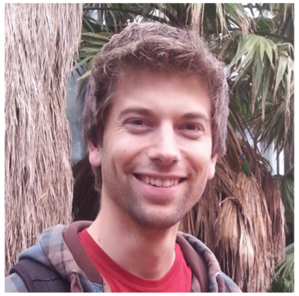
Modelling and Analysis of a Hierarchy of Distance Bounding Attacks

Tom Chothia, Joeri de Ruiter and Ben Smyth

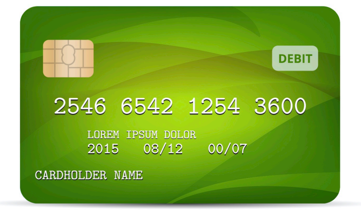


Introduction

- A unified framework for distance bounding attacks.
- Examples: Contactless EMV & NXP’s DB protocol.
- A modelling language for DB protocols.
- A hierarchy of security properties, matched to particular attacker models.
- Automatically checking previously defined symbolic properties.


09-25-2017 Mon 01:01:25
Camera 01
WEST MIDLANDS POLICE

Core EMV Protocol



Shop

Card

Bank's
Verification key

Kbank: 3-DES key shared with bank
Kcard: an RSA public
Kcert : Bank cert. for Kcard

UN, amount, currency, . . .

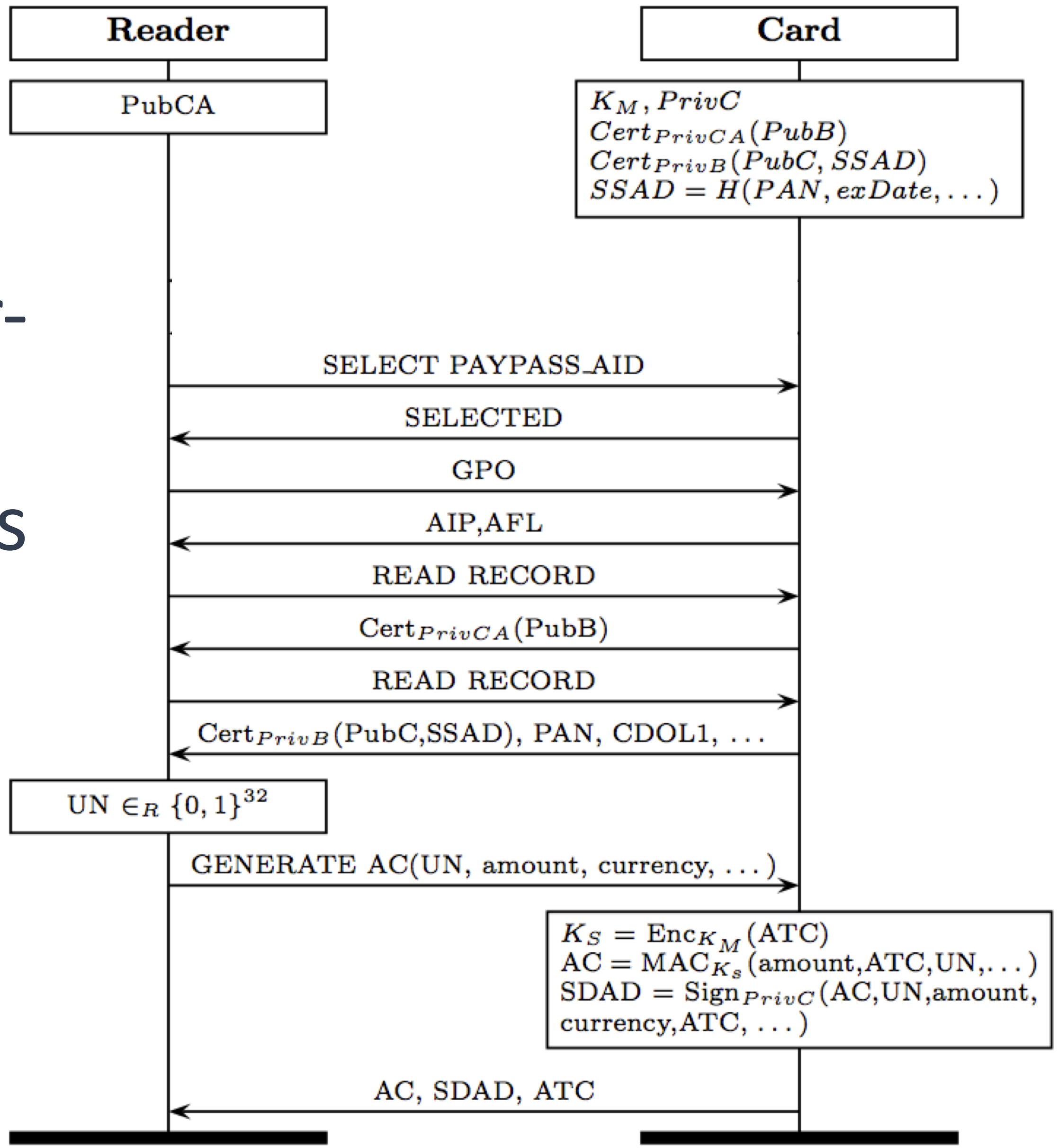
Generate nonce: Nc
Session key based on ATC: Ks=Enc_{Kbank}(ATC)

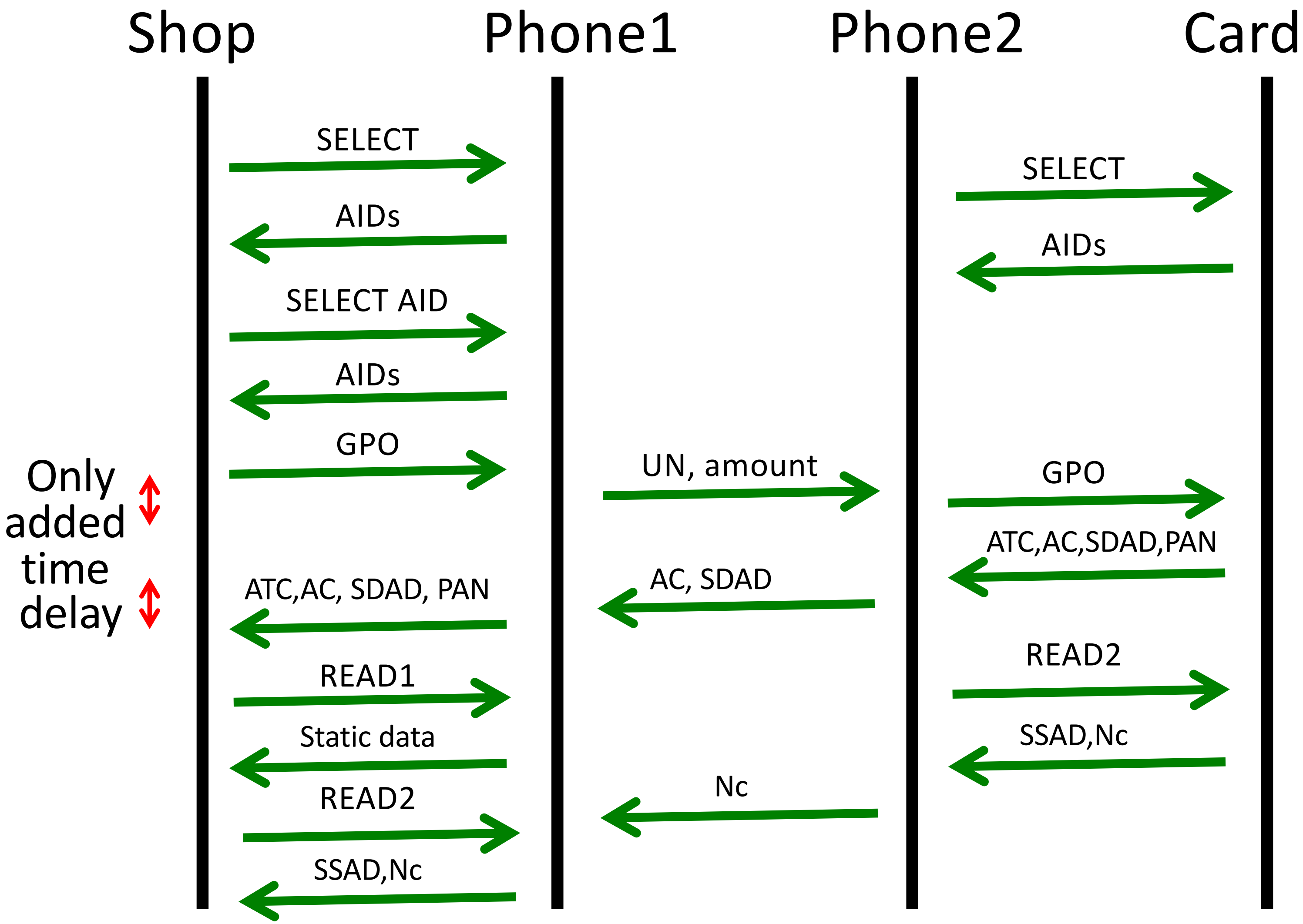
MAC_{Ks}(amount,currency,UN,..)
$Sign_{Kcard}$ (amount,currency,UN,Nc..)

