

Trust and Agreement in Cryptographic Protocols

Joshua D. Guttman

Jonathan C. Herzog Jonathan K. Millen John D. Ramsdell

Brian T. Sniffen F. Javier Thayer

December 2005

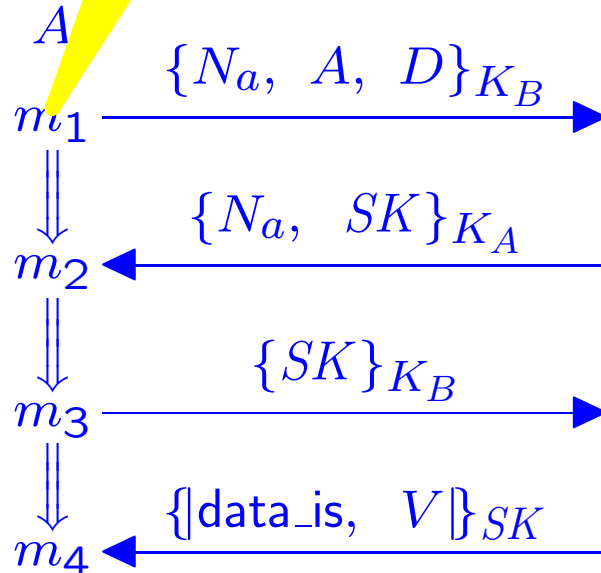
Supported by: **MITRE-Sponsored Research**

Trust Engineering

- How to design new, application-specific protocols
- What good are new crypto protocols?
 - Tune authentication, confidentiality to distributed applications
 - Crafted to transactions in
 - Electronic commerce
 - Web services
 - Remote attestation
 - “Trust engineering:”
 - Craft protocol to match trust goals of participants
- CPPL:
 - A domain-specific Cryptographic Protocol Programming Language

Stock quote web service, I

$\text{requests}(A, B, D, N_a)$



K_A, K_B

N_a

SK

$\{t\}_K, \{\{t\}\}_K$

V

Public asymmetric encryption keys of A, B

Nonce, one-time random bitstring

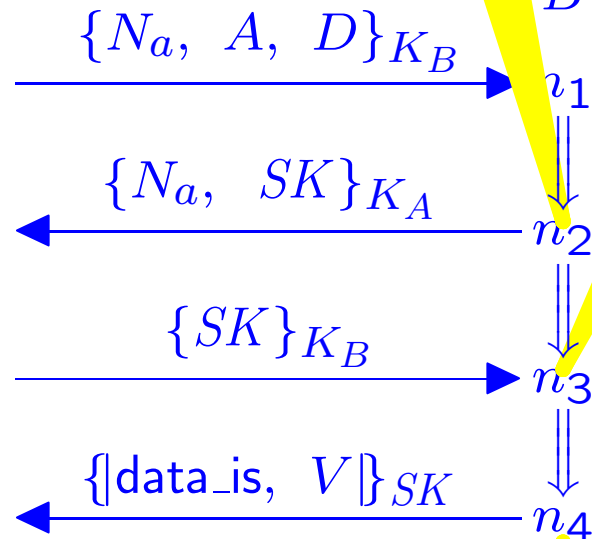
One-time symmetric session key

Asymmetric, symmetric encryption of t with K

Stream of real-time stock quotes

$\text{pubkey}(A, K_A)$

A says
 $\text{requests}(A, B, D, N_a)$



$\text{will_pay}(A, D)$
and $\text{curr_val}(D, V)$

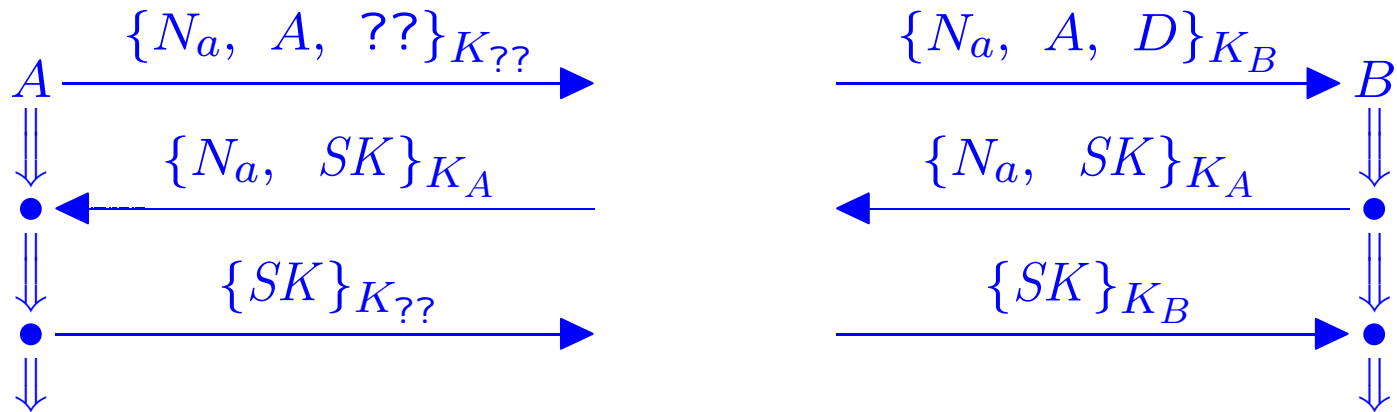
Needham-Schroeder

Trust Engineering

- How to design new, application-specific protocols
- What good are new crypto protocols?
 - Tune authentication, confidentiality to distributed applications
 - Crafted to transactions in
 - Electronic commerce
 - Web services
 - Remote attestation
 - “Trust engineering:” Craft protocol to match trust goals of participants
- Trust goals: Ensure
 - Agreement among participants
 - Safe execution with limited trust
- Semantics of a protocol:
 - Requirements for a run
 - + what is learnt from successful run

How does our protocol work?

