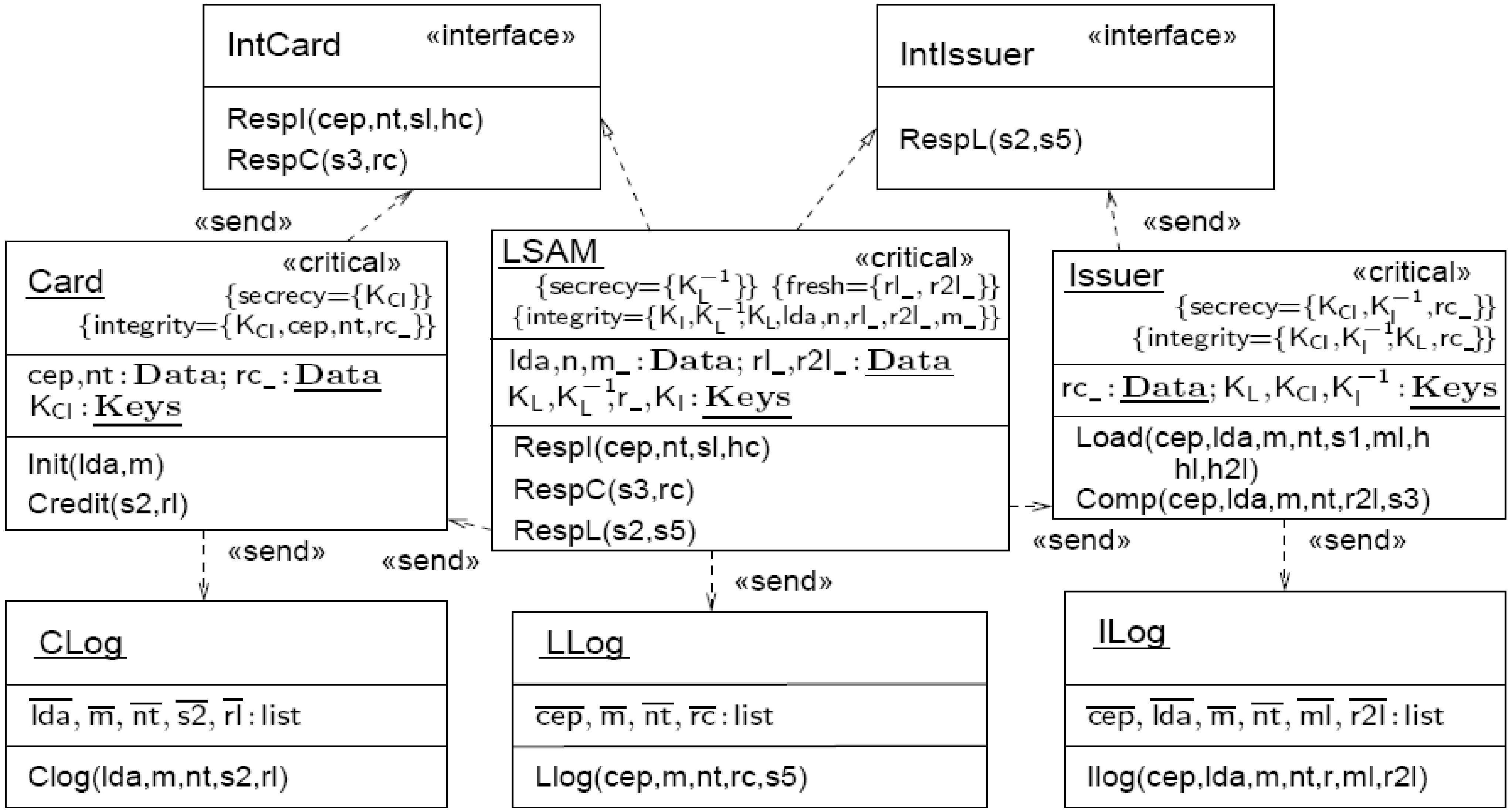
  for private key $K^{-1}_L$ of load acquirer with associated public key $K_L$ and
- In RespL, issuer should also send signature certifying validity of
  - $hc_{nt}$ : RespL(s2, $Sign_{K^{-1}_I}$ ($hc_{nt}$))
  - For private key $K^{-1}_I$ of card issuer with associated public key $K_I$.

