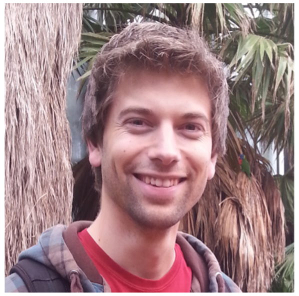
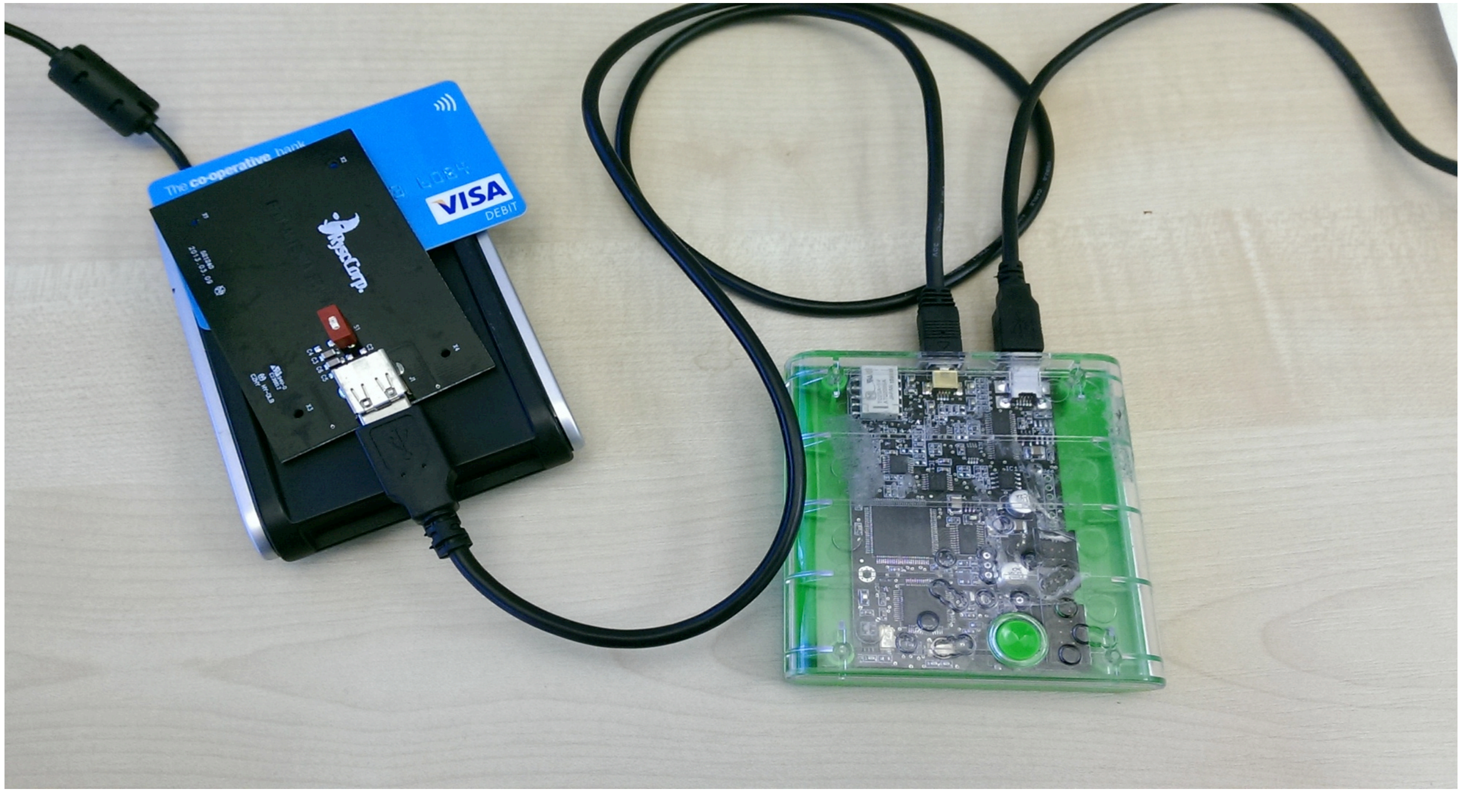
Modelling and Analysis of a Hierarchy of Distance Bounding Attacks

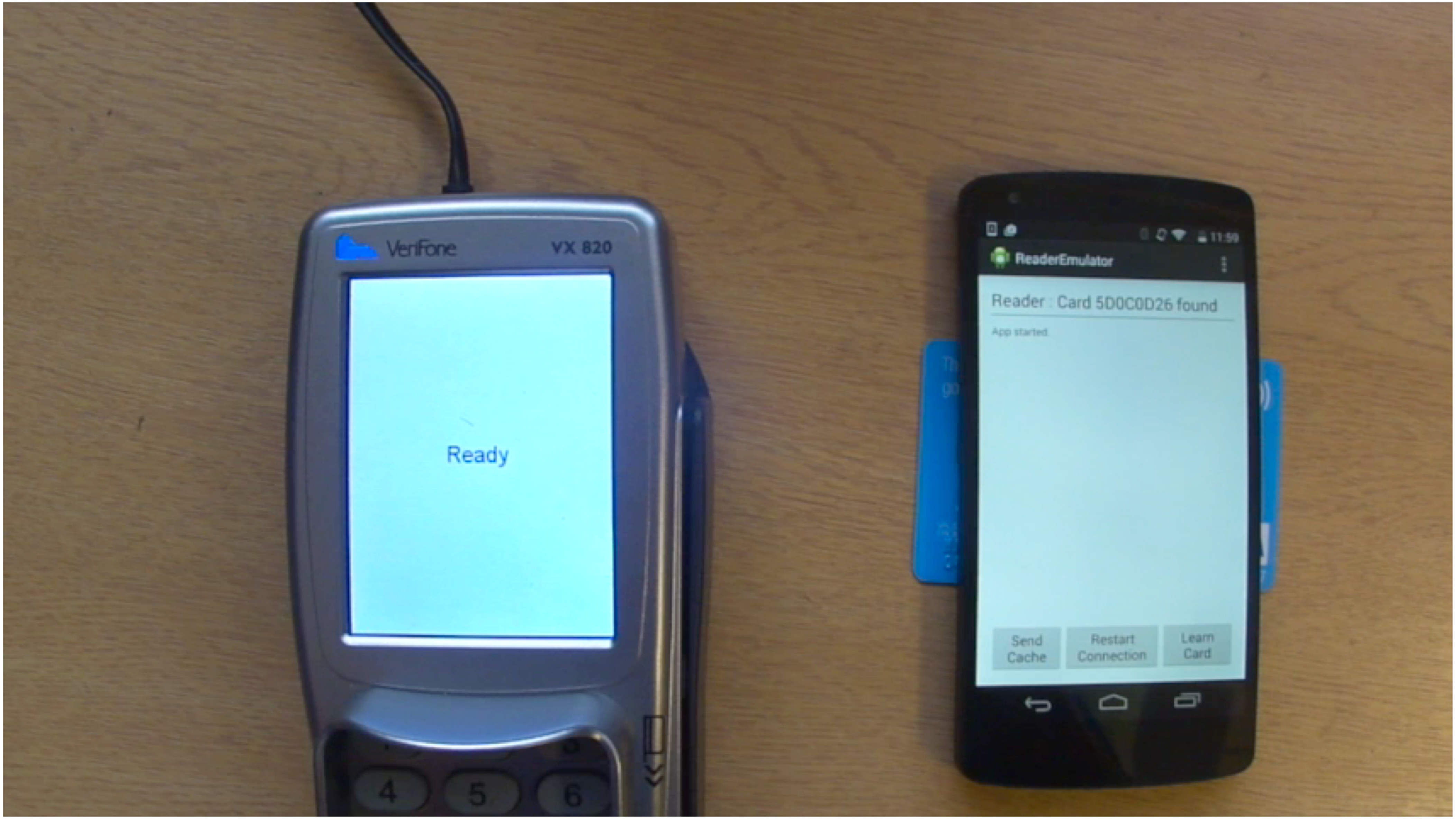
Tom Chothia, Joeri de Ruiter and Ben Smyth