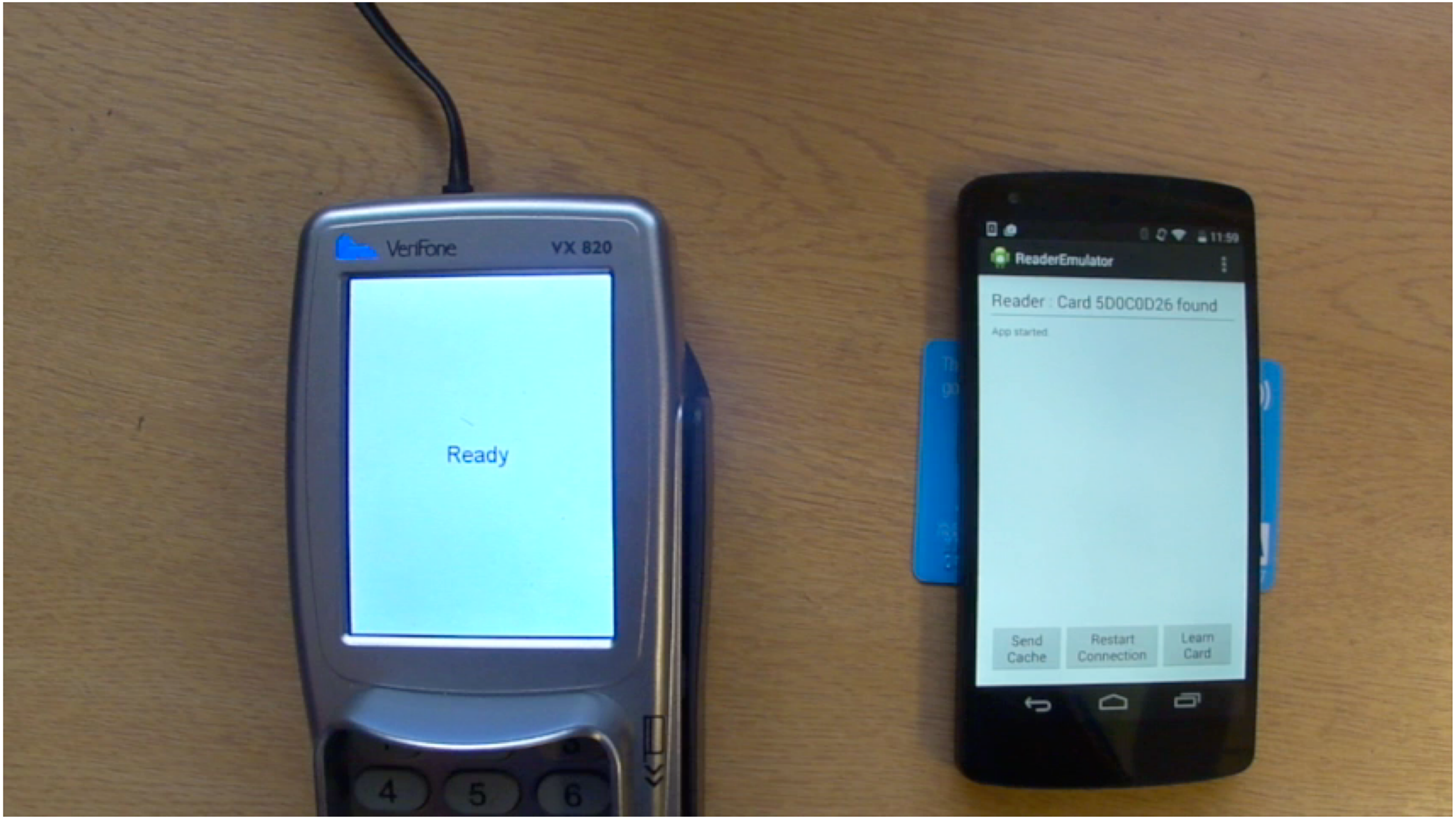
Kcert, ATC

AC, ATC

Master-card's PayPass




Shop
Phone1
Phone2
Card
SELECT
AIDs
SELECT AID
AIDs
GPO
Only added time delay
ATC,AC, SDAD, PAN
READ1
Static data
READ2
SSAD,Nc
UN, amount
AC, SDAD
Nc
SELECT
AIDs
GPO
ATC,AC,SDAD,PAN
READ2
SSAD,Nc

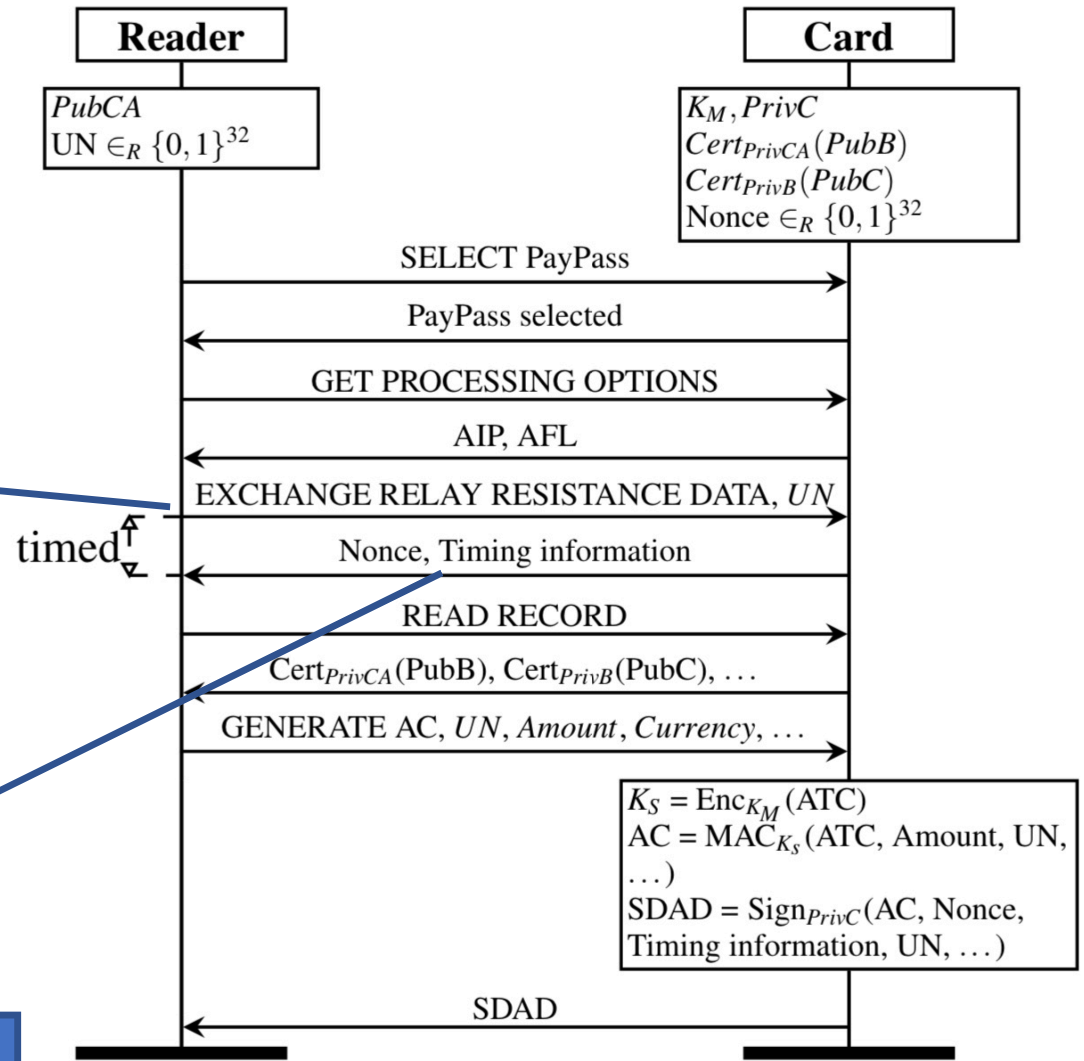

VeriFone
VX 820
Ready
ReaderEmulator
Reader : Card 5D0C0D26 found
App started
Send Cache
Restart Connection
Learn Card

MasterCard's Relay Resistance Protocol (RRP) (similar to PaySafe)