# Modifed Specification

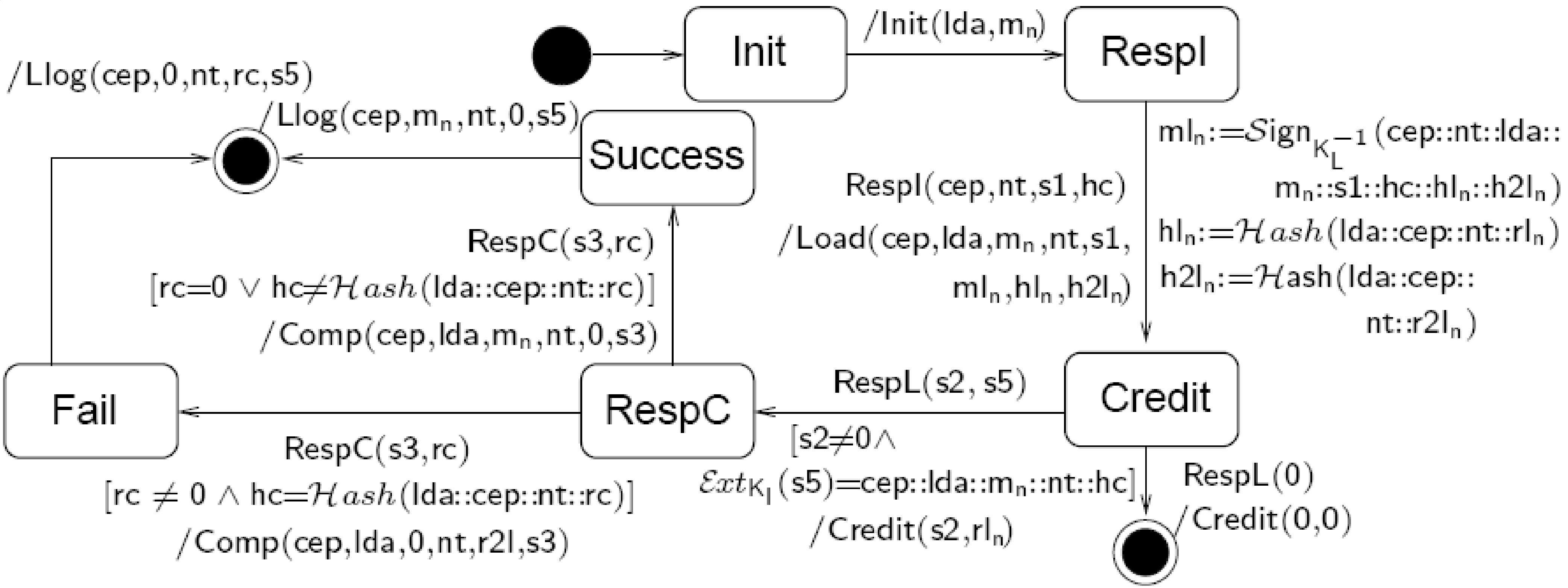
- Modified UML subsystem specification L'.
  - For better readability in modified ULM subsystem, L' split in pieces.
    - Enlarged class and
    - Modified statechart diagrams
    - Given with corresponding exemplary sequence diagram.
  - Assume: Public keys have been exchanged in initialization phase of system
    - Not considered here.