Introduction

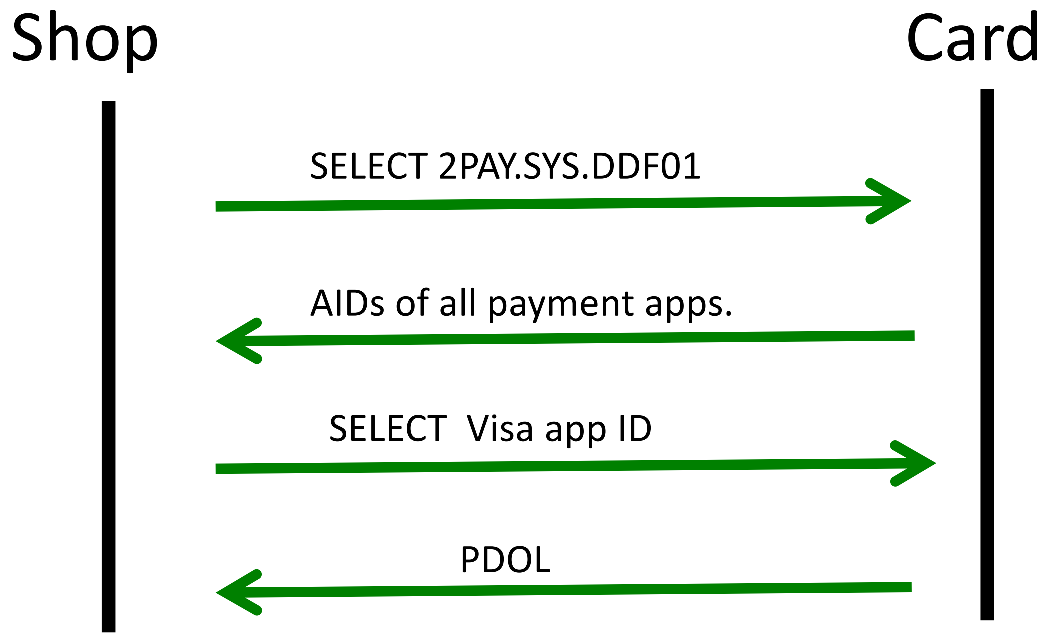
- Contactless EMV & relay attacks
- A protocol to stop relay attacks on EMV
- A extension of the applied pi-calculus to model DB protocols
- Automatically checking previously defined symbolic properties.
- A Hierarchy of DB properties.
- Examples: Contactless EMV & NXP's DB protocol.


The co-operative bank
VISA
DEBIT


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


09-25-2017 Mon 01:01:52
Camera 01
WEST MIDLANDS POLICE

Visa’s Protocols



PDOL = Processing Options Data Object List

- list of data the reader must provide to the card.

PDOL

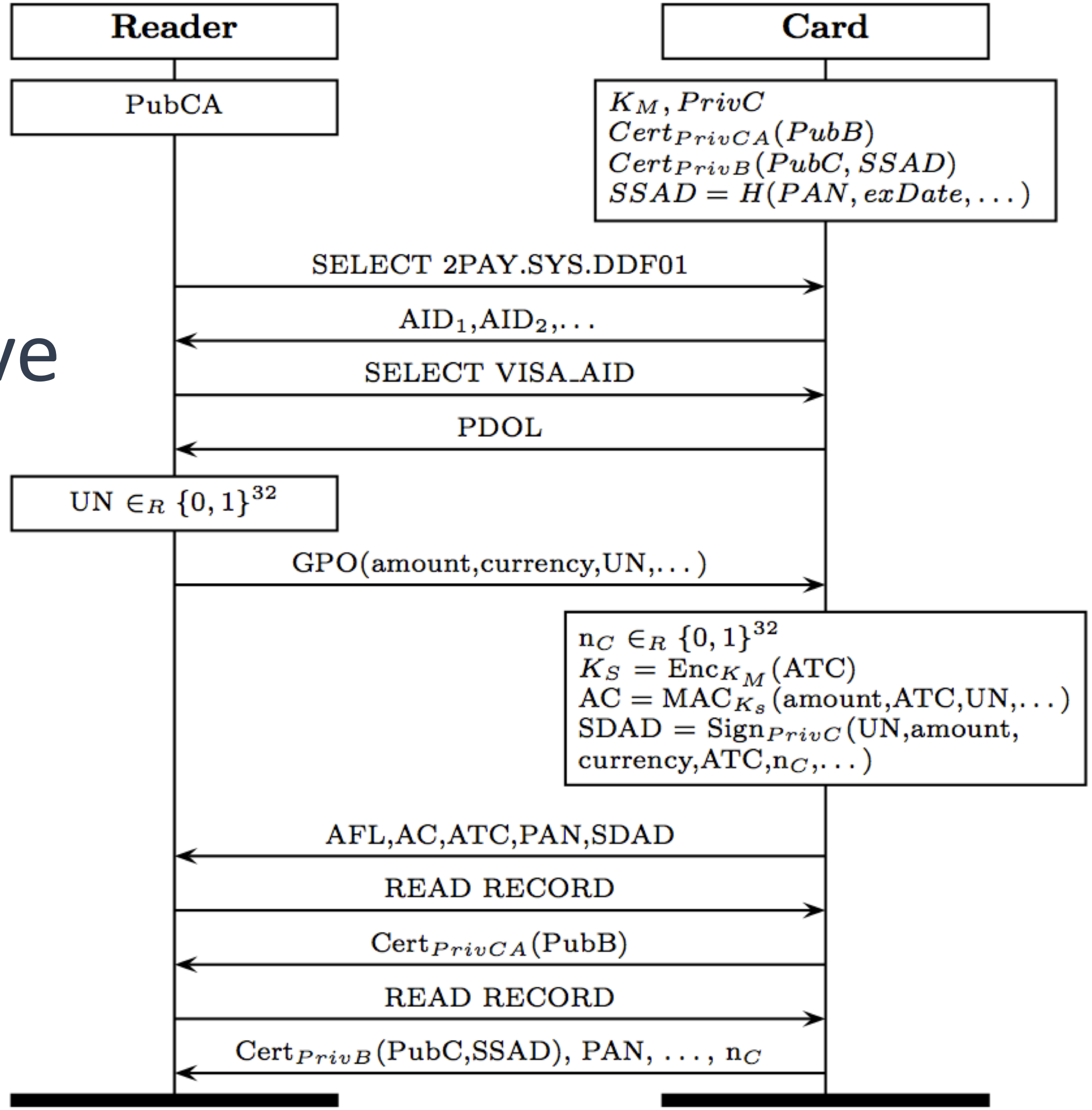
9F38189F66049F02069F03069F1A0295055F2A029A039C019F37045F2D02656E9000

which parses as:

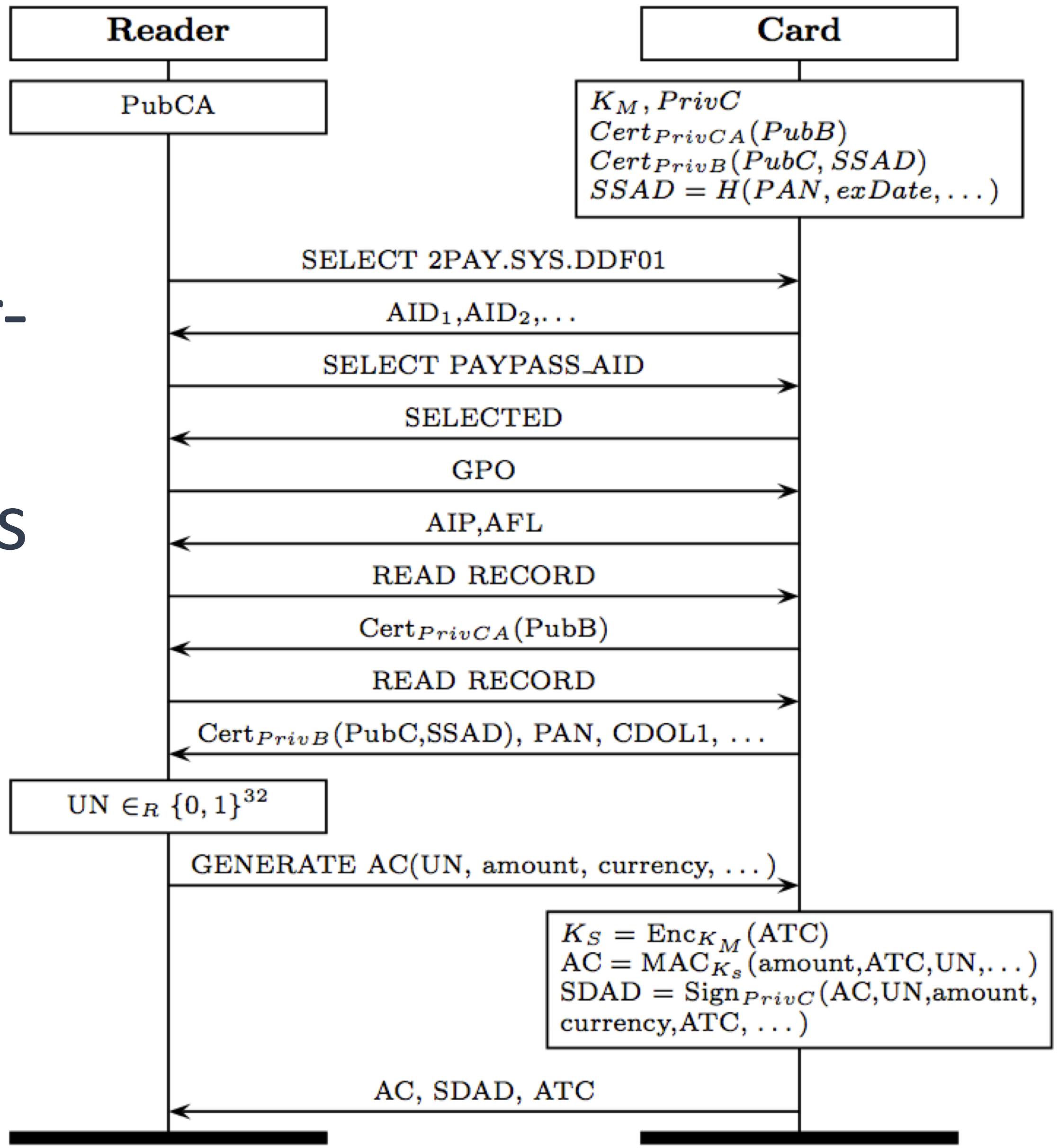
9F38 | len:18 Processing Options Data Object List (PDOL)