Assume A 's private key K_A^{-1} uncompromised

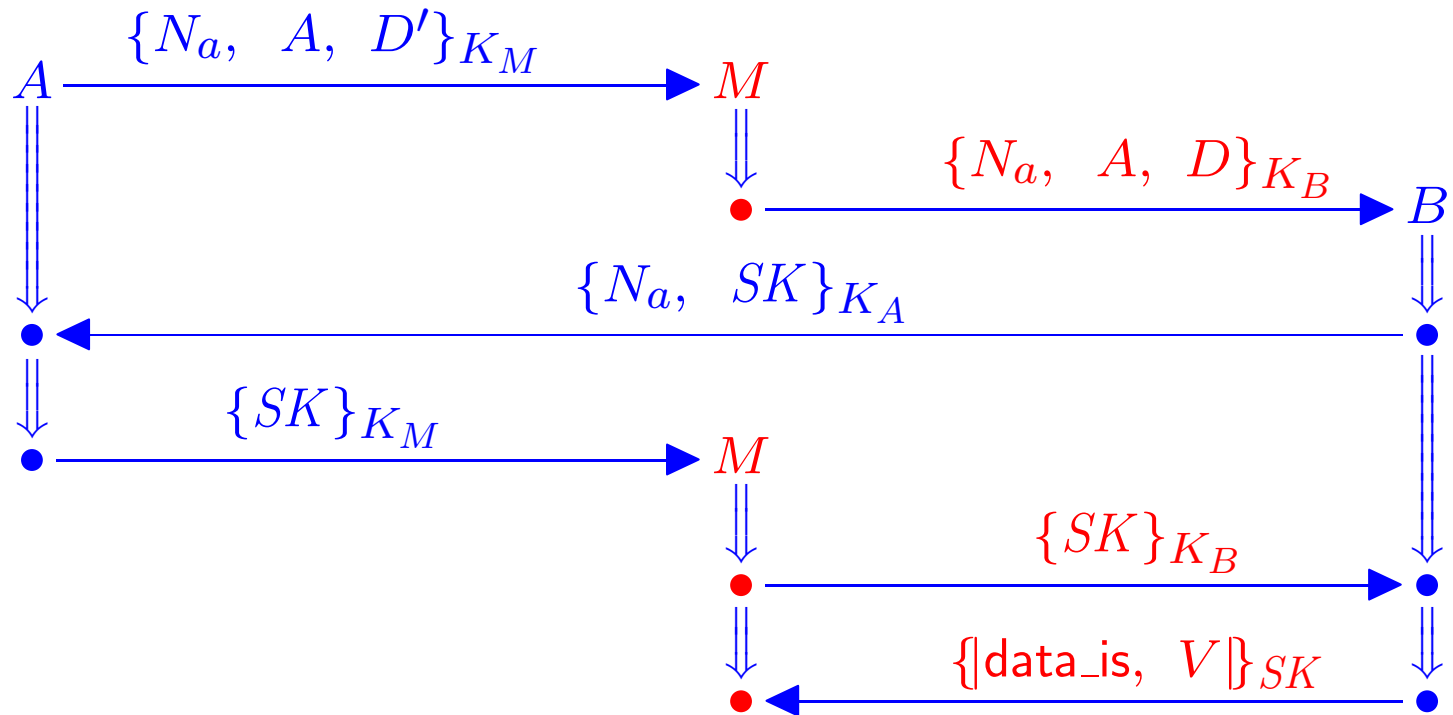


- K_A, K_B Public asymmetric encryption keys of A, B
- N_a Nonce, one-time random bitstring
- SK One-time symmetric session key
- $\{t\}_K$ Encryption of t with K
- V Stream of real-time stock quotes

Whoops

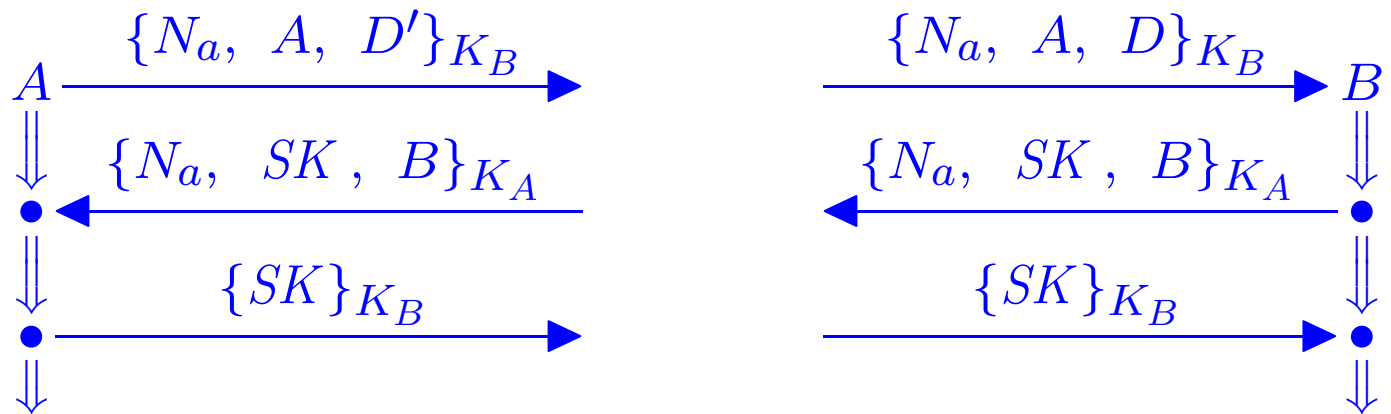
Attack Scenario

If $?? = D', M,$



(Gavin Lowe)

Needham-Schroeder-Lowe stock quote service



In fact $D' = D$,
assuming K_B^{-1} also uncompromised

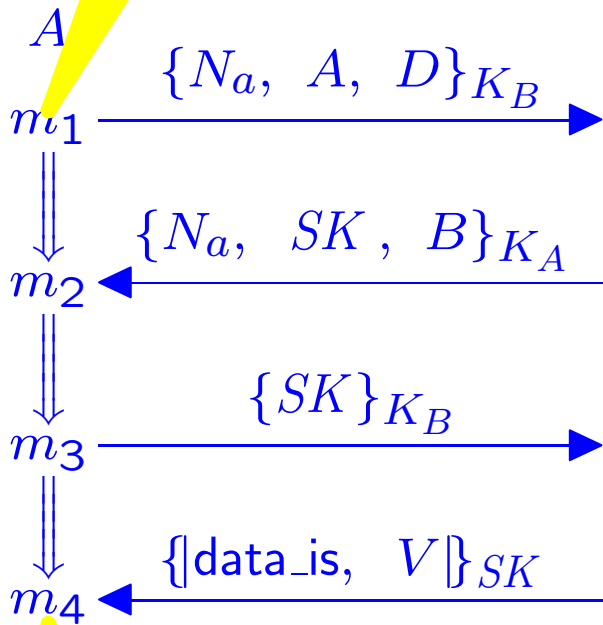
(Details on another occasion)