# Repaired load transaction class diagram

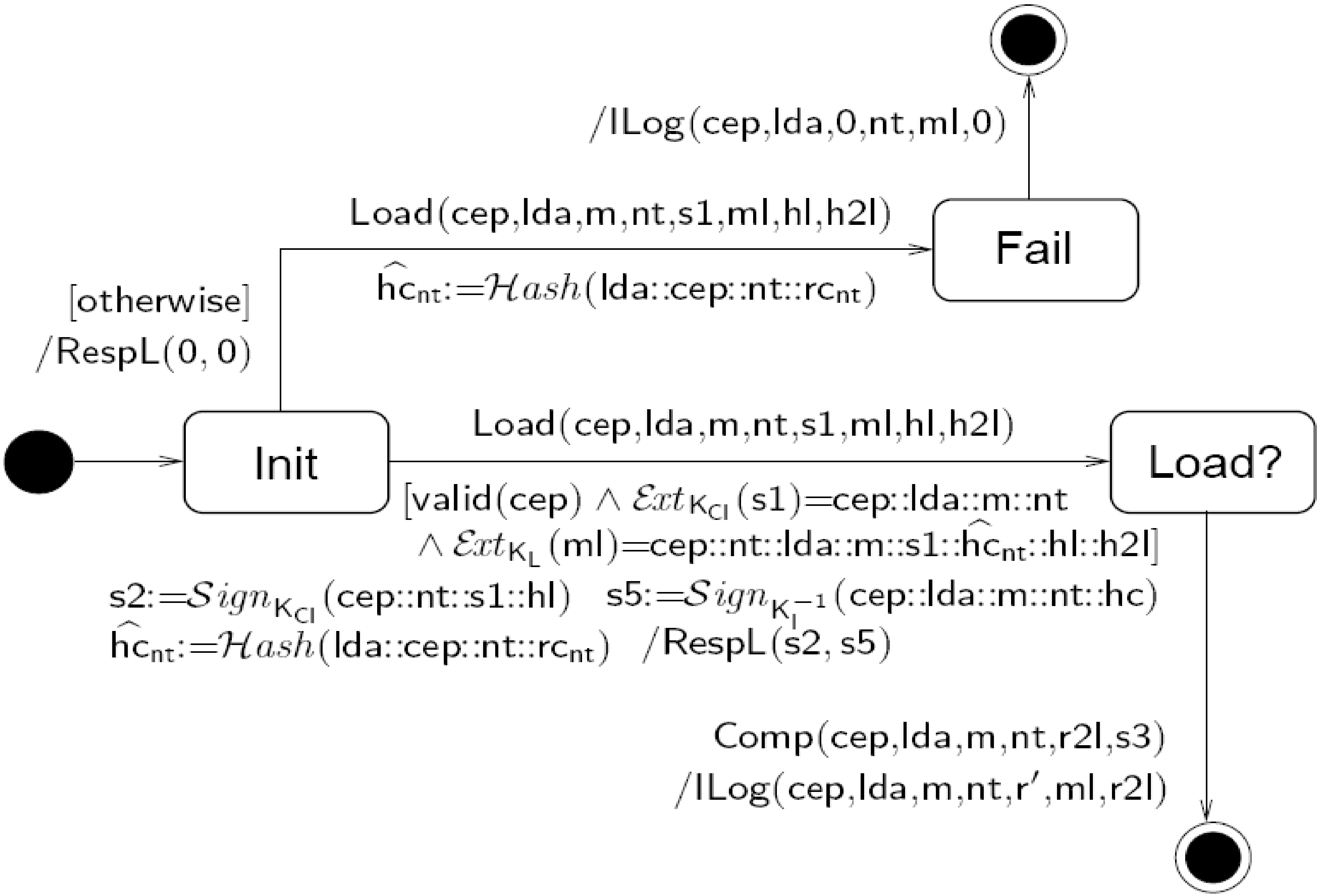




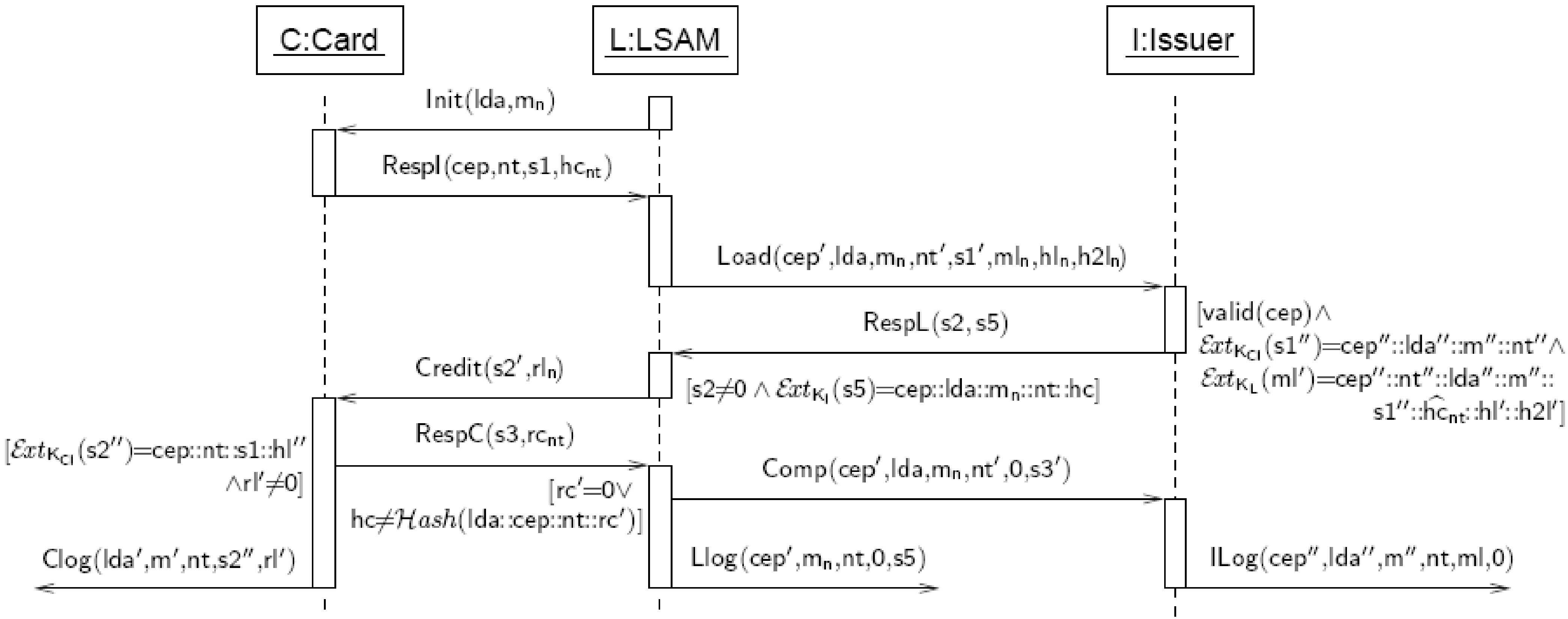
# Repaired Load Transaction: Load Acquirer