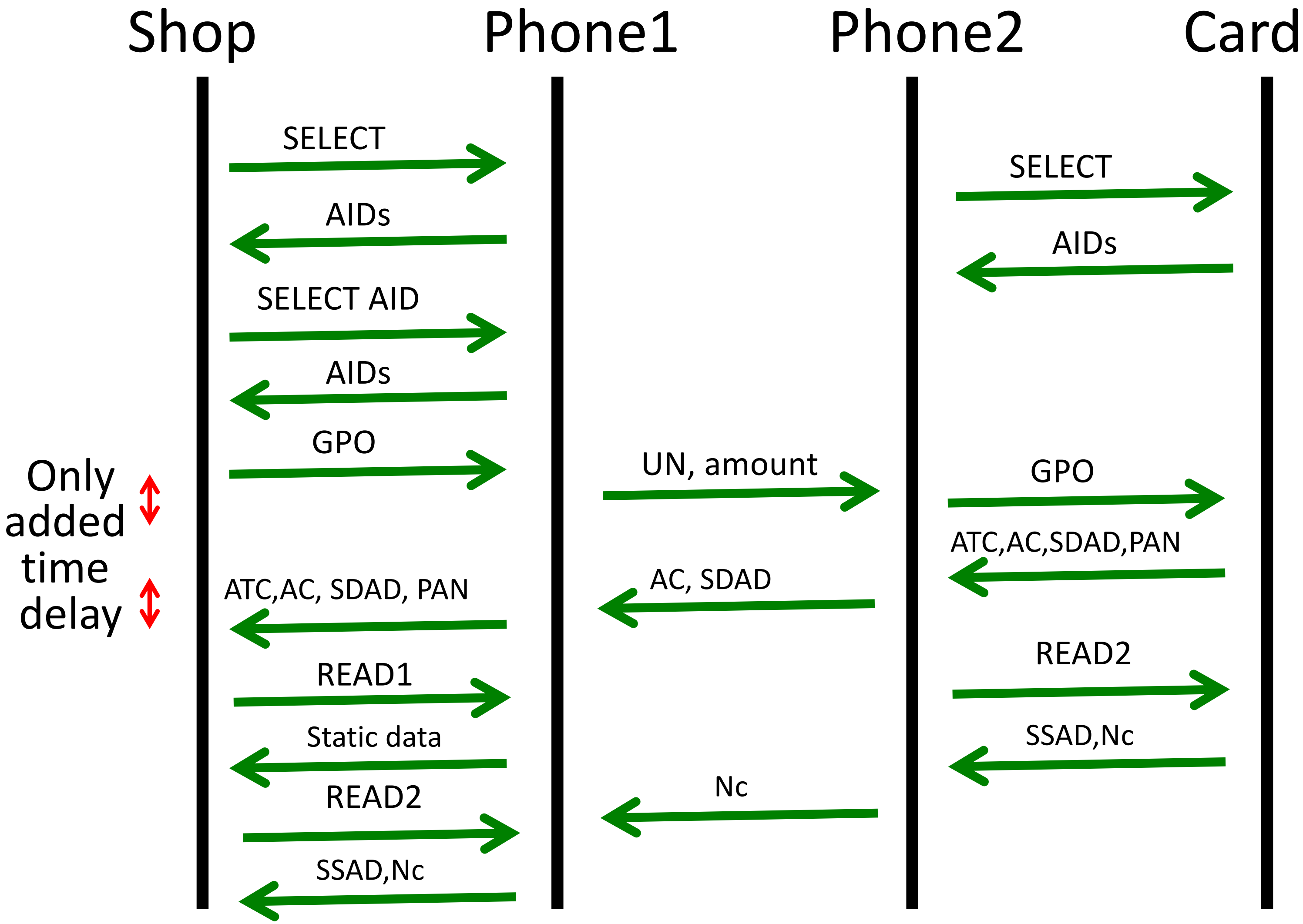
9F66	len:04	Card Production Life Cycle
9F02	len:06	Amount, Authorised (Numeric)
9F03	len:06	Amount, Other (Numeric)
9F1A	len:02	Terminal Country Code
95	len:05	Terminal Verification Results
5F2A	len:02	Transaction Currency Code
9A	len:03	Transaction Date
9C	len:01	Transaction Type
9F37	len:04	Unpredictable Number

Visa’s PayWave



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

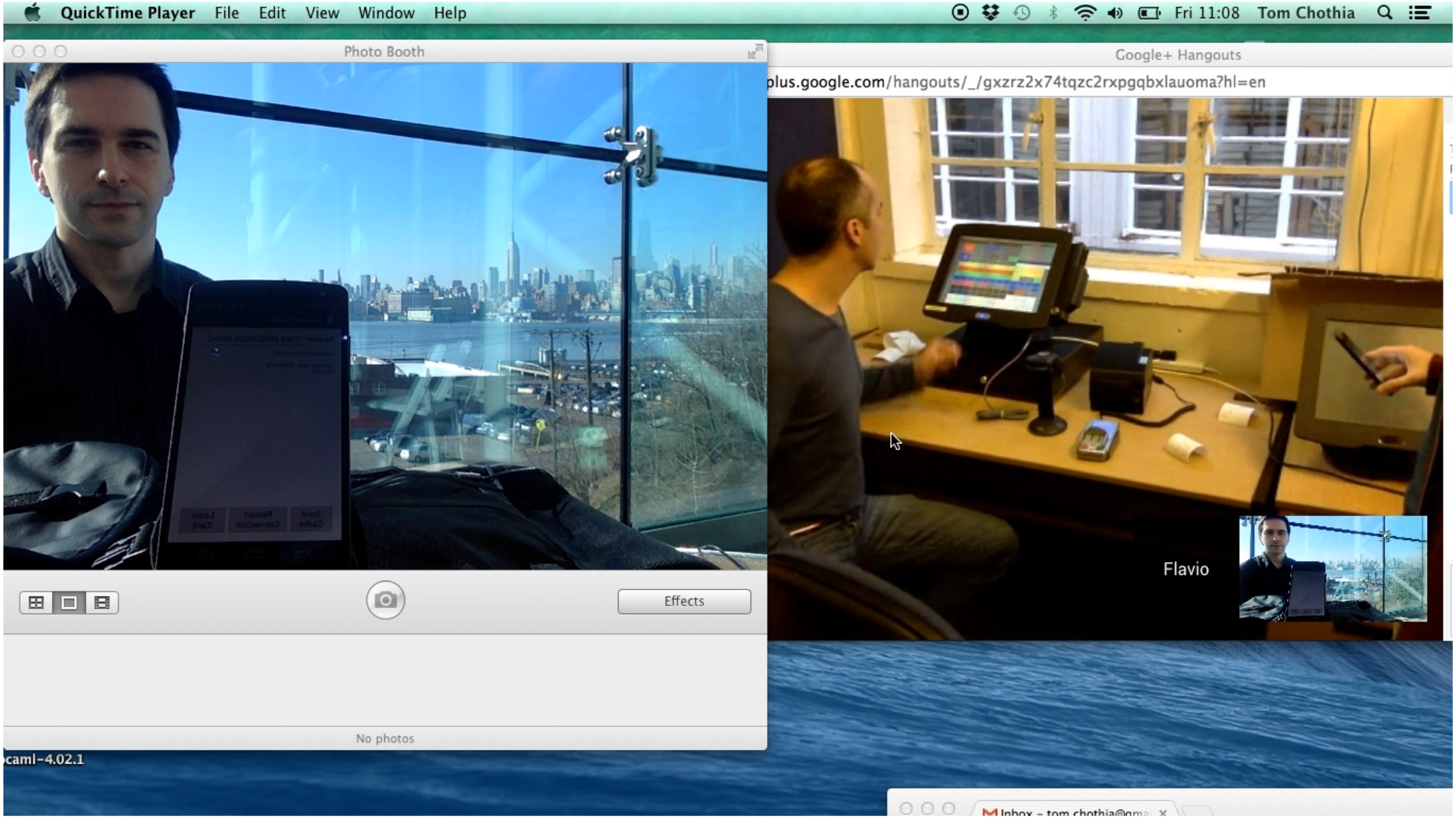
Relay timing

- We measured the exact transaction times for a number of cards.
  - Fastest 330ms
  - Slowest 637ms

- Fastest relayed transaction: 485ms

- Placement of card can have an affect > 80ms for longest messages.