Authentication Protocols: A Coordination Mechanism

- Causes principals to agree on certain parameters
- After a run, participant knows:
 - There is a protocol run by another principal
 - Some parameters match across runs
 - Some shared values are secrets
 - Other principal's run overlaps mine temporally
- Protocol design now tractable, based on a few theorems
 - “Authentication tests” determine extent of agreement
 - Formalize reasoning of previous slides via strands and bundles
- Clarify real-world consequences of protocol run
 - When is customer committed to paying?
 - When is merchant committed to shipping?
 - Whose word did you depend on when deciding?
- Trust decisions constrain protocol runs

Stock quote service, II

requests(A, B, D, N_a)

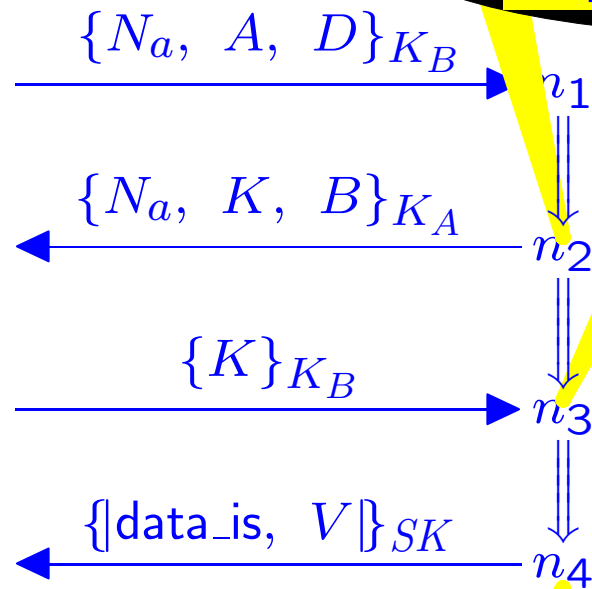


V is a stream of real-time stock quotes
 A 's request for D costs money

B says
 curr_val(D, V, N_a)

pubkey(A, K_A)

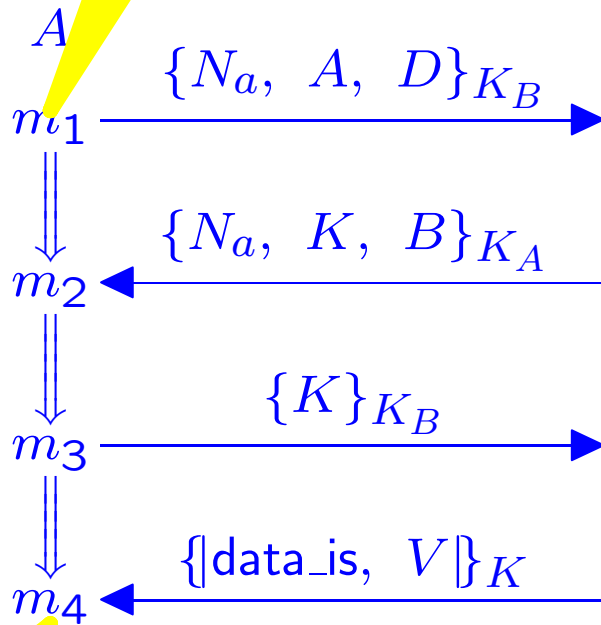
A says
 requests(A, B)



will_pay(A, D)
 and curr_val(D, V, N_a)

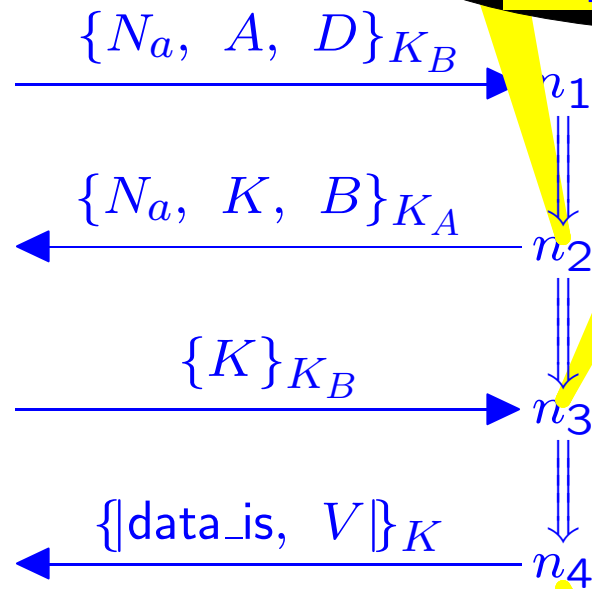
NSL Access

requests(A, B, D, N_a)



B says
curr_val(D, V, N_a)

pubkey(A, K_A)

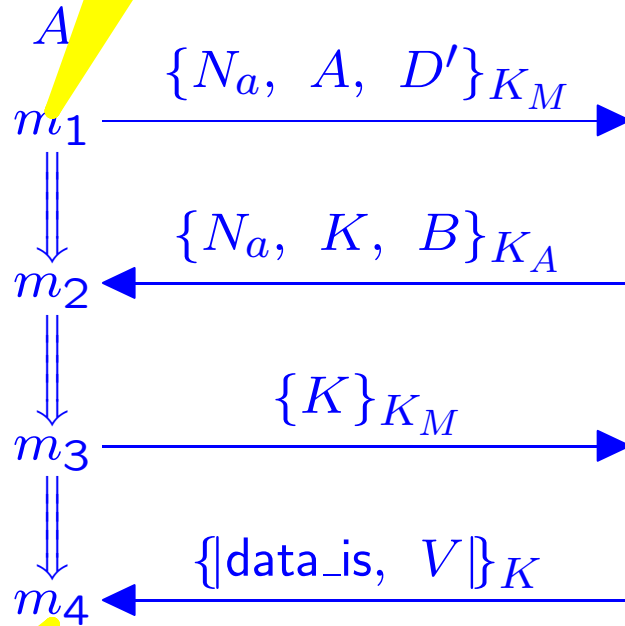


A says
requests(A, B)

will_pay(A, D)
and curr_val(D, V, N_a)