# Repaired Load Transaction: Card Issuer



# Sequence Diagram for Repaired Load Transaction




$(lda', m') ::= Init$
$s1 ::= \mathcal{S}ign_{K_{CI}}(cep::lda'::m'::nt)$
$hc_{nt} ::= \mathcal{H}ash(lda'::cep::nt::rc_{nt})$
$s3 ::= \mathcal{S}ign_{K_{CI}}(cep::lda::m'::nt)$
$(s2'', rl') ::= Credit$
$hl'' ::= \mathcal{H}ash(lda'::cep::nt::rl')$

$s2' ::= RespL_1$
$(s3', rc') ::= RespC$
$(cep', nt', s1', hc') ::= RespI$
$hl_n ::= \mathcal{H}ash(lda::cep'::nt'::rl)$
$h2l_n ::= \mathcal{H}ash(lda::cep'::nt'::r2l_n)$
$ml_n ::= \mathcal{S}ign_{K_L^{-1}}(cep'::nt'::lda::m_n::s1'::hc'::hl_n::h2l_n)$

$(cep'', lda'', m'', nt'', s1'', ml', hl', h2l') ::= Load$
$r' ::= \mathcal{D}ec_{K_{LI}}(R)$
$s2 ::= \mathcal{S}ign_{K_{CI}}(cep''::nt''::s1''::hl')$
$\widehat{hc}_{nt} ::= \mathcal{H}ash(lda''::cep''::nt''::rc_{nt})$
$s5 ::= \mathcal{S}ign_{K_I^{-1}}(cep''::lda''::m''::nt''::hc')$

# UMLsec Annotations satisfied ?

- Proposition. L' provides secrecy of $K_{CI}$, $K^{-1}_{L}$, $K^{-1}_{I}$ and integrity of $K_{CI}$, $K^{-1}_{L}$ , $K^{-1}_{I}$ , cep, nt, $rc_{nt}$, lda, n, $rl_n$, $r2l_n$, $m_n$
  - Meaning: Adversary shouldn't be able to make atttributes take on values previously known only to him.

  against insider adversaries with $\mathbf{K}^{p}_{A} \cap \{K_{CI}, K^{-1}_{L}, K^{-1}_{I}\} = \emptyset$.

- Now consider formalizations of above security goals w.r.t. modified specification. They use following two notational definitions.
  - Let **K** be joint knowledge set of all participants except L: any object in classes Card or Issuer, any adversary (not able to penetrate smart card on which L resides, according to threat scenario), and any object in LSAM except L.
  - Let $\mathbf{K}_L$ be knowledge set of L.