- ABN Amro (Dutch)
  - Time for card to complete a purchase: 637ms
  - Time for relay to complete a purchase:627ms.


QuickTime Player File Edit View Window Help
Fri 11:08 Tom Chothia
Photo Booth
Effects
No photos
ocaml-4.02.1
Google+ Hangouts
plus.google.com/hangouts/_/gxzrz2x74tqzc2rxpgqbxlauoma?hl=en
Flavio

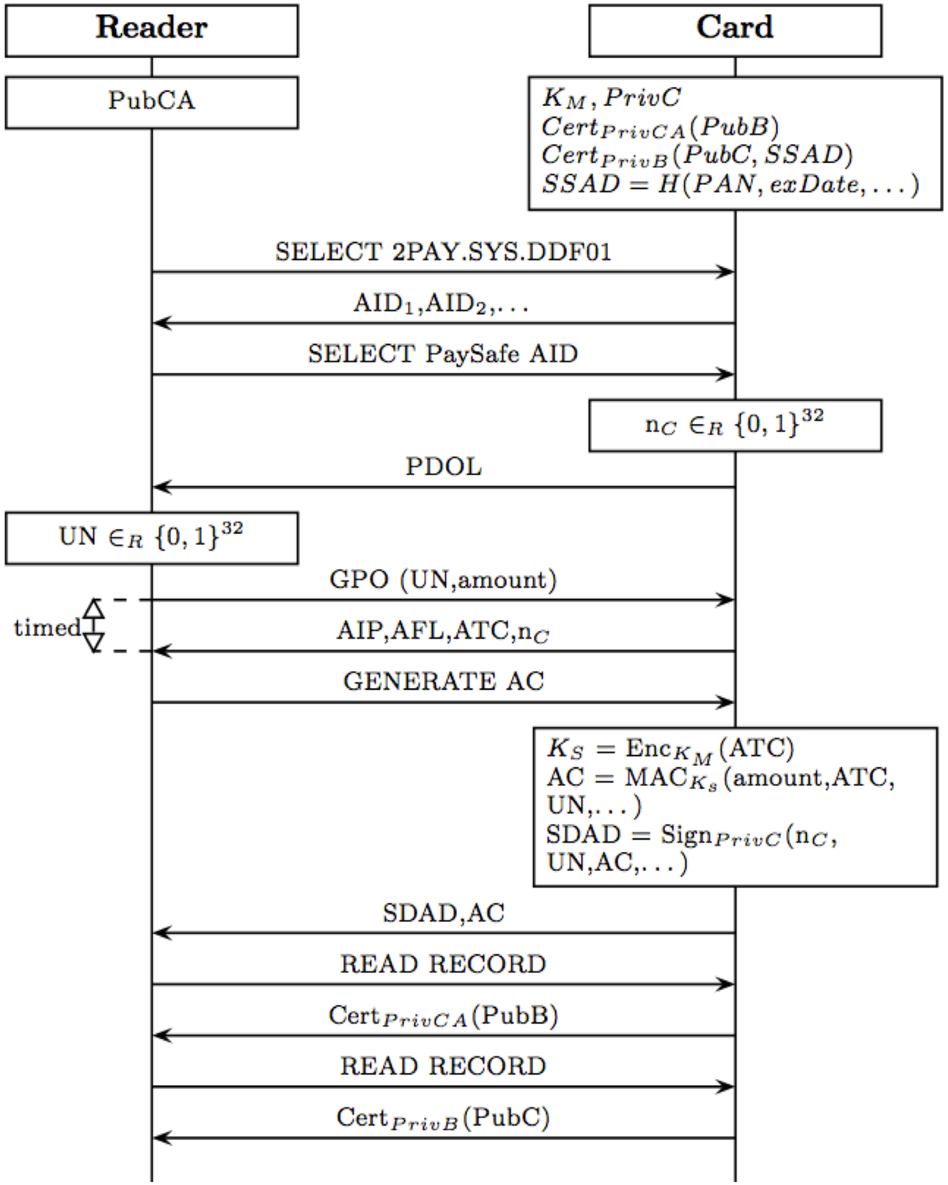
Attacker model

- We only want to stop someone using a relay to steal money or a car. Nothing more, nothing less.

- We assume the relay adds a significant delay.

- We work in the symbolic model:
  - Idealized crypto
  - Message integrity
  - No side channels

PaySafe



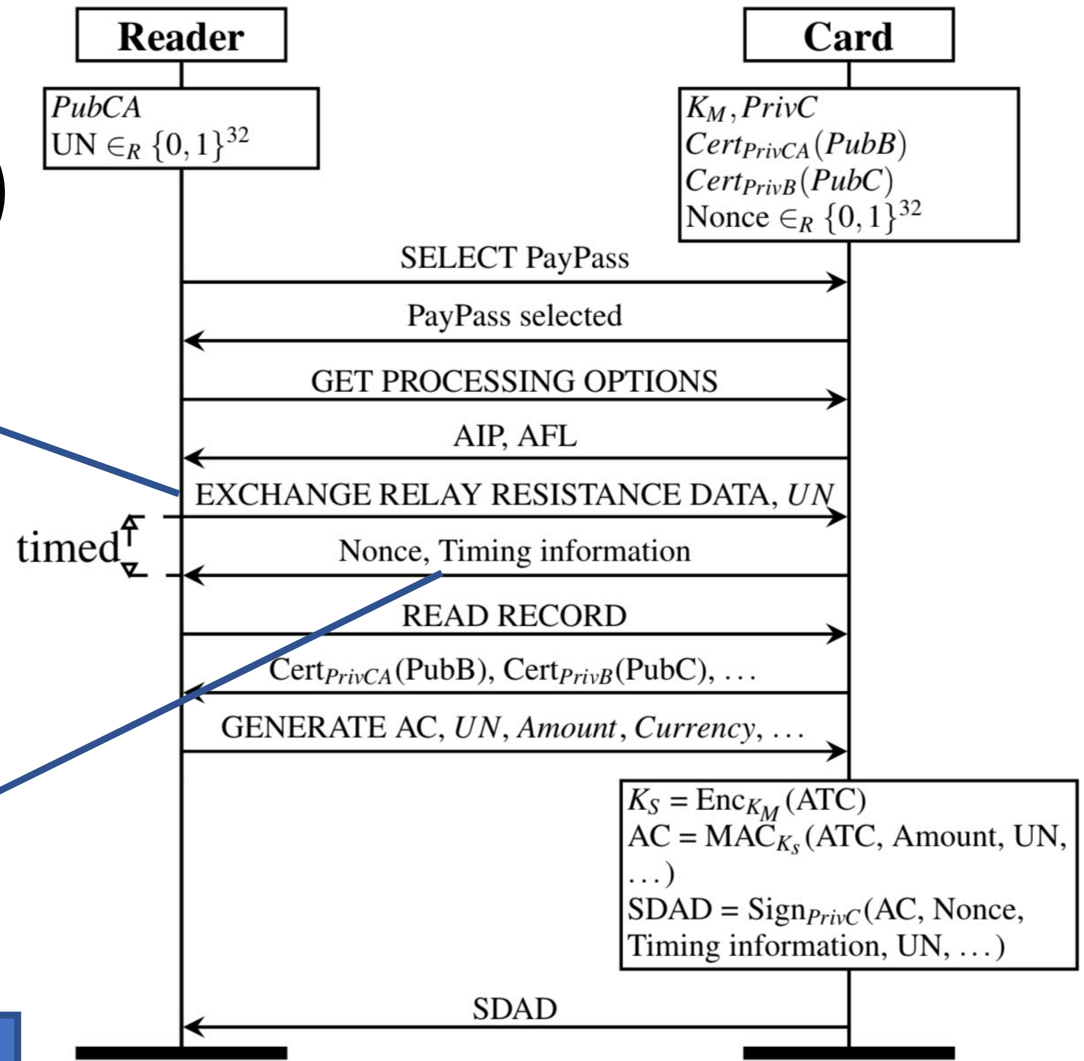
Reader
Card
PubCA
$K_M, PrivC$
$Cert_{PrivCA}(PubB)$
$Cert_{PrivB}(PubC, SSAD)$
$SSAD = H(PAN, exDate, \ldots)$
SELECT 2PAY.SYS.DDF01
$\mathrm{AID}_1,\mathrm{AID}_2,\ldots$
SELECT PaySafe AID
$\mathrm{n}_C \in_R \{0,1\}^{32}$
PDOL
$\mathrm{UN} \in_R \{0,1\}^{32}$
GPO (UN,amount)
timed
$\mathrm{AIP,AFL,ATC,n}_C$
GENERATE AC
$K_S = \mathrm{Enc}_{K_M}(\mathrm{ATC})$
$\mathrm{AC} = \mathrm{MAC}_{K_s}(\mathrm{amount,ATC,}$
$\mathrm{UN},\ldots)$
$\mathrm{SDAD} = \mathrm{Sign}_{PrivC}(\mathrm{n}_C,$
$\mathrm{UN,AC},\ldots)$
SDAD,AC
READ RECORD
$\mathrm{Cert}_{PrivCA}(\mathrm{PubB})$
READ RECORD
$\mathrm{Cert}_{PrivB}(\mathrm{PubC})$