Uses New Command

Timing profile sent by card

We check this as auth. property

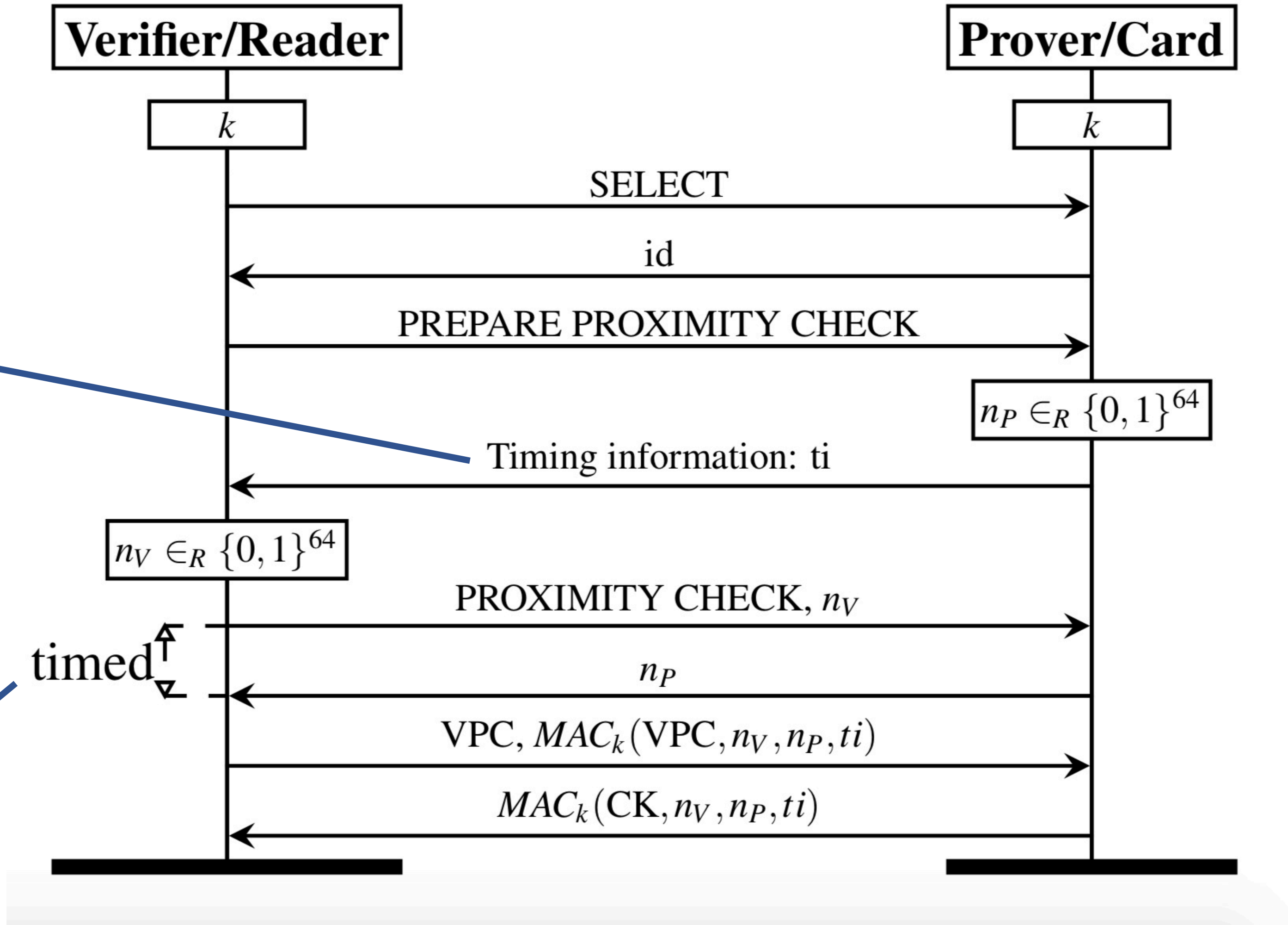


NXP distance bounding protocol

- NXP sell a distance bounding smart card.
- NXP have patented a distance bounding ☺
- Patent documents are really hard to read ☹

“This need may be met by the subject matter according to the independent claims. Advantageous embodiments of the present invention are set forth in the dependent claims.”

NXP Protocol.



Only in one version

Can be split into 8 one bytes message

Some Questions

- How can we formally (symbolically) define these protocols?
- How can we say if these protocols are “secure”?
- What does “secure” even mean in this context?

Our modelling language for DB

```
in (x).P
out <x>.P
P | Q
!P
new a.P
let x = D in P else Q
event(X).P
startTimer.P
stopTimer.P
```

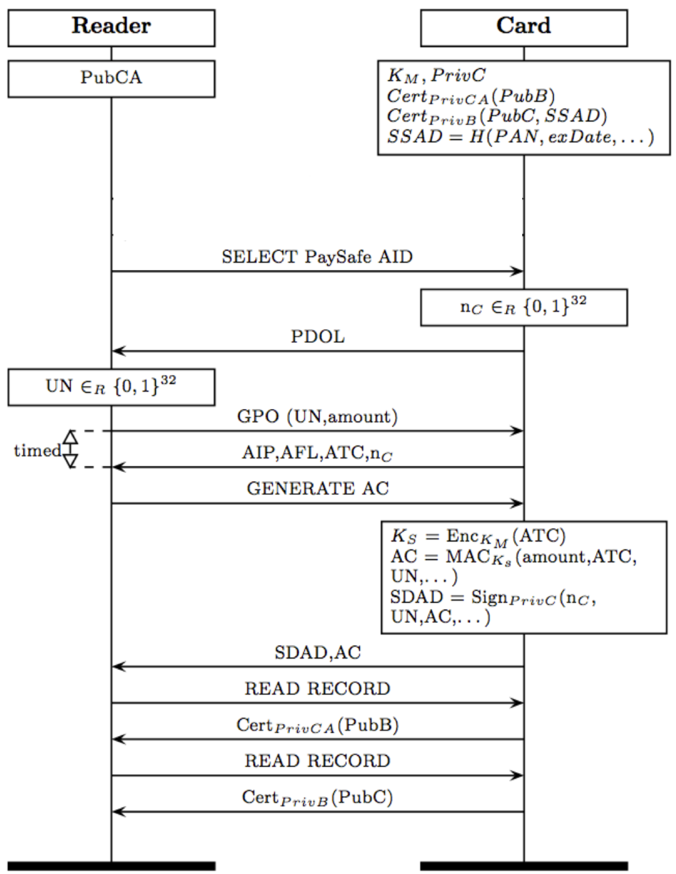
Locations: `L = [ P ]` or `L | L`

Eg.

```
[ EMVCard ] | [ ShopReader ]
  [ EMVCard | ShopReader ]
```

PaySafe Model

```
let Verifier =
  out c<SELECT,AID>.
  in c(pdol).
  new UN.
  out c<GET_PROCESSING_OPTIONS,UN,amount>.
  in c(aip,afl,NC).
  out c<GENERATE_AC>.
  in c(SDAD,AC).
  out c<READ_RECORD>.
  in c(cCert).
  let cKey, cId = checksign(cCert,getPubKey(BANK_ID)) in
  let (=UN,=NC,=rAmount,ATC,AC)=checksign(SDAD,cKey) in
  event Verified(cId).
```



PaySafe Model

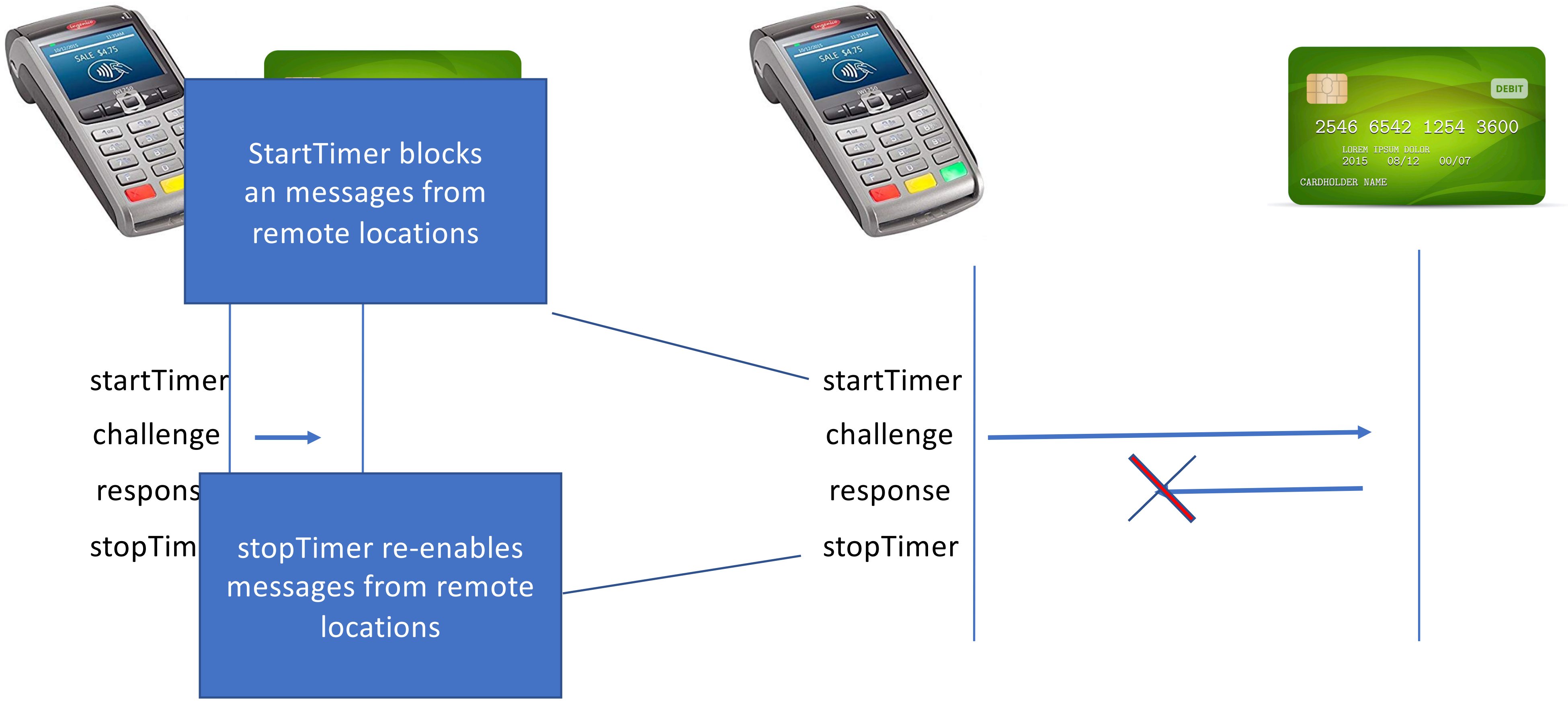
Unbounded number ids each for an unbounded number of runs

```
Verifiers = !(new amount.!Verifier)
Provers =  !(new id. let idP = id in
            let cCert = sign(getPubKey(idP), idP),
                            getPrivKey(BANK_ID)) in
            !event Start(idP). Prover ]
```

```
let Verifier =
  out c<SELECT,AID>.
  in c(pdol).
  new UN.
  out c<GET_PROCESSING_OPTIONS,UN,amount>.
  in c(aip,afl,NC).
  startTimer. out c<GENERATE_AC>.
  in c(SDAD,AC). stopTimer.
  out c<READ_RECORD>.
  in c(cCert).
  let cKey, cId = checksign(cCert,getPubKey(BANK_ID)) in
  let (=UN,=NC,=rAmount,ATC,AC)=checksign(SDAD,cKey) in
  event Verified(cId).
```

```
[ Verifiers ] | [ Provers ]

[ Verifiers | Provers ]
```