1 Jan Jürjens, Secure Systems Development with UML, Springer 2004. Sect. 4.1.2

# Proposition: Proof

Proof:

- Secrecy evident since these values never sent outside smart cards.
  - Current threat scenario, smart cards assumed to be impenetrable.
- Similarly, integrity of
  - $K_{CI}$, $K^{-1}_{L}$ , $K^{-1}_{I}$ , cep, $rc_{nt}$, lda, $rl_{n}$, $r2_{ln}$, $m_{n}$

  evident since not changed during execution of specification.
- Note: secure definition of $m_{n}$
  - Outside current specification.
- Relies on secure connection between terminal (cash entered) and LSAM.
- Creation of random values $rc_{nt}$, $rl_{n}$, $r2_{ln}$ outside current scope.
- Finally, integrity of nt (resp. n)[1] follows from fact, card (resp. LSAM) changes value of nt (resp. n) during protocol irrespective of behavior of environment.

1 Jan Jürjens, Secure Systems Development with UML, Springer 2004. Sect. 4.1.2

# Provided Security Guarantees



Theorem.

- In presence of adversaries of type A = insider with
  - $\mathbf{K}^{p}_{A} \cap \{K_{CI}, K^{-1}_{L}, K^{-1}_{I}\} = \bigcup \{rc_{nt} : nt \in \mathbb{N}\} \cup \{rl_{n}, r2l_{n} : n \in \mathbb{N}\} = \emptyset$

  following security guarantees are provided by L':
  - Cardholder security.
  - Load acquirer security.
  - Card issuer security.

# Cardholder security

Cardholder security:

- For any message Clog(lda, m, nt, s2, rl) sent to c : CLog,
  - if $m \neq 0$ (card seems to have been loaded with m) then $rl \neq 0$ and
    - $Ext_{K_{cl}}(s2) = cep :: nt :: Sign_{K^{-1}_{CI}}(cep :: lda :: m :: nt) :: Hash(lda :: cep :: nt :: rl)$

  holds (card issuer certifies rl to be valid proof for transaction).

  For any two messages
  - Clog(lda,m,nt, s2, rl) and Clog(lda',m', nt', s2',rl')

  sent to c : CLog, we have $nt \neq nt0$.

# Load acquirer security

Load acquirer security:

- Suppose we have $ml_n \in \mathbf{K}$ and $rl_n \in \mathbf{K}$
  - $ml_n = Sign_{K^{-1}_L}(cep :: nt :: lda :: m_n :: s1 :: y :: hl_n :: h2l_n)$
    - with $hl_n = Hash(lda :: cep :: nt :: rl_n)$ and
    - $h2l_n = Hash(lda :: cep :: nt :: r2l_n)$, for some cep, nt, s1, and y.

  At end of execution of L either of the two conditions hold:
  - Message Llog(cep, lda,mn, nt, x) has been sent to l : LLog
    - Implies L has received and retains $m_n$ in cash; or
  - Message Llog(cep; lda; 0; nt; x) sent to l : LLog, for some x
    - Load acquirer assumes, load failed and returns amount $m_n$ to cardholder.

  and we have $x' \in K_L$ and $z \in K$ with $z = Sign_{K^{-1}_{CI}}(cep :: lda :: mn :: nt :: y0)$

  where $y' = Hash(lda :: cep :: nt :: x') = y$ (load acquirer can prove; load was aborted).

# Card issuer security

Card issuer security:

- For each message Clog(lda, m, nt, s2; rl) sent to c : CLog, if
  - $m \neq 0$ and
  - $Ext_{K_{CI}}(s2) = cep::nt :: Sign_{K_{CI}}(cep::lda::m::nt) :: Hash(lda::cep::nt::rl)$

  holds for some lda, then
  - Card issuer has valid signature $ml_n$ corresponding to transaction.

# Cardholder Security Proof I

Cardholder security:

- Suppose: Message Clog(lda,m, nt, s2, rl) sent to c : Clog, with $m \neq 0$.
- Need to show:
  - $rl \neq 0$ and
  - $Ext_{K_{CI}}(s2) = cep::nt::Sign_{K_{CI}}(cep::lda::m::nt)::Hash(lda::cep::nt::rl)$

  holds.
- By assumption: Connection between C : Card and c : CLog secure
  - Since objects on same smart card.
- Implies C actually sent Clog(lda, m, nt, s2, rl).
- According to specification of C: can only happen if $rl \neq 0$ and if $Ext_{K_{CI}}(s2) = cep :: nt :: s1 :: hl$ holds.
  - $s1 = Sign_{K_{CI}}(cep:: lda :: m:: nt)$ and $hl = Hash(lda ::cep:: nt :: rl)$.