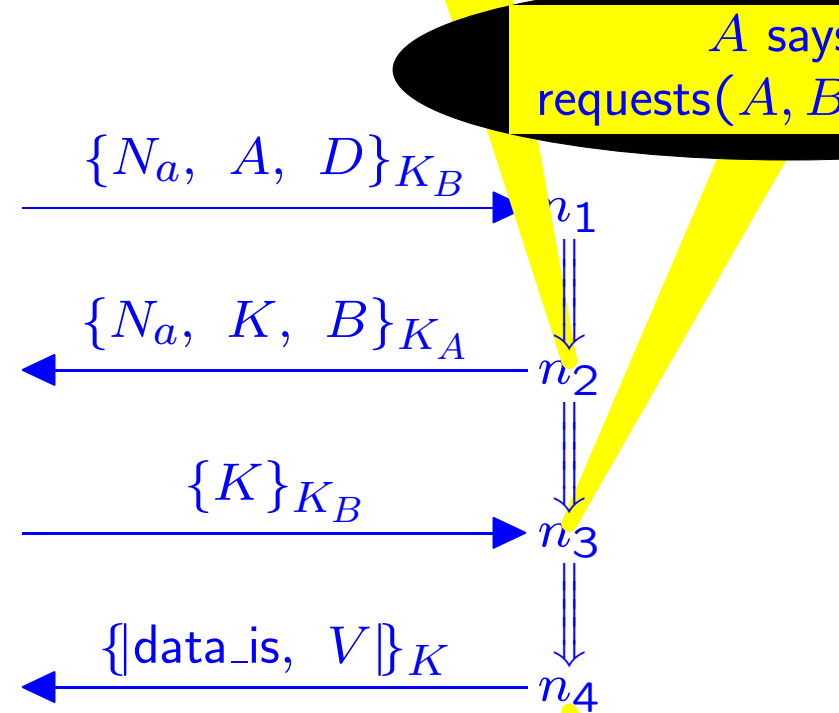
NSL Access Failure

requests(A, P, D', N_a)



M says
curr_val(D', V, N_a)

pubkey(A, K_A)



will_pay(A, D)
and curr_val($D, V,$

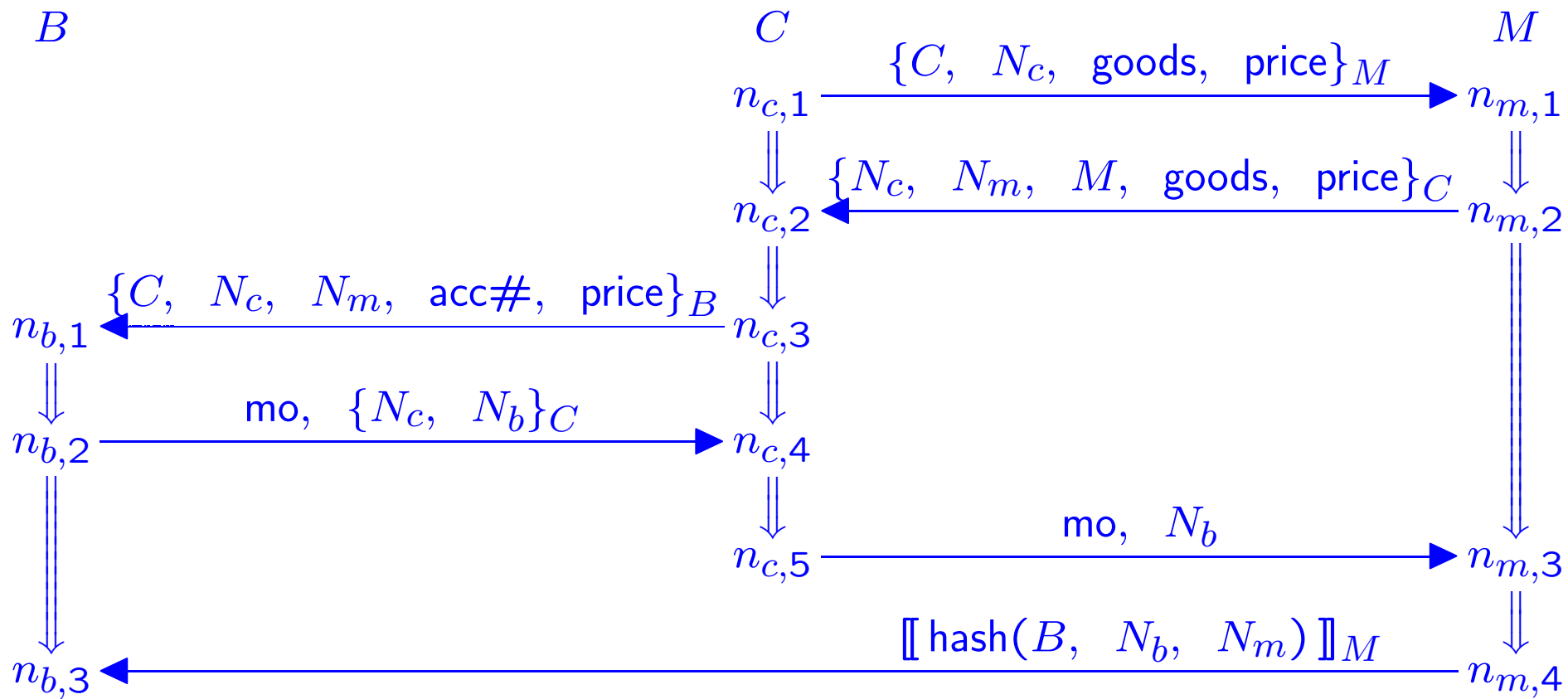
Trust management and protocols

- Each principal P
 - Reasons locally in initial theory Th_P , e.g. a theory in Datalog
 - Derives guarantee before transmitting message
 - Relies on assertions of others as premises
- Premises: formulas associated with message receptions
 - Specifies what recipient may rely on, e.g. “ A says requests(A, B, D, N_a)”
 - Provides local representation of remote guarantee
 - Th_P determines whether ϕ follows from P' says ϕ
- Role of protocol
 - When I rely on you having asserted a formula, then you did guarantee that assertion
 - Coordination mechanism for rely/guarantees
 - **Sound** protocol: “relies” always backed by “guarantees” even with malicious adversary