MasterCard's Relay Resistance Protocol (RRP)

Uses New Command

Timing profile sent by card

We check this as auth. property



NXP distance bounding protocol

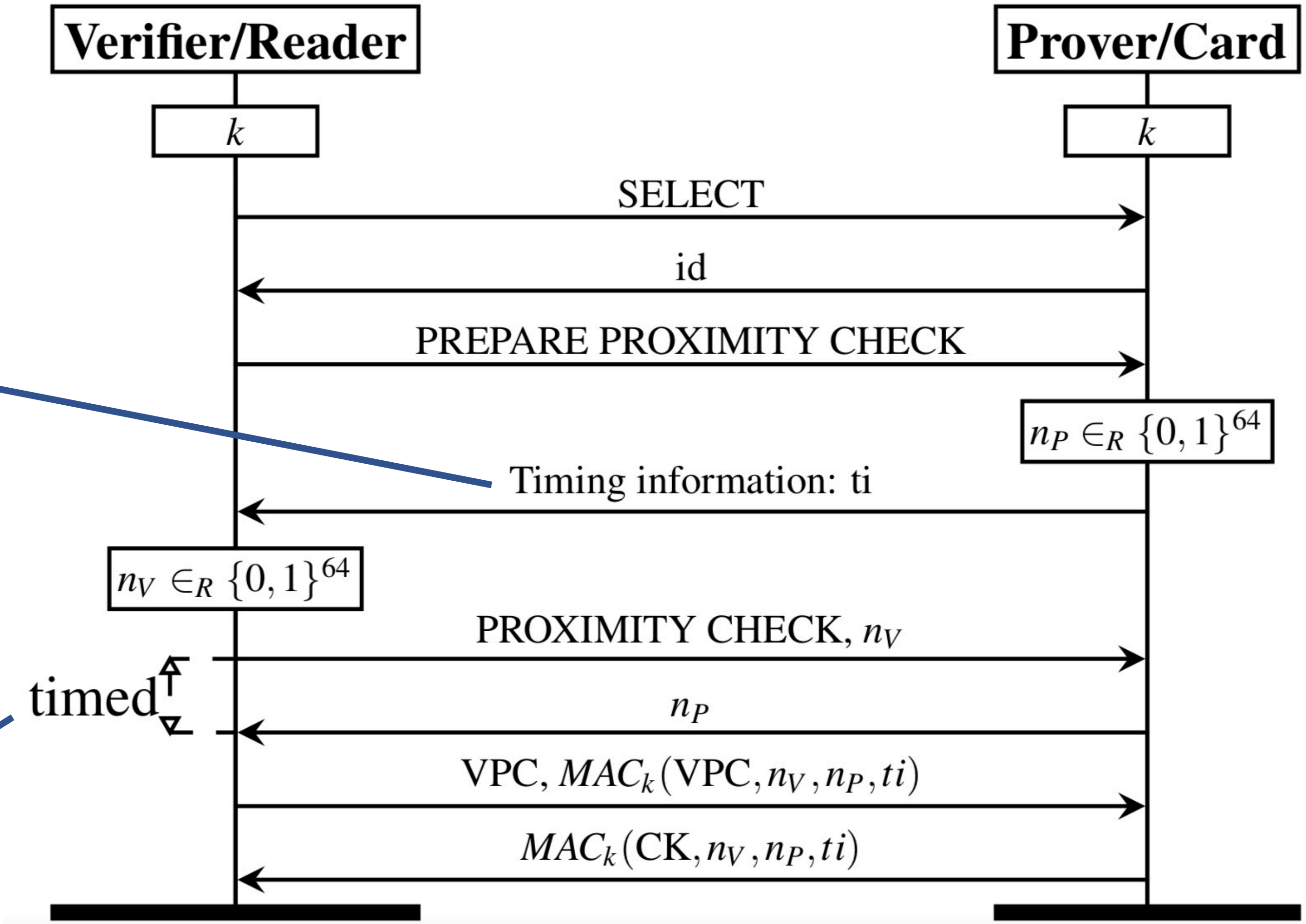
- NXP sell a distance bounding smart card.
- NXP have patented a distance bounding ☺
- Patent documents are really hard to read ☹

“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 bytes



Applied pi-calculus

`in(x).P`	input
`out<x>.P`	output
`P	Q`	two processes running in parallel
`!P`	infinite number of copies of process P
`new a.P`	a new name “a”, e.g. a nonce, session key
`let x=D in P else Q`	pattern matched e.g. encryption
`event(P)`	event used for testing
`t:P`	numbered phase jump, enforces order

Example

1. A -> B : na
2. B -> A : enc((na,nb), kab)
3. A -> B : nb

A and B correctly authenticate each other if for every:

- event(endA(na,nb)) there is a unique event(B_using(na,nb))
- event(endB(na,nb)) there is a unique event(A_using(na,nb))

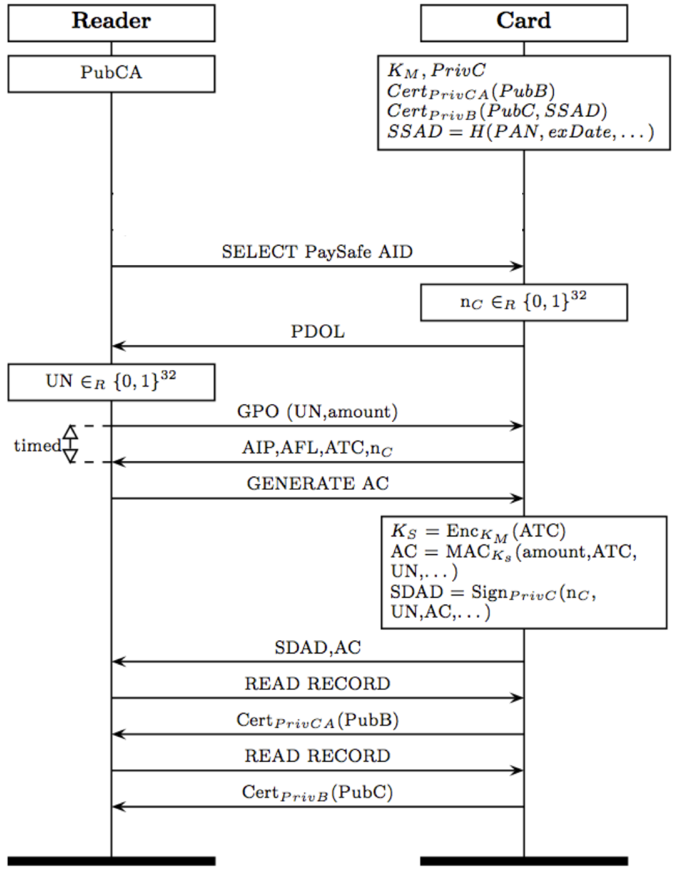
```
A = new na. out c<na>.
    in c(x).let (=na,nb)=dec(x).
    event(A_using(na,nb)).
    out(nb).
    event(endA(na,nb))

B = in(n). new nb.
    event(B_using(na,nb)).
    out c<enc((nb,na),kab)>.
    in(y).let (=nb) = y.
    event(endB(na,nb))

System = new kab.(!A | !B)
```