# Cardholder Security Proof II

- Suppose two messages
  - Clog(Ida, m, nt, s2, rl) and Clog(Ida', m', nt', s2', rl').

  have been sent to c : CLog.
- Need to show nt ≠ nt'.
  - By threat scenario we can conclude C sent the two messages to c.
  - Suppose (WLOG) Clog(Ida, m, nt, s2, rl) was sent first.
  - According to statechart specification for C, C reaches final state immediately afterwards.
  - According to overall activity diagram (given in specification),
    - C starts new protocol run only after nt incremented
      - (rolling over not possible).
  - Thus have nt' ≥ nt + 1, in particular nt ≠ nt'.

# Load Acquirer Proof I

Load acquirer security:

- Suppose: We have $ml_n \in \mathbf{K}$ and $rl_n \in \mathbf{K}$
  - $ml_n = Sign_{K^{-1}_L}(cep :: nt :: lda :: m_n :: s1 :: y :: hl_n :: h2l_n)$ with
    - $hl_n = Hash(lda :: cep :: nt :: rl_n)$ and
    - $h2l_n = Hash(lda :: cep :: nt :: r2l_n)$,

    for some cep, nt, s1, and y,

  and message Llog(cep,0,nt,x) has been sent to l : LLog, for some x.
- Need to show $\exists\, x' \in \mathbf{K}_L$ and $z \in \mathbf{K}$ .
  - With $Z = Sign_{K^{-1}_l}(cep :: lda :: m_n :: nt :: y')$
    - Where $y' = Hash(lda :: cep :: nt :: x') = y$.

# Load Acquirer Proof II

- By threat scenario, communication link between L and I is secure (according to specification only L can send messages to I).
- Implies message Llog(cep, 0, nt, x) to I : LLog originated at L.
- According to specifications of L, this implies:
  - L previously received message RespC(s3, x') with
    - x' = x, x' ≠ 0 and such that Hash(lda :: cep :: nt :: x') = y'
      - for y' received in message RespI(cep, nt, s1, y') previously in same protocol run,
  - And such that for second argument of message RespL(s2, z),
    - Received immediately before RespC(s3, x').

    $Ext_{K_I}(z)$ =cep:: lda :: $m_n$ :: nt :: y' holds.
    - (in particular we have x', z ∈ $\mathbf{K}_L$).

# Card Issuer Security Proof I

Card issuer security:

- Suppose message Clog(lda, m, nt, s2, rl) was sent to c : Clog, where
  - $m \neq 0$ and
  - $Ext_{K_{CI}}(s2)=cep::nt::Sign_{K_{CI}}(cep::lda::m::nt)::Hash(lda::cep::nt::rl)$

  holds for some lda.
- Need to show:
  - Issuer has valid signature $ml_n$ corresponding to this transaction.

# Card Issuer Security Proof II

- From specification of C we see:
  - C received message Credit(s2,rl) just before in same protocol run
  - and $Ext_{K_{CI}}(s2) = cep :: nt :: s1 :: hl$ holds, where
    - $s1 := Sign_{K_{CI}}(cep :: lda :: m :: nt)$ and
    - $hl := Hash(lda :: cep :: nt :: rl)$.
- Since $K_{CI}$ kept secret by C and I (as prosposed).
  - Conclude: I created s2.
- According to specification of I, can only be if
  - $ml \in K_I$ with $Ext_{K_L}(ml) = cep :: nt :: lda :: m :: s1 :: \hat{hc}_{nt} :: hl :: h2l$.

# Note on Changed Condition of: Load Acquirer security



- Changed condition of load acquirer security slightly to accommodate changes in protocol.
- To see that it is formalized in adequate way, note that value
  - $ml_n = Sign_{K^{-1}_L}(cep :: nt :: lda :: m_n :: s1 :: hc :: hl_n :: h2l_n)$

  known outside L only after load acquirer has received amount $m_n$ in cash.
  - Follows from facts that
    - Protocol at L started only after cash is inserted,
    - $ml_n$ is signed with key K¡1L , and
    - Key only accessible to L, by previous Proposition.
- Critical question:
  - Cash returned to cardholder after $rl_n$ becomes known outside L?
- According to specification of L may happen only after message of form Llog(cep, 0, nt, rc) sent to l : LLog.

# Summary

- **Example security analysis**
  - Practical use of UMLsec
  - Formal proof
  - Apply fix for vulnerability