Relevant
notion of “trust”

EPMO: Limited Agreement



Electronic Purchase using Money Order
 $\text{mo} = \llbracket \text{hash}(C, N_c, N_b, N_m, \text{price}) \rrbracket_B$

A Domain Specific Language

- CPPL, a Cryptographic Protocol Programming Language
 - Expresses cryptographic protocols
 - Programmer treats crypto primitives as black boxes
 - Controls behavior via trust constraints
 - Equipped with a useful semantics
 - Useful for proving protocol security
 - Useful in structuring compiler we wrote



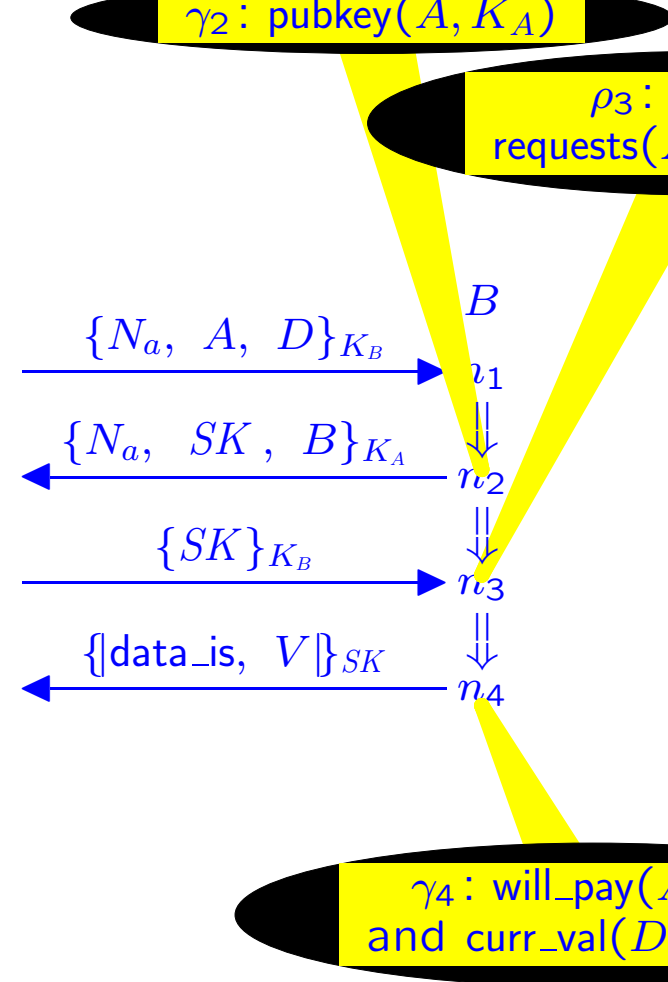
In the
strand space
framework



spi or applied pi
would also work

Coding the Server

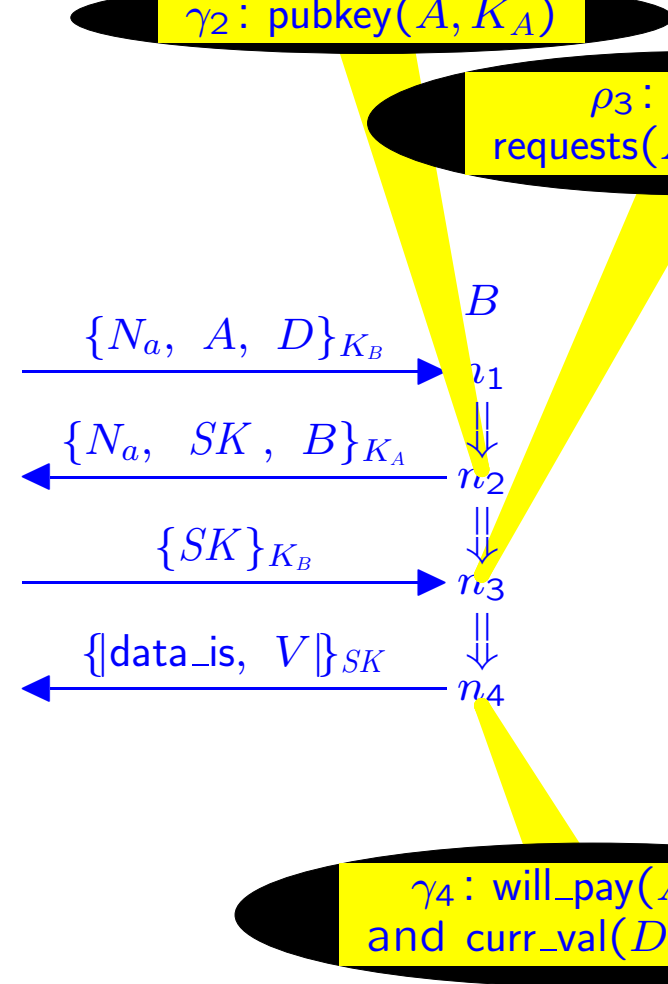
```
let chan = accept in
  receive chan {n_a, a, d} kb
  --> _
  let sk = new symkey in
    send _
    --> chan {n_a, sk, b} ka
    receive chan {sk} kb
    --> _
    send _
    --> chan {|data_is, v|} sk
  return
```



Coding the Server: Trust formulas

```

let chan = accept in
receive chan {n_a, a, d} kb
  --> true
let sk = new symkey in
send pubkey(a,ka)
  --> chan {n_a, sk, b} ka
receive chan {sk} kb
  --> says_requests(a,a,b,d,n_a)
send will_pay(a,d) and curr_val(d,v,n_a)
  --> chan {|data_is, v|} sk
return
  