StartTimer blocks an messages from remote locations
startTimer
challenge
respons
stopTim
stopTimer re-enables messages from remote locations
startTimer
challenge
response
stopTimer
DEBIT
2546 6542 1254 3600
LOREM IPSUM DOLOR
2015 08/12 00/07
CARDHOLDER NAME

Key observation: The semantics just needs to block outputs from remote locations while a timer is running

We write [Process]$_{\text{<number of timers running>}}$

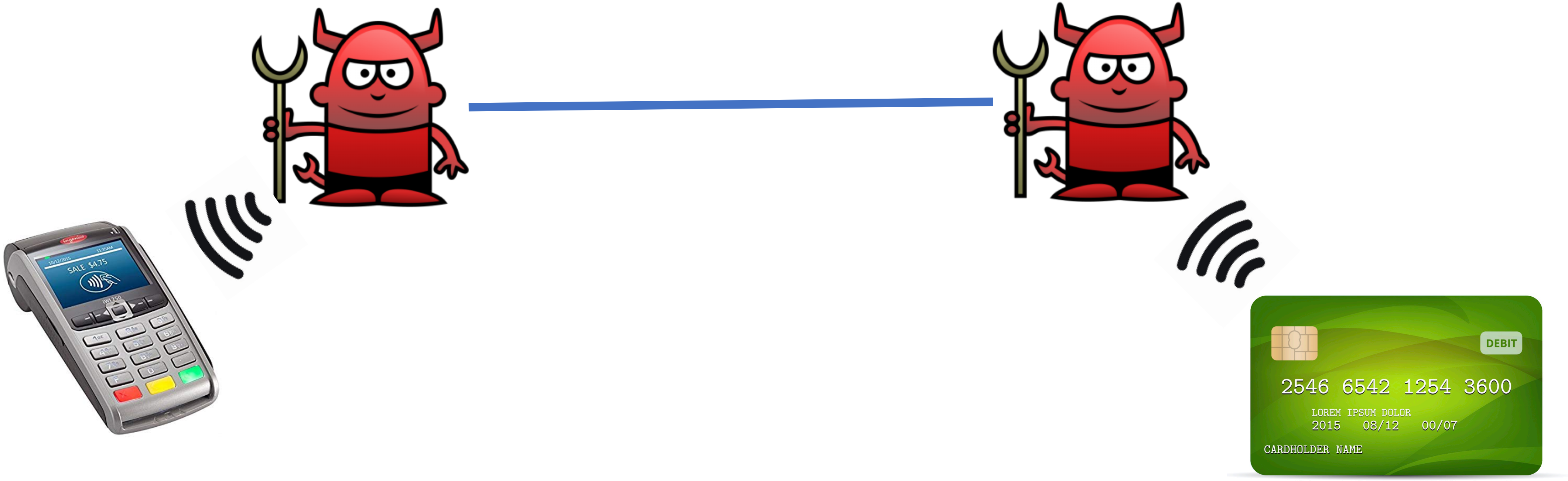
[in c(x).P | out c<n>.Q]$_r$ -> [P{n/x} | Q]$_r$

[out c<n>.Q]$_r$ | [P]$_0$ -> [Q]$_r$ | [out c<n>|P]$_0$

[out c<n>.Q]$_r$ -> [out c<n> | Q]$_r$

Definitions for the symbolic literature

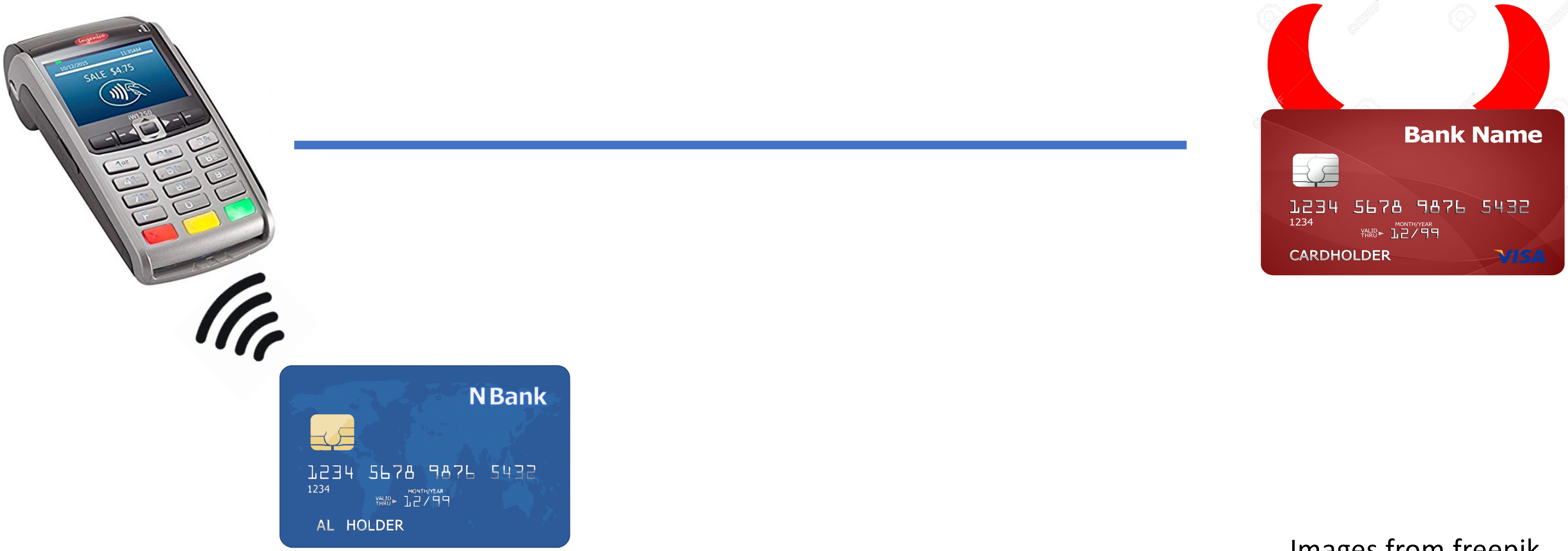
Relay/Mafia Fraud: attackers relay and interfere with messages



Images from freepik

Distance Fraud: remote dishonest prover tricks the verifier

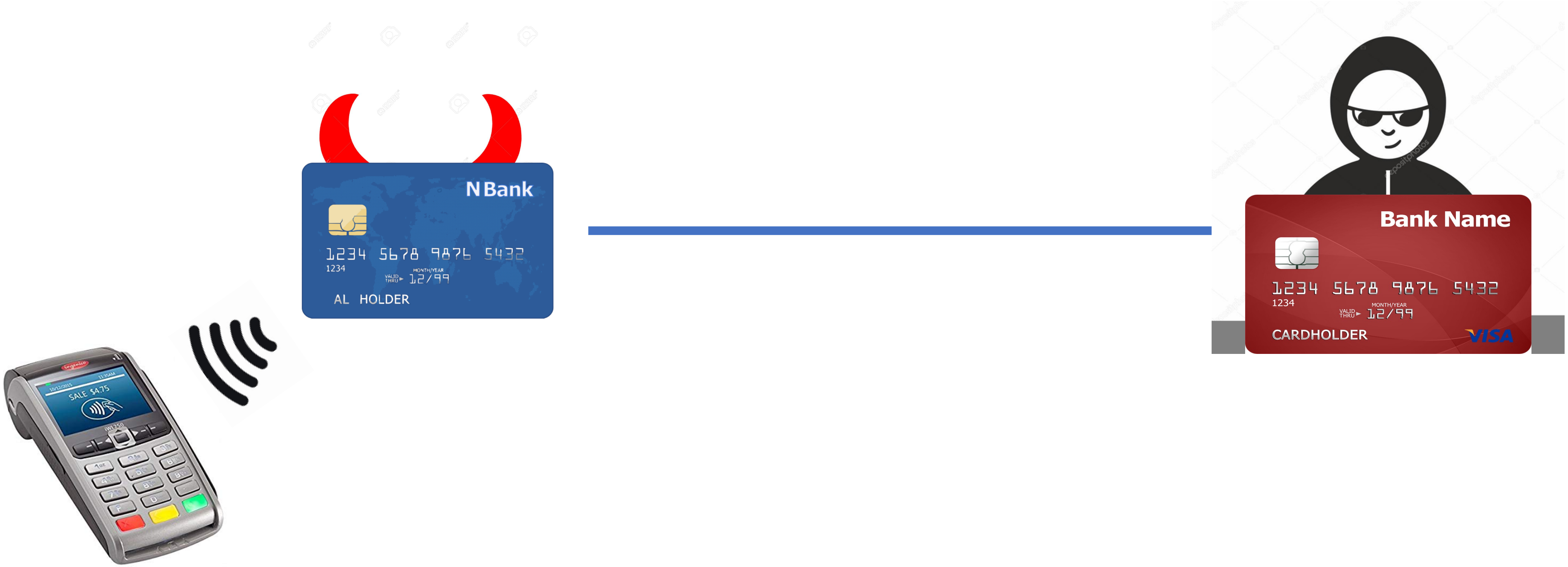
Distance Hijacking: remote dishonest prover uses a local honest prover