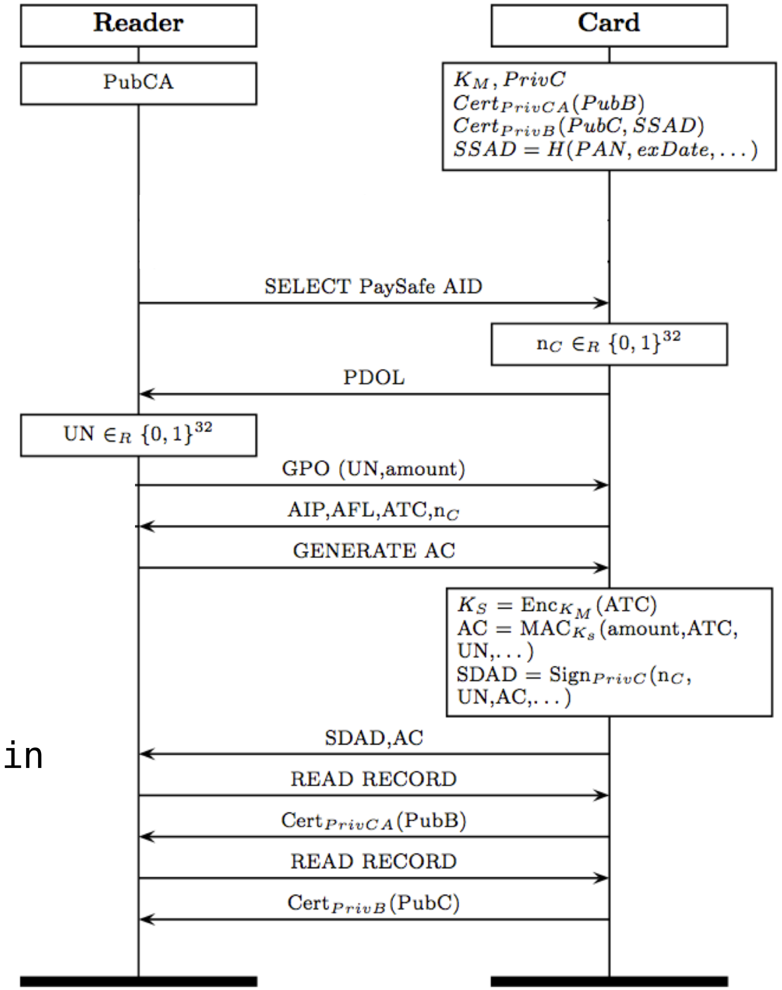
PaySafe Model

```
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

```
let Prover =
  in c(=SELECT,=AID).
  new NC. new ATC.
  out c<PDOL>.
  in c(=GET_PROCESSING_OPTIONS,UN,amount).
  out c<AIP,AFL,cNC>.
  in c(=GENERATE_AC, amount).
  let macKey = genKey(cATC, sharedKey(idP)) in
  let AC = mac((cAmount,cUN,cATC), macSessionKey) in
  let SDAD = sign(UN,NC,amount,ATC,AC),getPrivKey(idP)) in
  out c<cSDAD,AC>.
  in c(=READ_RECORD).
  out c<cCardCert>.
```



Extended Applied pi-calculus for DB

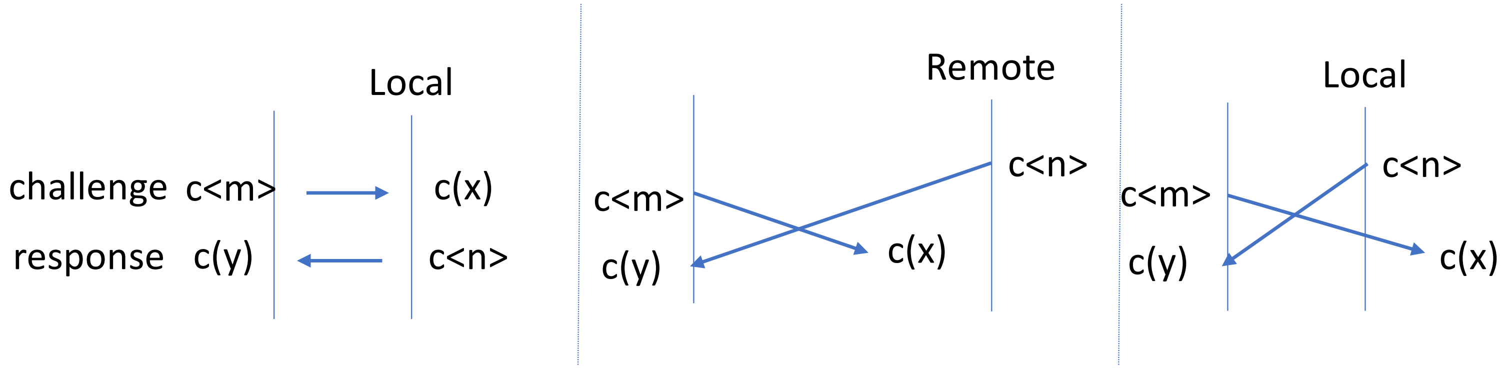
```
in(x).P
out<x>.P
P | Q
!P
new a.P
let x = D in P else
event(P)
```

```
startTimer.P
stopTimer.P
```

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

Eg.

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



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

$$[\ \mathtt{in}\ c(x).P \mid \mathtt{out}\ c<n>.Q\ ]_r \rightarrow [\ P\{n/x\} \mid Q\ ]_r$$

$$[\mathtt{out}\ c<n>.Q]_r \mid [P]_0 \rightarrow [Q]_r \mid [\mathtt{out}\ c<n> \mid P]_0$$

$$[\mathtt{out}\ c<n>.Q\ ]_r \rightarrow [\ \mathtt{out}\ c<n> \mid Q\ ]_r$$

PaySafe Model