```



Semantics of receive and send

$$\frac{\sigma_1 = \sigma \oplus \sigma' \quad \sigma_1 ; \Delta, \phi \sigma_1 \vdash c : s}{\sigma ; \Delta \vdash (x \text{ recv } m \phi c) : -(x, m) \sigma_1, \phi \sigma_1 \Rightarrow s}$$

$$\text{dom}(\sigma') \subseteq \text{vars}(m) \quad \text{vars}(x, m, \phi) \subseteq \text{dom}(\sigma_1)$$

\oplus means disjoint union of fns

$-(x, m), \phi \Rightarrow s$:

receive m from channel x , relying on ϕ ; then do rest of strand s

$$\frac{\sigma_1 = \sigma \oplus \sigma' \quad \Delta \Vdash \phi \sigma_1 \quad \sigma_1 ; \Delta \vdash c : s}{\sigma ; \Delta \vdash (\text{send } \phi x m c) : +(x, m) \sigma_1, \phi \sigma_1 \Rightarrow s}$$

$$\text{dom}(\sigma') \subseteq \text{vars}(\phi) \quad \text{vars}(x, m, \phi) \subseteq \text{dom}(\sigma_1)$$

$+(x, m), \phi \Rightarrow s$:

transmit m on channel x , guaranteeing ϕ ; then do rest of strand s

Server with choice

```
receive chan_a {n_a, a, d} kb --> true
let k = new symkey in
  send pubkey(a, ka) --> chan_a {n_a, k, b}ka
  receive chan_a {k}kb --> says_requests(a, a, b, d, n_a)
  send with
    corp_subscriber(a,d) and curr_val(d, v, n_a)
      --> chan_a {data_is, v}k
  return
|
  home_subscriber(a,d) and approx_val(d, v, n_a)
    --> chan_a {approx_data_is, v}k
  return
end
```

CPPL Principles

- Principal maintains an environment during run
 - Variables progressively become bound, never change value after
 - Values are atomic (nonce, name, key, etc)
- Message transmission, reception:
 - Reception:
 - Branch on form of message
 - New variables bound from msg components
 - Rely on assertion of sender
 - Transmission:
 - Branch on successful guarantee
 - New variables bound from successful guarantee (as in logic programming)
- Derive guarantees using:
 - Th_P , your initial theory
 - Values of variables bound up to this point
 - Rely formulas for earlier msg receptions

Subprotocols

- Subprotocols encapsulated by rely/guarantee
 - Callee relies on assertion of caller
 - Property of input parameters
 - Callee guarantees result for caller
 - Relation on input, output parameters
 - Caller and callee are same principal P (same theory Th_P)
- Subprotocol call, return: local message transmissions
 - Call: Message from caller to callee
 - Return: Message from callee to caller
 - RPC-like mechanism (“LPC”)
- Flow of information on subprotocol call, return matches convention
 - Guarantee before transmitting
 - Rely when receiving

Protocol Headers

```
ac_nsl_serv(b, kb): (a, c, d, v)
  rely pubkey(b, kb)
  guarantee supplied(a, c, d, v)
in ... end
```

```
ac_nsl_client(a, ka, b, d): (n_a, c, v)
  rely pubkey(a, ka)
  guarantee c = "hi_cost" and says_curr_val(b, d, v, n_a)
           or c = "lo_cost" and says_approx_val(b, d, v, n_a)
in ... end
```

Subprotocol Headers

```
retrieve_pubkey (b, a, c, cver, d, kd) : (a, ka)
  rely          certifying_authority(c, a)
               and directory_service(d, c)
               and pubkey(d, kd)
               and sign_verification_key_of(c, cver)
  guarantee pubkey(a, ka)
in
...
end
```

Precondition/postcondition
specifies effect of
successful run of subprotocol

Subprotocol call site

call with

pubkey(a, ka)

--> null_protocol () ()

true

use key ka...

|

certifying_authority(c, a)

and directory_service(d, c)

and pubkey(d, kd)

and sign_verification_key_of(c, cver)

--> retrieve_pubkey (b, a, c, cver, d, kd) (a, ka)

pubkey(a, ka)

use key ka...

Trust and Protocols

- Crypto protocols coordinate principals
 - Agree on parameter values
 - Agree on assertions made
- Trust decisions at runtime can control protocol behavior
 - Stop protocol run if trust constraints fail
 - Choose branch conditional on successful trust constraint
 - Message transmissions and subprotocol calls
- Strand-based semantics
 - Provides good protocol verification methods, design heuristics
 - Motivated language design and implementation
- Status: Second version of compiler now complete
 - Datalog trust engine, Crypto library
- Trust engineering using cryptographic protocols