Images from freepik

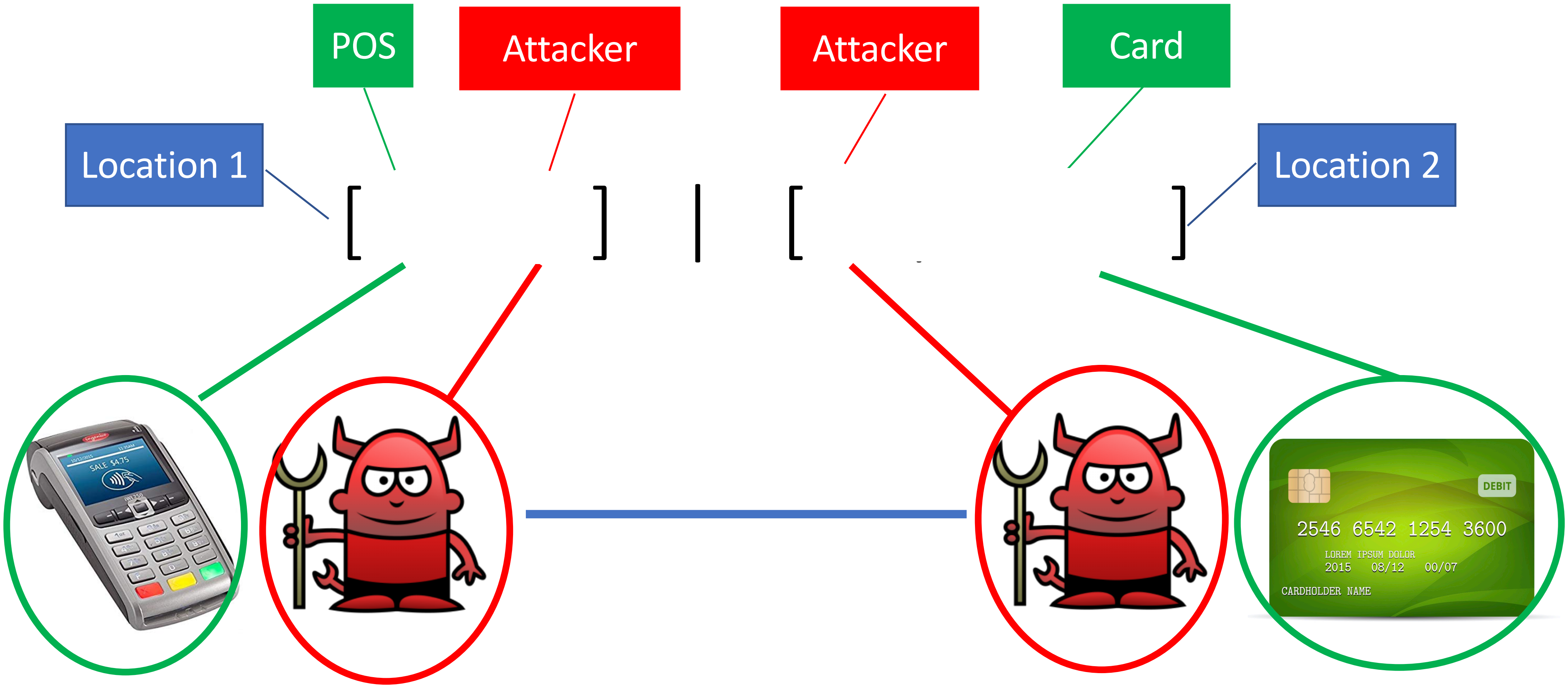
Terrorist Fraud: A remote dishonest prover* and local attacker

Assisted Distance Fraud: remote dishonest prover* and local dishonest prover



Definitions for the symbolic literature

- Relay/Mafia Fraud: attackers relay and interfere with messages
- Lone Distance Fraud: remote dishonest prover tricks the verifier
- Distance Hijacking: remote dishonest prover uses a local honest prover
- Terrorist Fraud: A remote dishonest prover* and local attacker
- Assisted Distance Fraud: remote dishonest prover* and local dishonest prover



POS
Attacker
Attacker
Card
Location 1
Location 2
SALE $4.75
DEBIT
2546 6542 1254 3600
LOREM IPSUM DOLOR
2015 08/12 00/07
CARDHOLDER NAME

Relay Attack

- There exists relay attack against the protocol P and V if there exists A such that

```
[V(id)|A] | [P(id)|A]
```

I.e.

```
[ V | A ] | [P(id) | A]
->* [X] | [ new id.Q | Y ]
->  [X] | [ Q{a/id}  | Y ]
    [ event verified(a).R | W] | [Z]
```

Distance Fraud

- Dishonest prover DP-A(id) = !new id.<board cast all secret values> | A

- **Lone Dist**
  verifier.

E.g.: For RRP:
DP-A(id) = A | ! new id. out c<id>!
let cert = sign((getPubKey(id), id), getPrivKey(BANK_ID)) in
out c<getPrivKey(id), cert, sharedKey(id))).

- **Distance Hijacking**: remote dishonest prover uses a local honest prover

[V(id)|P(id')] | [DP-A(id)]

Terrorist Frauds

- Terrorist
- **Terrorist**
- **Assisted**
- **Assisted**

```
E.g.:  For RRP:
TP–A(id) =  A | ! new id. out c<id>. (
                ! in c (atc, message);
                  let macKey=genKey(atc, sharedKey(idP)) in
                  let messageMAC = mac(message, macKey) in
                  out c<messageMAC>
              | ! in c(message);
                  let signed=sign(message,getPrivKey(id)) in
                  out c<signed)>
              | out c<cardCert, id>.
```

[V(id)|DP-A(id’)] |[TP-A(id)]

Assisted Distance Fraud
[V(id)|DP-A(id')] |[TP-A(id)]

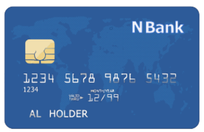
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]

Distance Fraud
[V(id)] | [DP-A(id)]

Terrorist Fraud
[V(id)|A] | [TP-A(id)]

Mafia fraud/Relay
[V(id)|A] | [P(id)|A]

Our Building Blocks

- Arbitrary number of provers  P(id)
- Verifier looking for one of “id”  V(id)
- A Dolev–Yao attacker  A
- Other Provers 

P(id’)

Trying to trick verifier

- A dishonest prover 

DP-A(id)
- A terrorist prover 

TP-A(id)

Verifier doesn’t care about

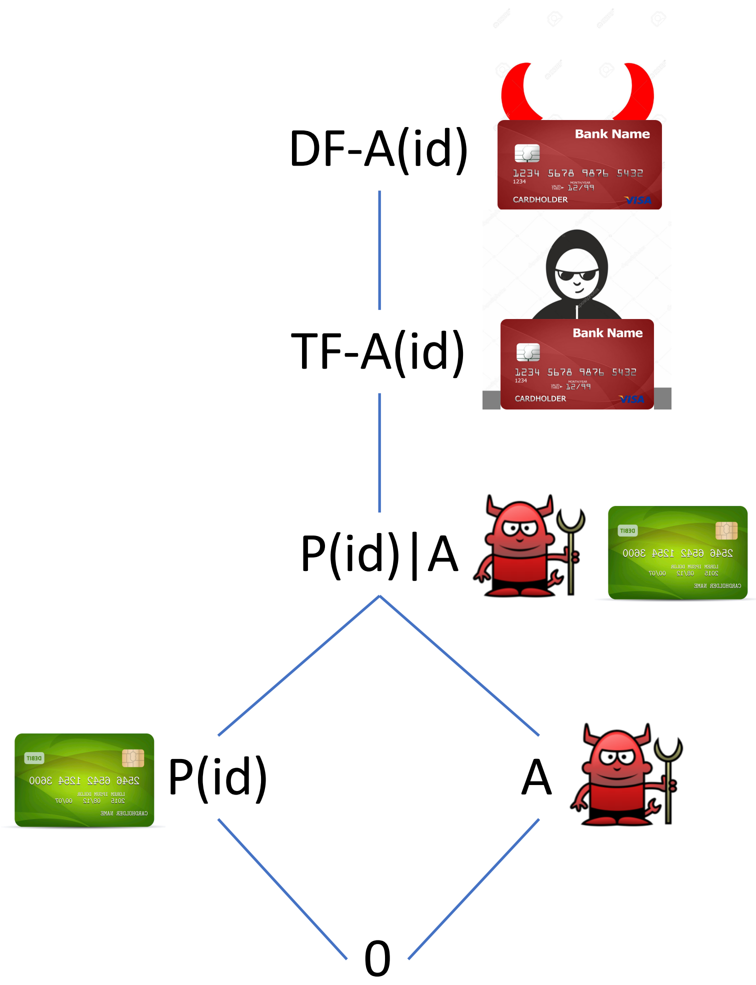
- A dishonest prover 

DP-A(id’)
- A terrorist prover 

TP-A(id’)

Ordering the Properties

- Our building blocks form a hierarchy.
- Each level is strictly more expressive than the one below.
- Replacing any process with the one above it, at a particular location, makes the attacker more powerful.