```
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 ]
```

Defining DB Protocols

To define a DB protocol we (P(id),V,n)

- Provers P(id) is of the form `P(id)=!new id.new n.let .. !Q`
- Verifier V is of the form `V=!new n.!V'`, and can perform `event verified(id)`.

We write**: verified(id):S** to mean the verifier accepts a run from prover "id"

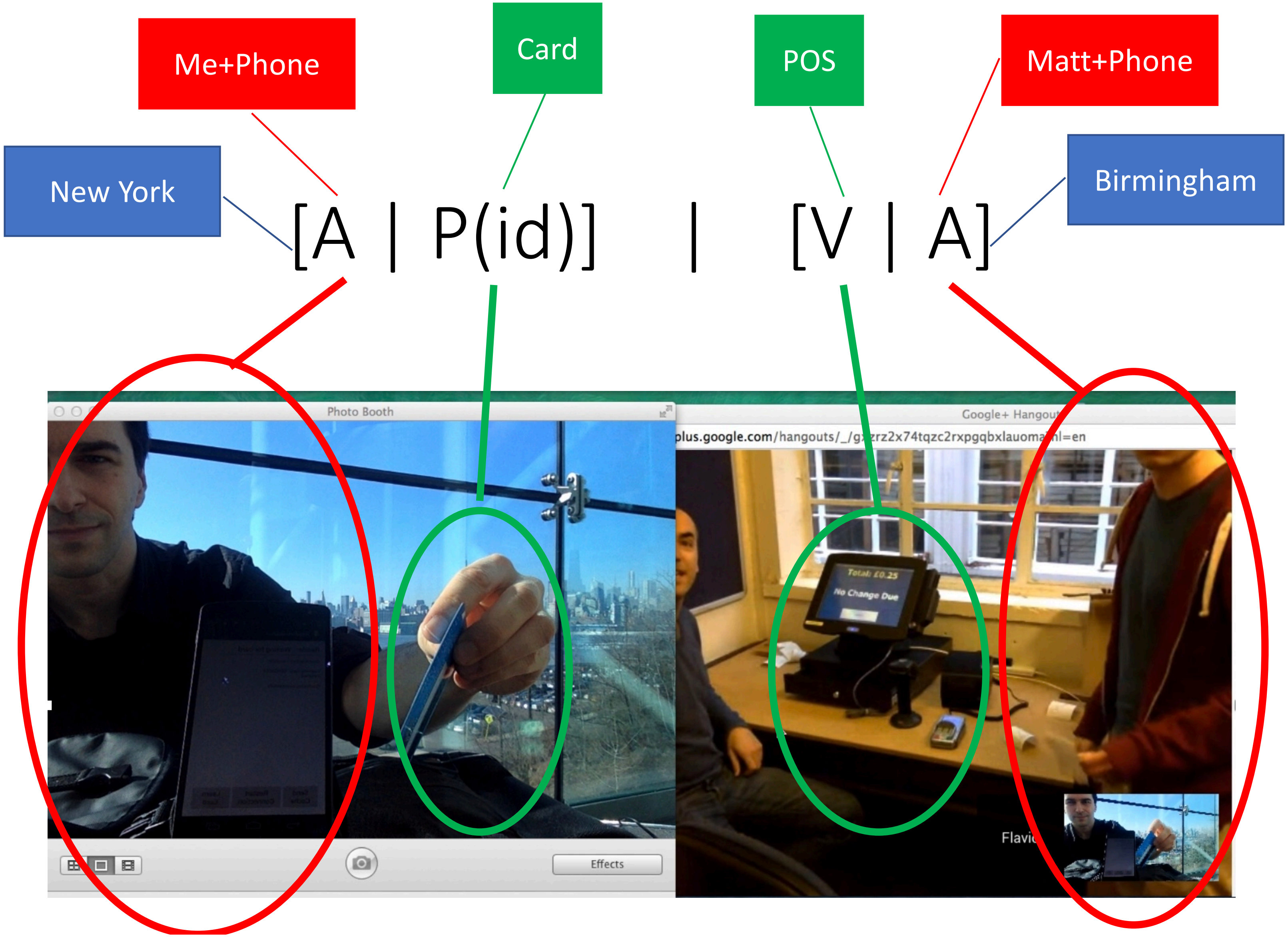
```
S ->* [new id.P|X]|L -> [P{a/id}|X] ->*
                        [event(verified(a)|Y] | R
```

We write " `[ V(id)|…] | …` " for "`verified(id):[V|…] | …` "

- E.g. `[V(id)|P(id')] | [P(id)|A]`

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 attacker and honest prover



Me+Phone
Card
POS
Matt+Phone
New York
Birmingham
[A | P(id)] | [V | A]
Photo Booth
Effects
Total: £0.25
No Change Due

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 Distance Fraud**: A dishonest prover remotely authenticates to a verifier.

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

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

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

Terrorist Frauds

- Terrorist Fraud, TP-A(id): = A | oracle for all functions and values
- **Terrorist Fraud**: A remote dishonest prover* and local attacker

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

- **Assisted Terrorist Fraud**: remote dishonest prover* and local attacker and honest prover

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

- **Assisted Distance Fraud**: remote dishonest prover* and local attacker and honest prover

  [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)]

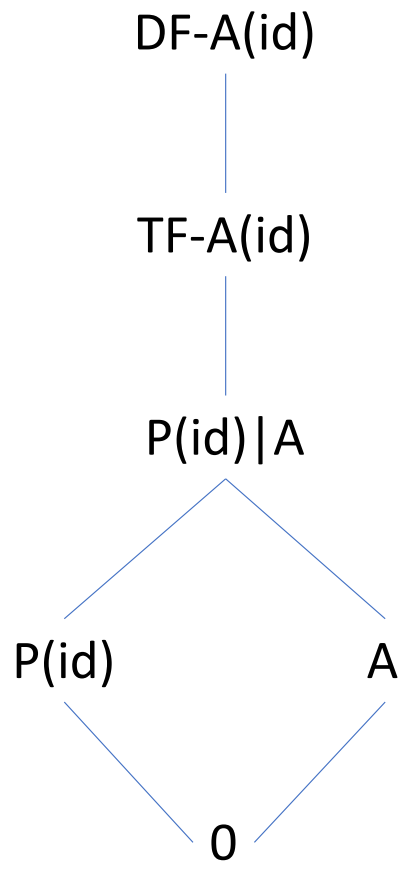
Distance Fraud
[V(id)] | [DP-A(id)]

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