<http://www.ccs.neu.edu/home/guttman>

Contrast: Earlier Work

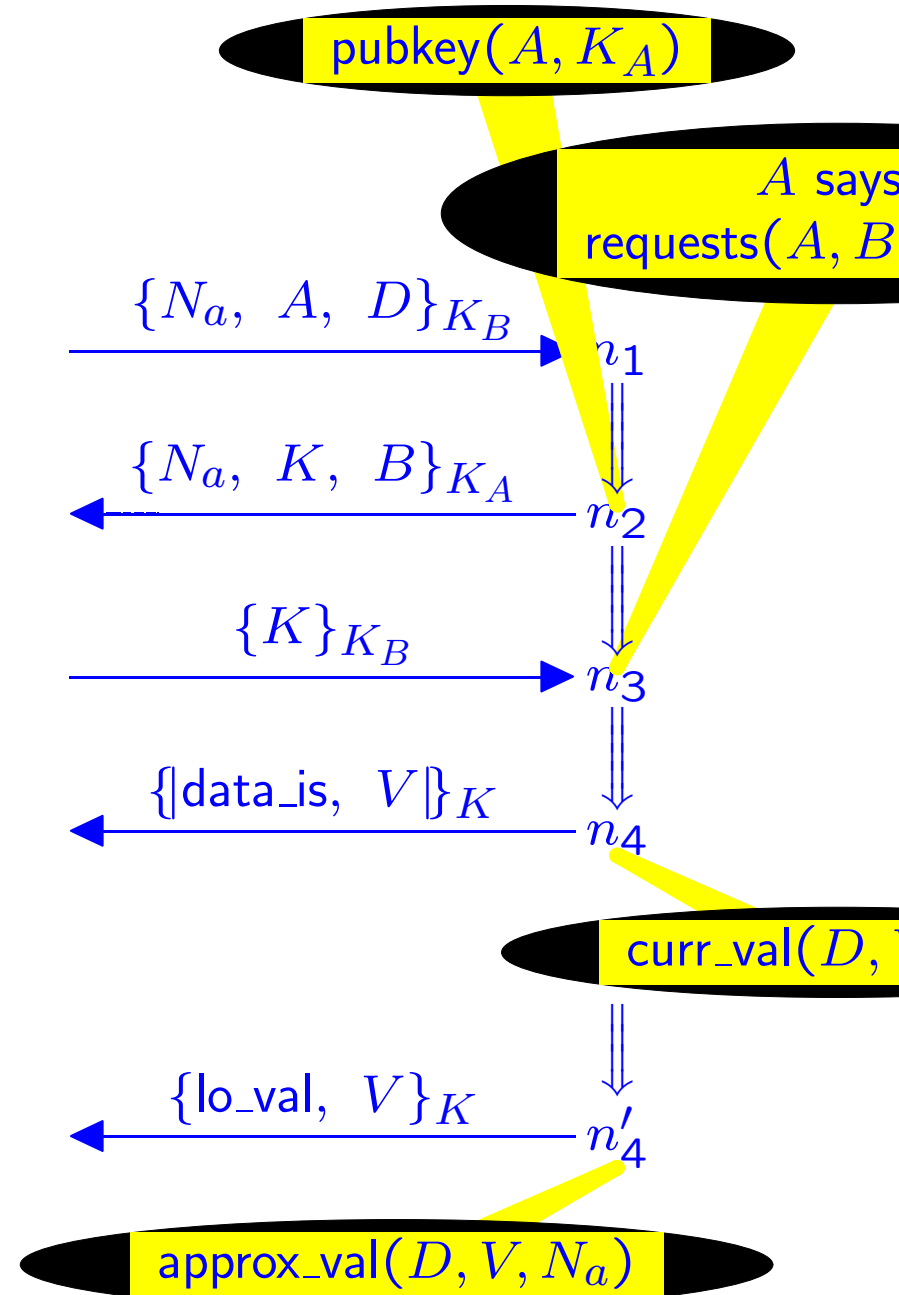
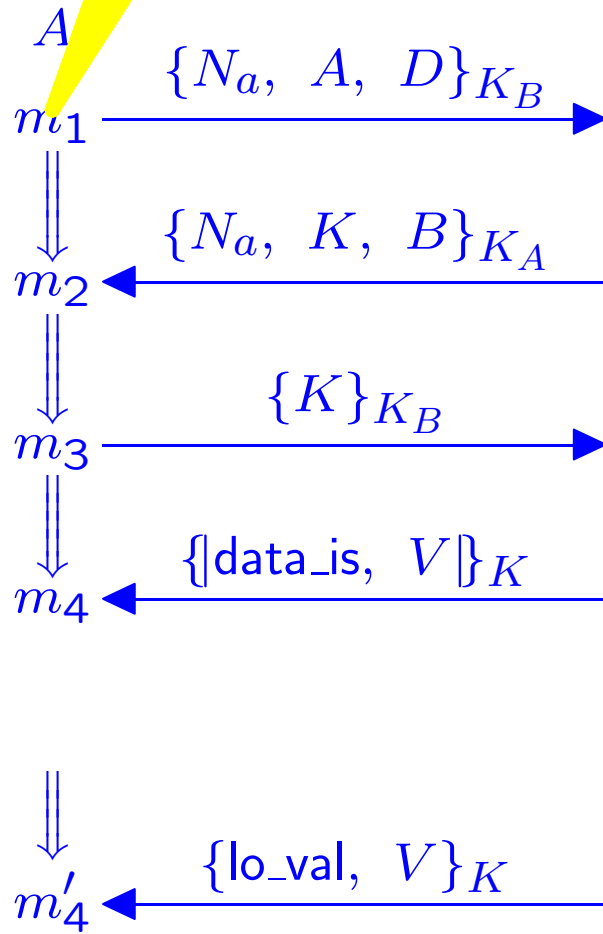
- The BAN tradition
 - Messages **are** formulas or formulas **idealize** messages
 - Who asserted the formulas?
 - Who drew consequences from formulas?
- Embedding formulas explicitly inside messages
 - Main view of logical trust mgt
 - Formulas parsed out of certificates
 - Problem of partial information?
- Our view: Formulas part of transmission/reception, not msg
 - Compatible with many insights of earlier views
 - Independent method to determine what events happened
 - Clarity about who makes assertions, who infers consequences
 - Partial information easy to handle
 - Rigorous notion of **soundness**



starts
with LAWB

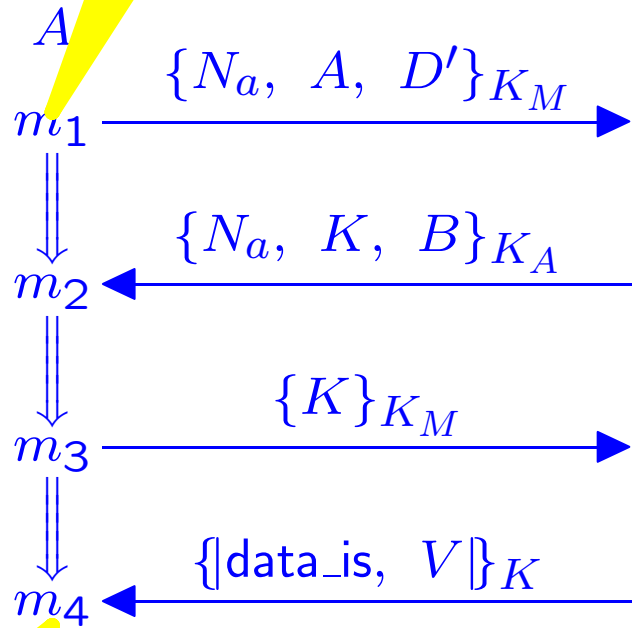
NSL Access

requests(A, B, D, N_a)