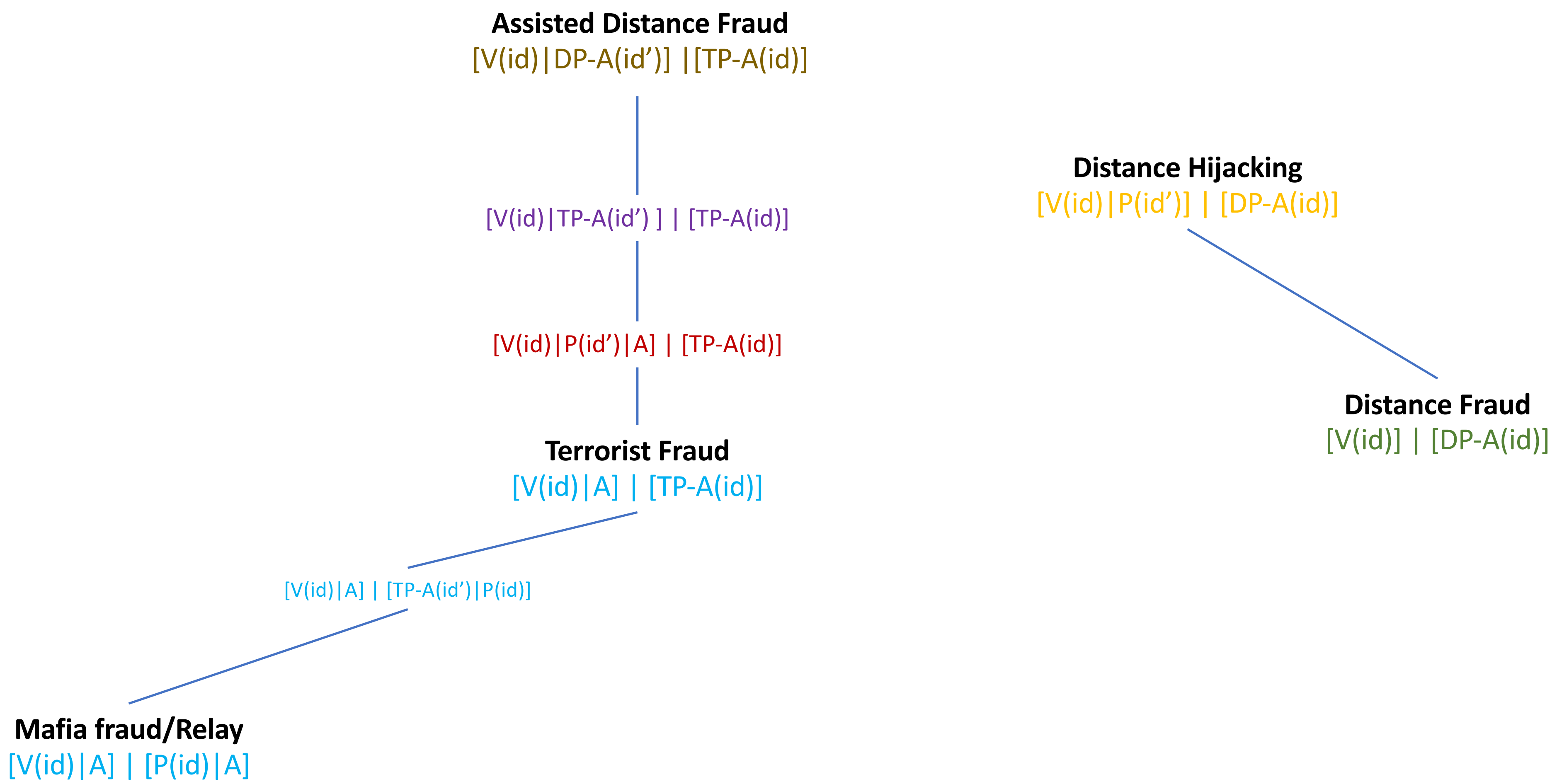


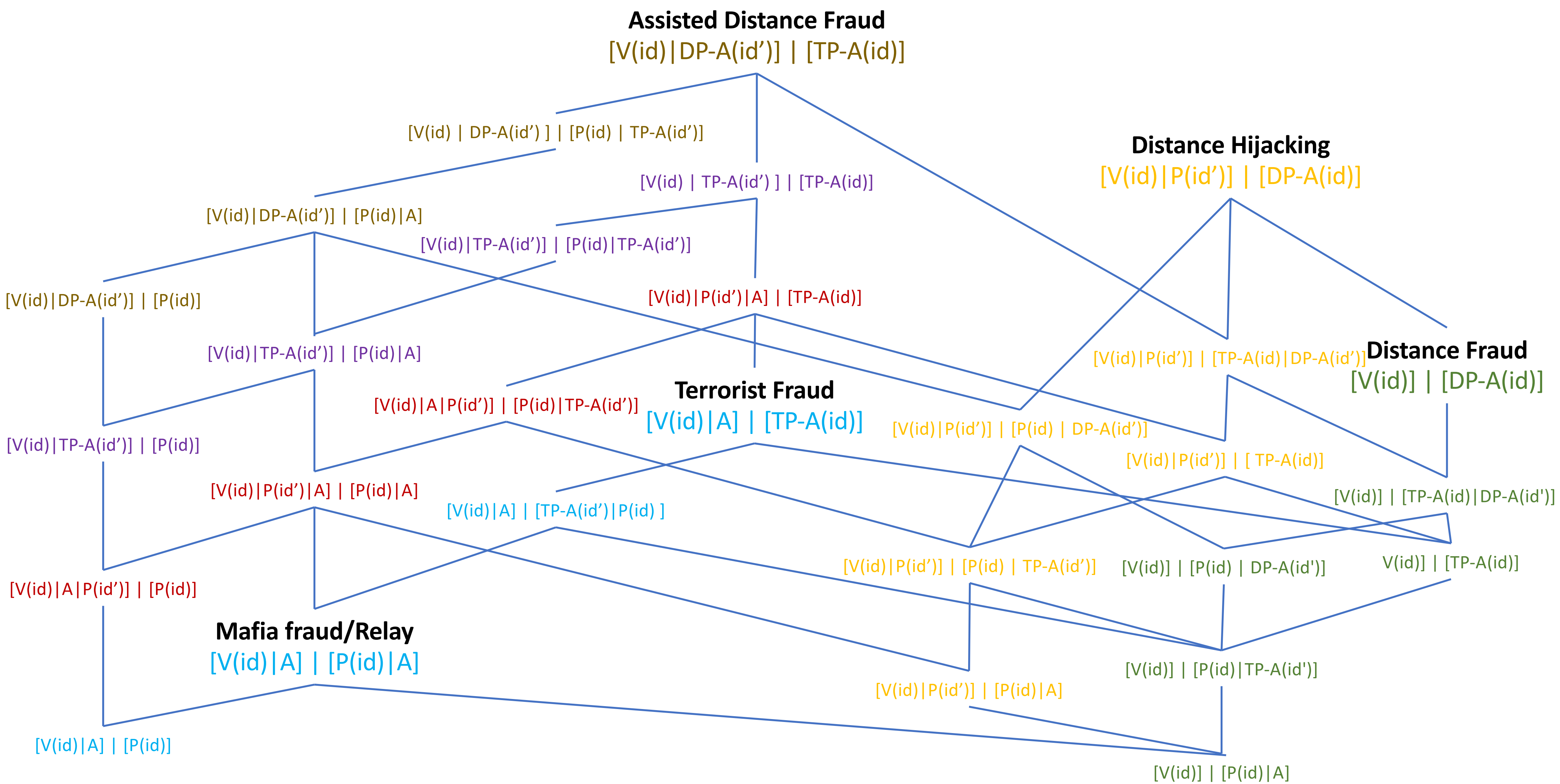
Equalities between processes

$$[V(id)\,|\,A] \;\;|\;\; [P(id)\,|\,A]$$

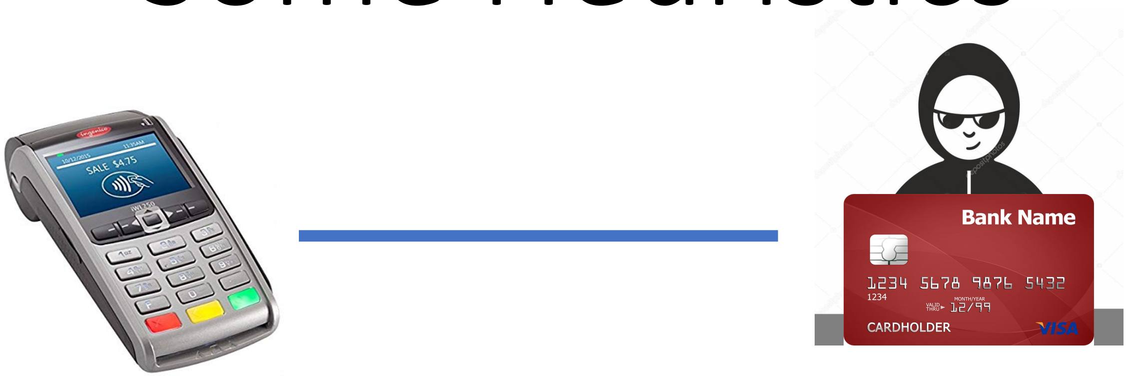
$$=$$

$$[V(id)\,|\,A] \;\;|\;\; [P(id)\,|\,A\,|\,P(id')]$$


Assisted Distance Fraud
[V(id)|DP-A(id')] |[TP-A(id)]
[V(id)|TP-A(id')] | [TP-A(id)]
[V(id)|P(id')|A] | [TP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
[V(id)|A] | [TP-A(id')|P(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]

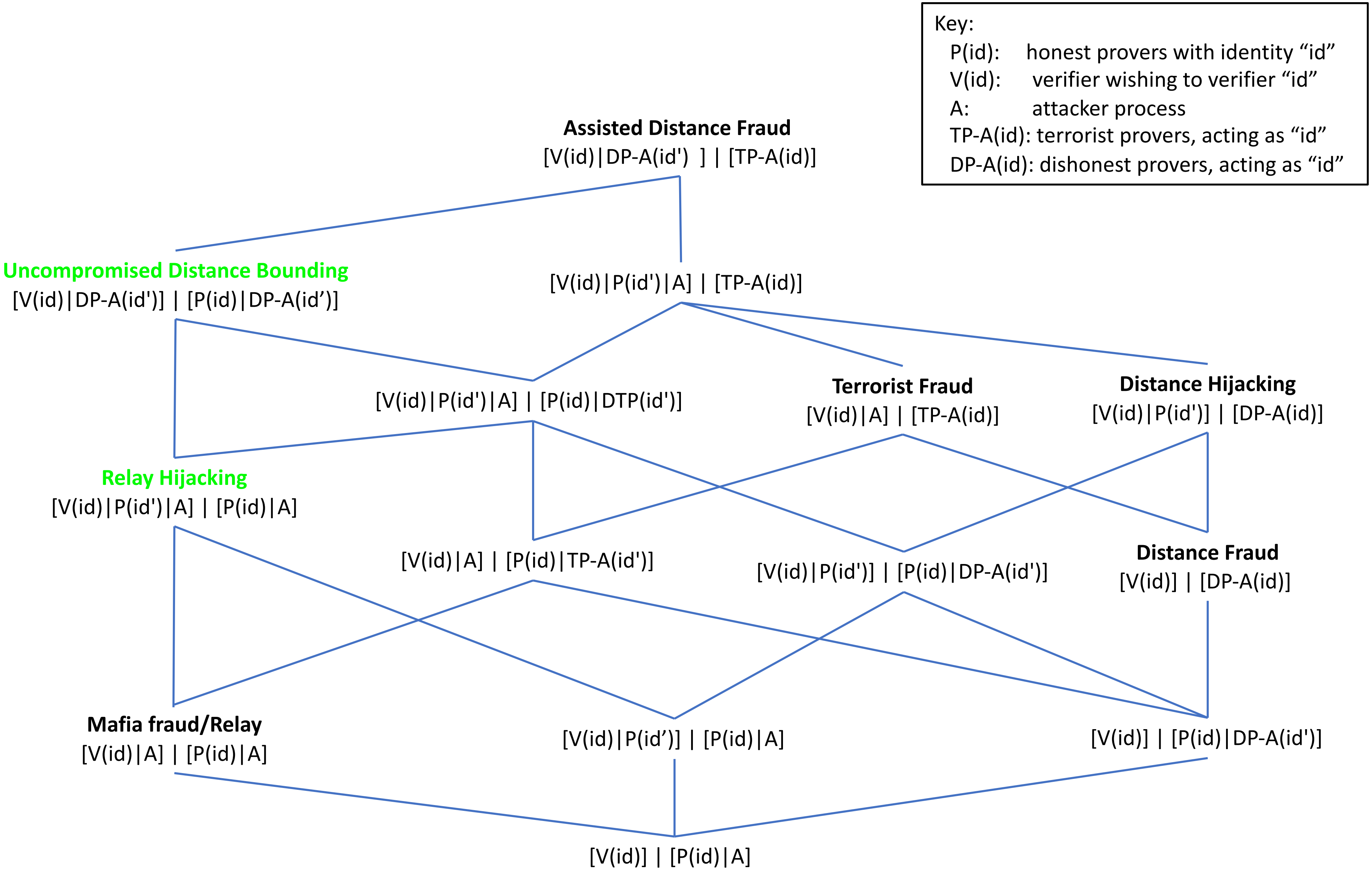

Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
[V(id) | DP-A(id')] | [P(id) | TP-A(id')]
[V(id) | TP-A(id')] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
[V(id)|DP-A(id')] | [P(id)|A]
[V(id)|TP-A(id')] | [P(id)|TP-A(id')]
[V(id)|DP-A(id')] | [P(id)]
[V(id)|P(id')|A] | [TP-A(id)]
[V(id)|TP-A(id')] | [P(id)|A]
[V(id)|P(id')] | [TP-A(id)|DP-A(id')]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
[V(id)|A|P(id')] | [P(id)|TP-A(id')]
[V(id)|P(id')] | [P(id) | DP-A(id')]
[V(id)|TP-A(id')] | [P(id)]
[V(id)|P(id')] | [TP-A(id)]
[V(id)|P(id')|A] | [P(id)|A]
[V(id)|A] | [TP-A(id')|P(id)]
[V(id)] | [TP-A(id)|DP-A(id')]
[V(id)|P(id')] | [P(id) | TP-A(id')]
[V(id)] | [P(id) | DP-A(id')]
V(id)] | [TP-A(id)]
[V(id)|A|P(id')] | [P(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)] | [P(id)|TP-A(id')]
[V(id)|P(id')] | [P(id)|A]
[V(id)|A] | [P(id)]
[V(id)] | [P(id)|A]

Some Heuristics



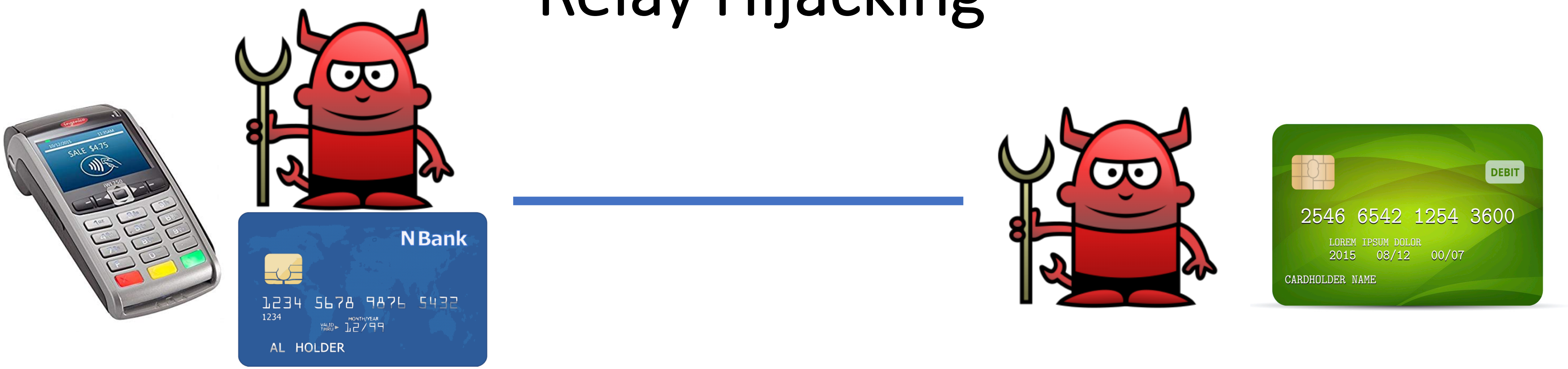
[V(id)] | [TP−A(id)]

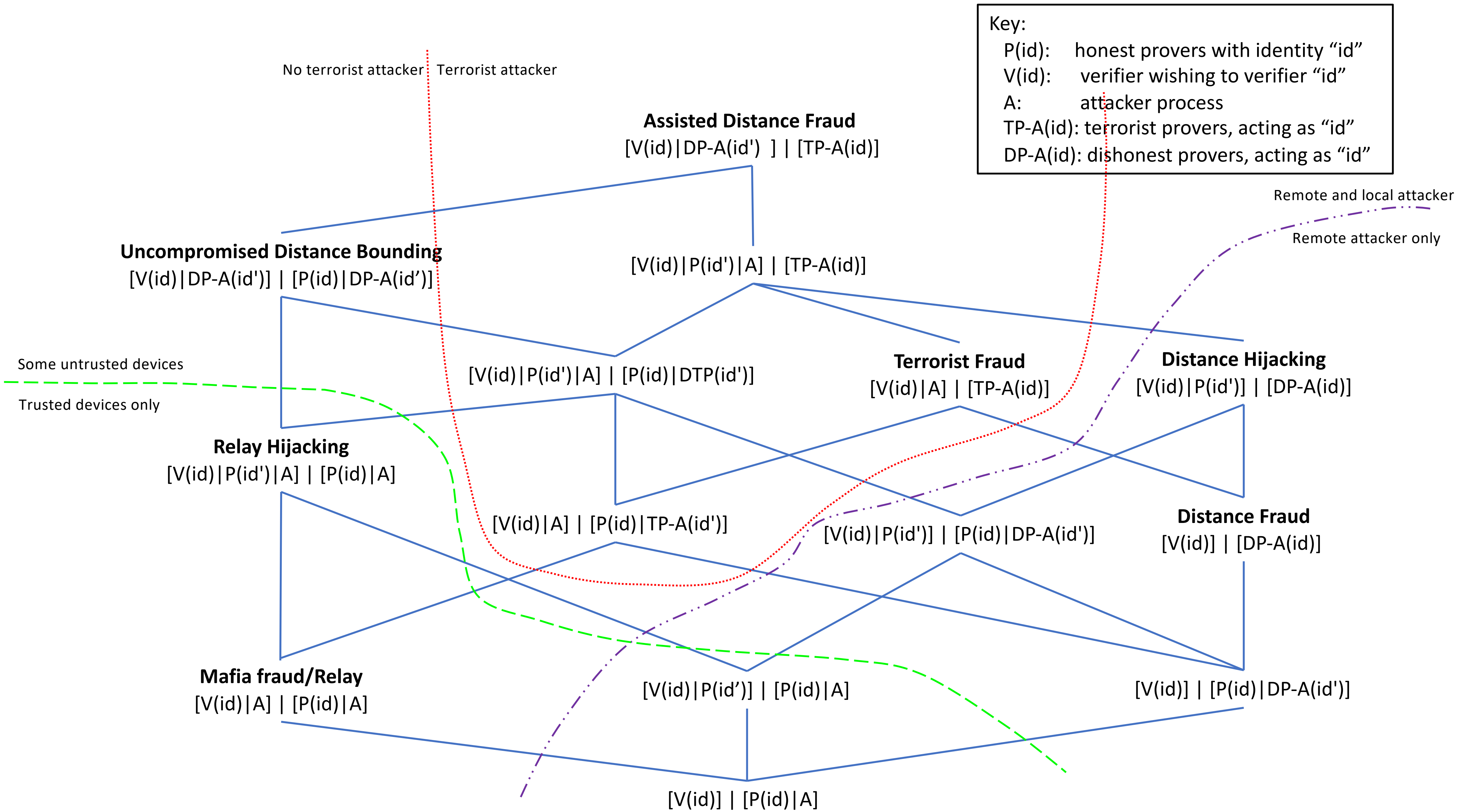
= [V(id)] | [DP−A(id)]