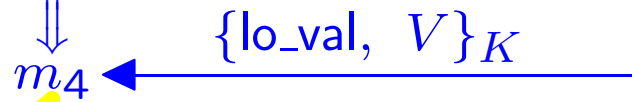


NSL Access Failure

requests(A, P, D', N_a)



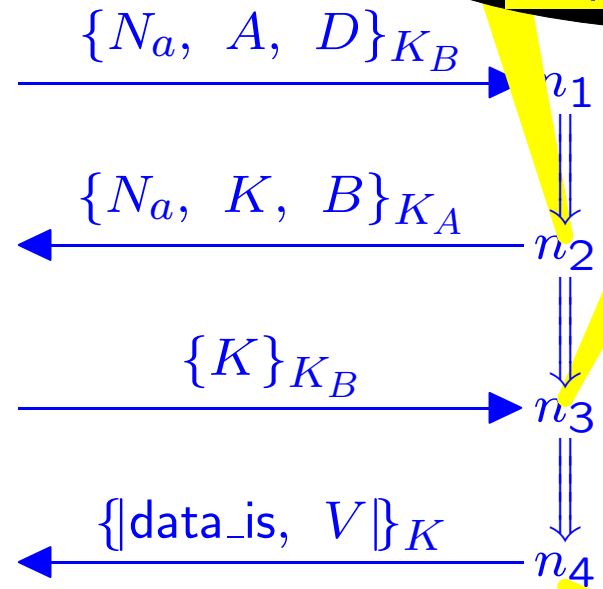
P says
curr_val(D', V, N_a)



P says
approx_val(D', V, N_a)

pubkey(A, K_A)

A says
requests(A, B)



corp_subscriber(
curr_val($D,$



home_subscriber(A, D) and
approx_val(D, V, N_a)

Soundness

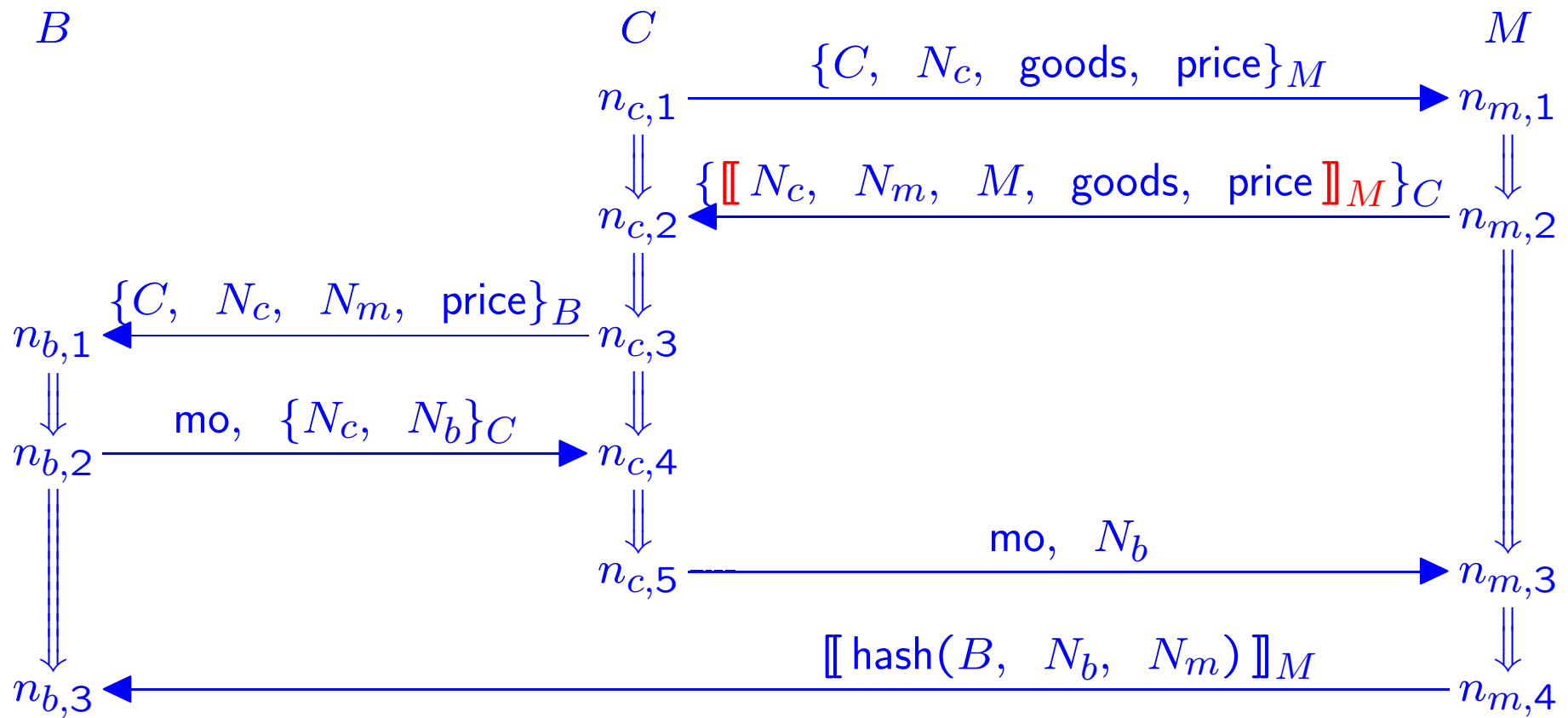
- Protocol Π is **sound** if:
for all executions \mathcal{B} , and message receptions $n \in \mathcal{B}$

$$\{\text{prin}(m) \text{ says } \gamma_m : m \prec_{\mathcal{B}} n\} \longrightarrow_{\mathcal{L}} \rho_n$$

where $\longrightarrow_{\mathcal{L}}$ is the consequence relation of the underlying logic

- Soundness follows from authentication properties
 - Strand space authentication methods perfect
 - Recency easy to incorporate

A Signed Alternate: SEPMO



Signed Electronic Purchase using Money Order

$$\text{mo} = \llbracket \text{hash}(C, N_c, N_b, N_m, \text{price}) \rrbracket_B$$