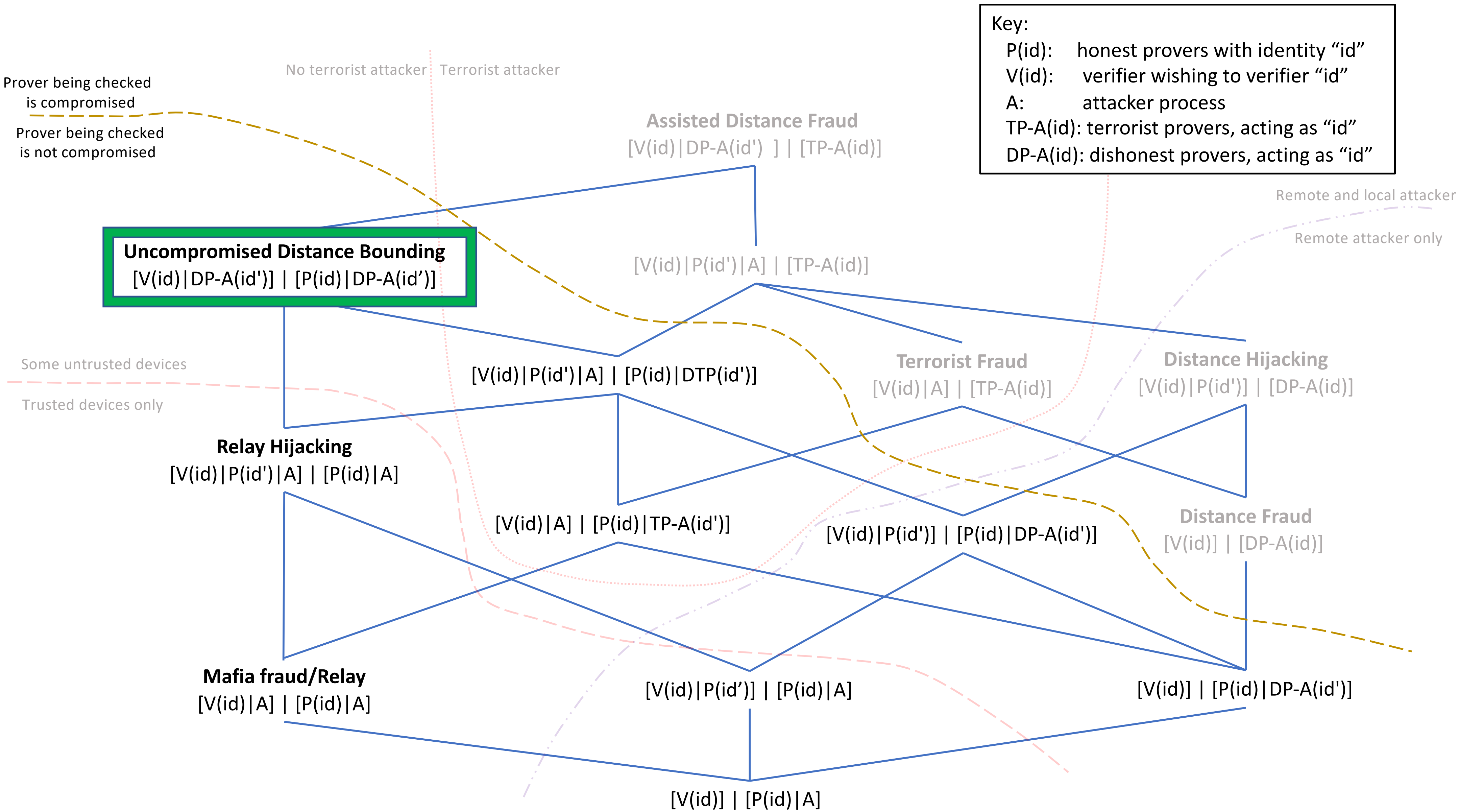

Key:
P(id): honest provers with identity “id”
V(id): verifier wishing to verifier “id”
A: attacker process
TP-A(id): terrorist provers, acting as “id”
DP-A(id): dishonest provers, acting as “id”
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id’)]
[V(id)|P(id')|A] | [TP-A(id)]
[V(id)|P(id')|A] | [P(id)|DTP(id')]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Relay Hijacking
[V(id)|P(id')|A] | [P(id)|A]
[V(id)|A] | [P(id)|TP-A(id')]
[V(id)|P(id')] | [P(id)|DP-A(id')]
Distance Fraud
[V(id)] | [DP-A(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)|P(id’)] | [P(id)|A]
[V(id)] | [P(id)|DP-A(id')]
[V(id)] | [P(id)|A]

Uncompromised Distance Bounding

Relay Hijacking




Key:
P(id): honest provers with identity “id”
V(id): verifier wishing to verifier “id”
A: attacker process
TP-A(id): terrorist provers, acting as “id”
DP-A(id): dishonest provers, acting as “id”
No terrorist attacker
Terrorist attacker
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id’)]
[V(id)|P(id')|A] | [TP-A(id)]
Remote and local attacker
Remote attacker only
Some untrusted devices
Trusted devices only
[V(id)|P(id')|A] | [P(id)|DTP(id')]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Relay Hijacking
[V(id)|P(id')|A] | [P(id)|A]
[V(id)|A] | [P(id)|TP-A(id')]
[V(id)|P(id')] | [P(id)|DP-A(id')]
Distance Fraud
[V(id)] | [DP-A(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)|P(id’)] | [P(id)|A]
[V(id)] | [P(id)|DP-A(id')]
[V(id)] | [P(id)|A]



Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Prover being checked is compromised
Prover being checked is not compromised
No terrorist attacker
Terrorist attacker
Remote and local attacker
Remote attacker only
Some untrusted devices
Trusted devices only
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id')]
[V(id)|P(id')|A] | [TP-A(id)]
[V(id)|P(id')|A] | [P(id)|DTP(id')]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Relay Hijacking
[V(id)|P(id')|A] | [P(id)|A]
[V(id)|A] | [P(id)|TP-A(id')]
[V(id)|P(id')] | [P(id)|DP-A(id')]
Distance Fraud
[V(id)] | [DP-A(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)|P(id')] | [P(id)|A]
[V(id)] | [P(id)|DP-A(id')]
[V(id)] | [P(id)|A]

Automatically Checking

- We translate our DB calculus into the applied pi-calculus, and use ProVerif to check processes automatically.

- The translation uses 3 phases:
  - Phase 1, before the timer start
  - Phase 2, while the timer is running
  - Phase 3, after the time stops.

startTimer jumps from phase 1 to phase 2.

stopTimer jumpes from phase 2 to phase 3.

Process at the same location as the verifier can act in all phases

Process at a different location can only act in Phase 1 and Phase 2.

Demo

	Mafia Fraud / Relay	Uncompromised Distance Bounding	Distance Fraud	Terrorist Fraud	Timing information authenticity
PaySafe	OK	OK	Attack	Attack	N/A
PaySafe with changes [28]	OK	OK	OK	Attack	N/A
MasterCard's RRP	OK	OK	Attack	Attack	OK
NXP's protocol (unique keys)	OK	OK	Attack	Attack	OK
NXP's protocol (global key)	OK	Attack	Attack	Attack	OK
NXP's variant 1 (unique keys)	OK	OK	Attack	Attack	N/A
NXP's variant 2 (unique keys)	OK	OK	Attack	Attack	N/A
Meadows et al. [30]	OK	OK	OK	Attack	N/A
MAD (One-Way) [36]	OK	OK	OK	Attack	N/A
CRCS [32]	OK	OK	OK	Attack	N/A
Hancke and Kuhn [24] Poulidor [35] Tree-based [5] Uniform [29]	OK	OK	OK	OK	N/A

Conclusion

- A unified framework for distance bounding attacks.
- Examples: Contactless EMV & NXP’s DB protocol.
- A modelling language for DB protocols.
- A hierarchy of security properties, matched to particular attacker models.
- Automatically checking previously defined symbolic properties.