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

Props

- 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.



Some equalities between processes

When testing for “id”

- P(id) is more powerful that P(id’)
- TF-A(id) is more powerful that TF-A(id’)
- DP-A-A(id) is more powerful that DP-A-A(id’)

[V(id)|A] | [P(id)|A]
= [V(id)|A] | [P(id)|A | P(id’)]

P(id) = P(id)|P(id’)
TP-A(id) = TP-A(id) | P(id’)
TP-A(id) = TP-A(id) | TP-A(id’)
DP-A-A(id) = DP-A-A(id) | P(id’)
DP-A-A(id) = DP-A-A(id) | TP-A(id’)
DP-A-A(id) = DP-A-A(id) | DP-A-A(id’)

TP-A(id) = TP-A(id) | A
DP-A(id) = DP-A(id) | TP-A(id)
DP-A(id) = DP-A(id) | A

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

We have that:
TP-A(id) = TP-A(id) | A
P(id)|A < TP-A(id)
TP-A(id) = TP-A(id) | TP-A(id’)

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

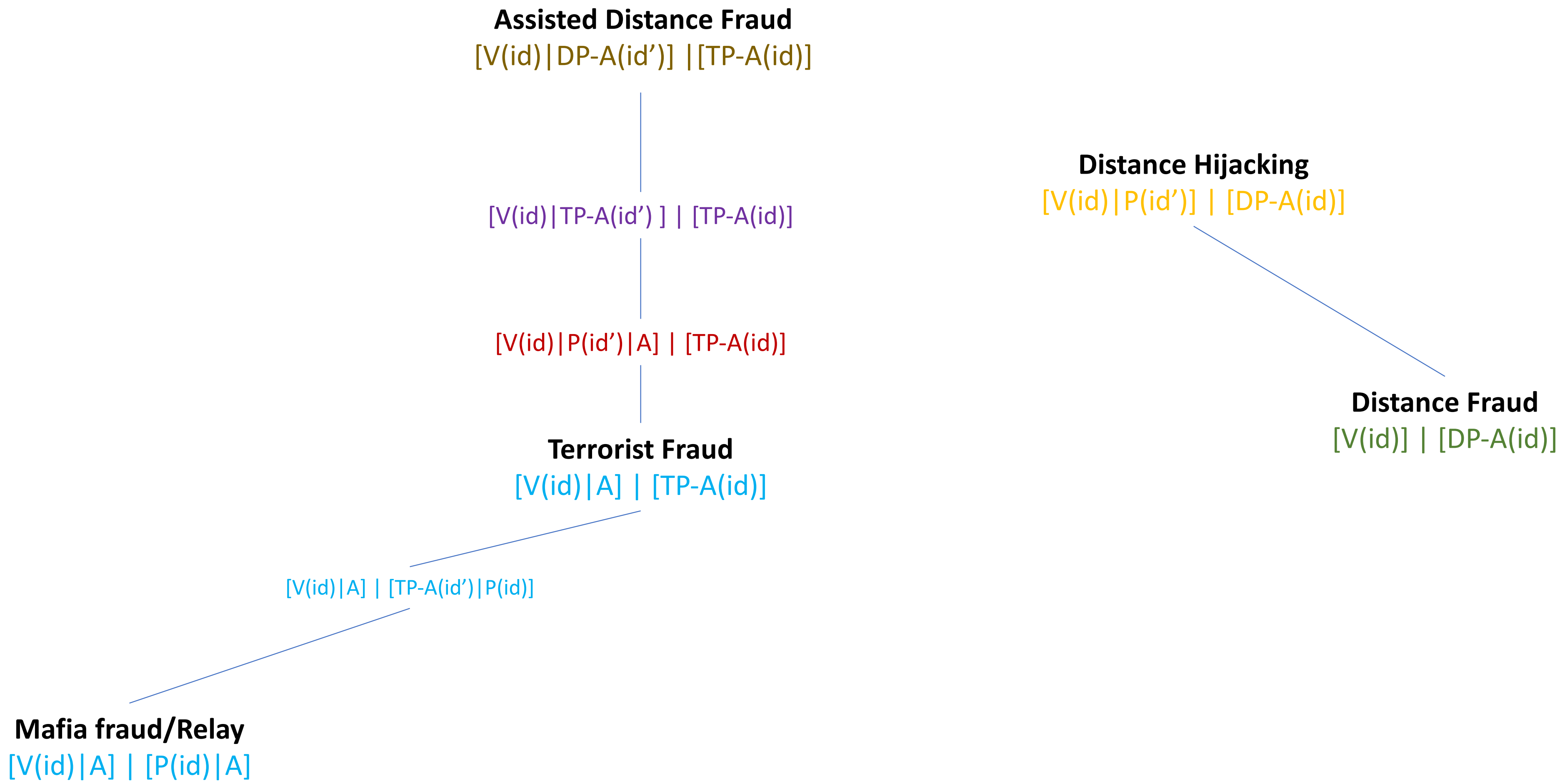
Distance Fraud
[V(id)] | [DP-A(id)]

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

[V(id)|A] | [TP-A(id’)|P(id)]

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

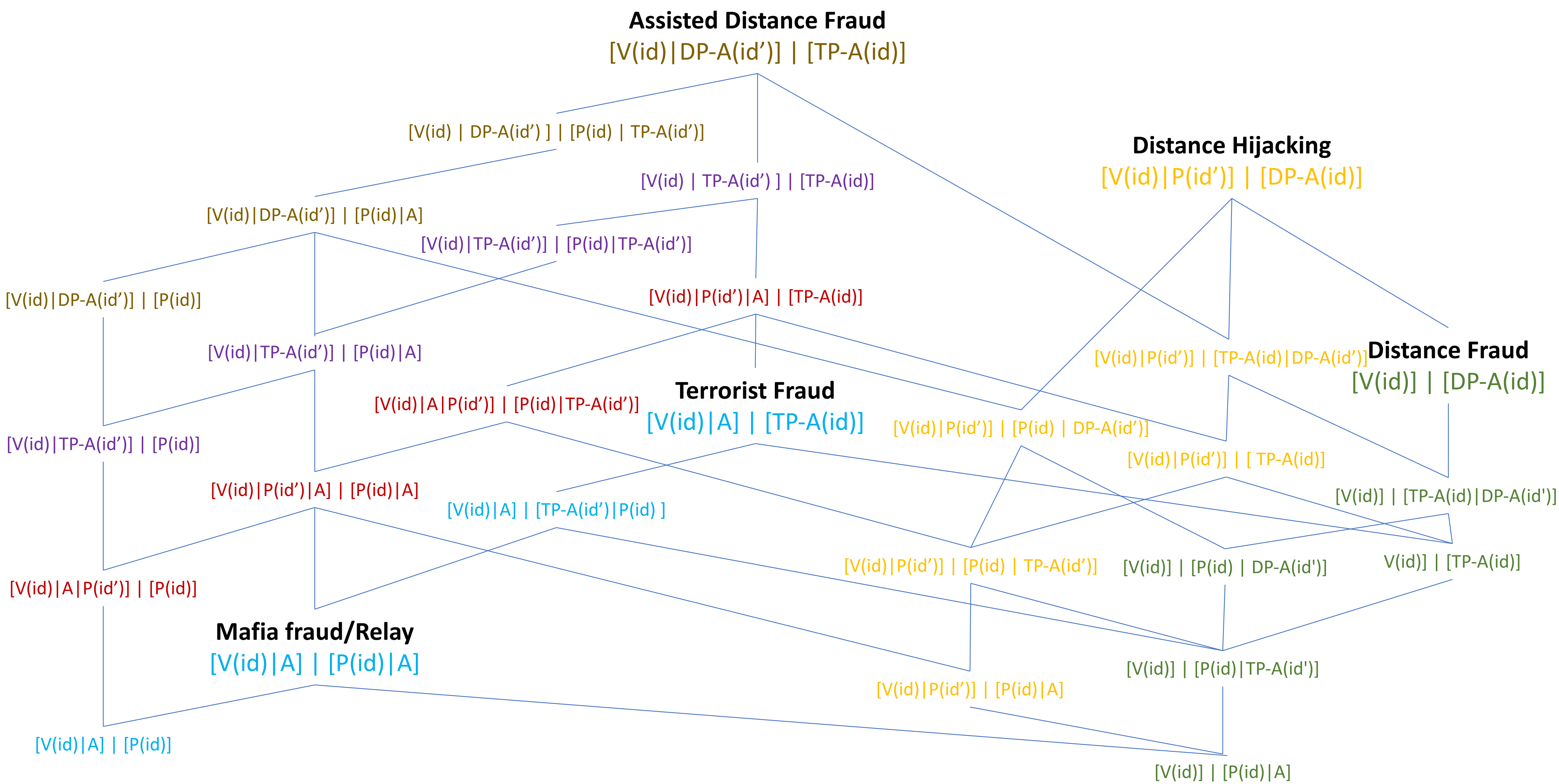
[V(id)|A] | [TP-A(id’)|P(id)]
= [V(id)|A] | [TP-A(id’)|P(id)|A]
< [V(id)|A] | [TP-A(id’)|TP-A(id)]
= [V(id)|A] | [TP-A(id’)]



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

Other Properties

- All possible combinations of our processes give us 16,384 scenarios.
- We disregard scenarios in which the prover and verifier are co-located and there is a prover: 1,792 scenarios
- (For now) only one of P(id),P(id)|A,TP-A(id),DP-A(id): 512 scenarios.
- Apply our inequalities: 72 scenarios.
- Apply our equalities: 28 scenarios.



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]

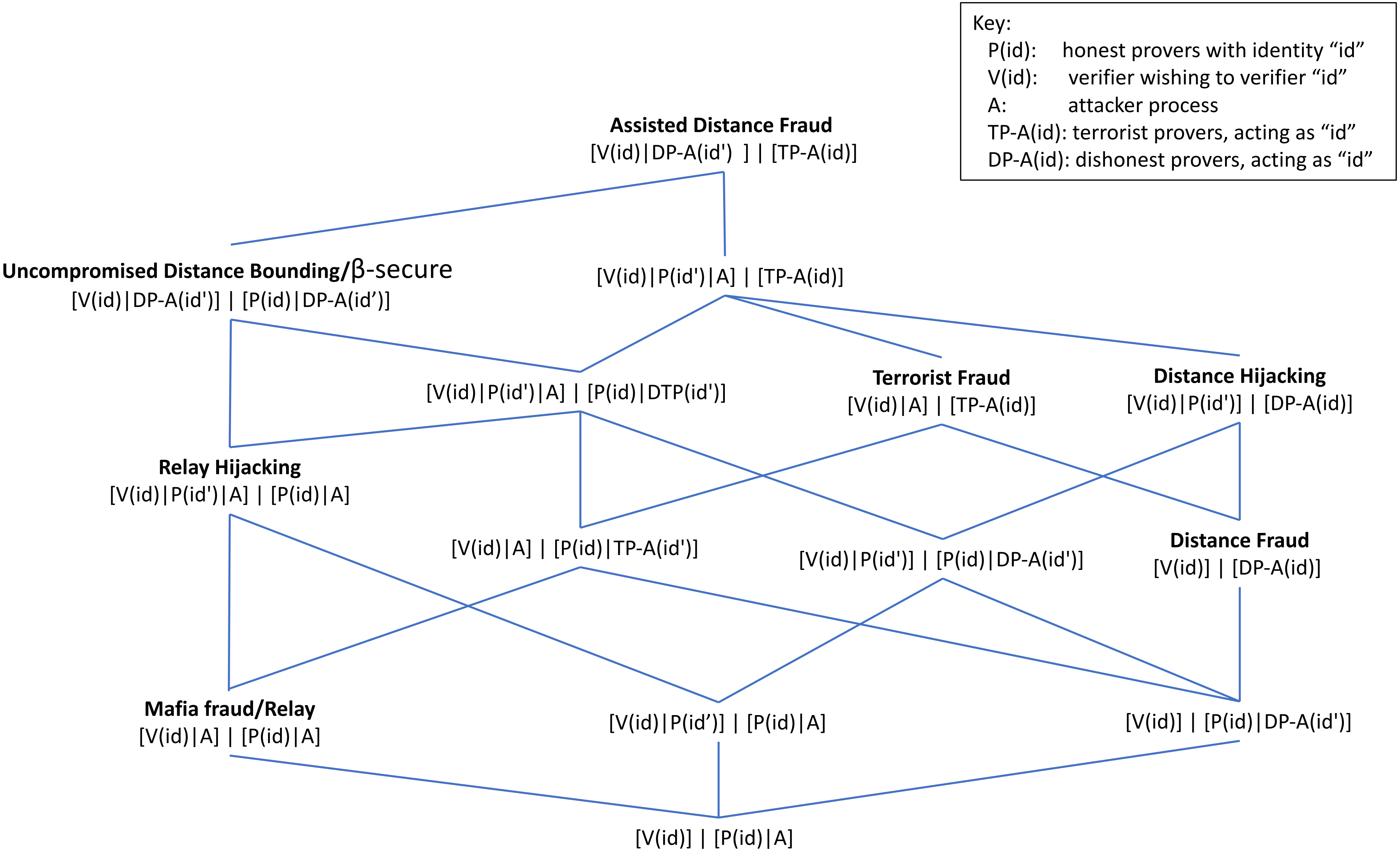
Some Heuristics

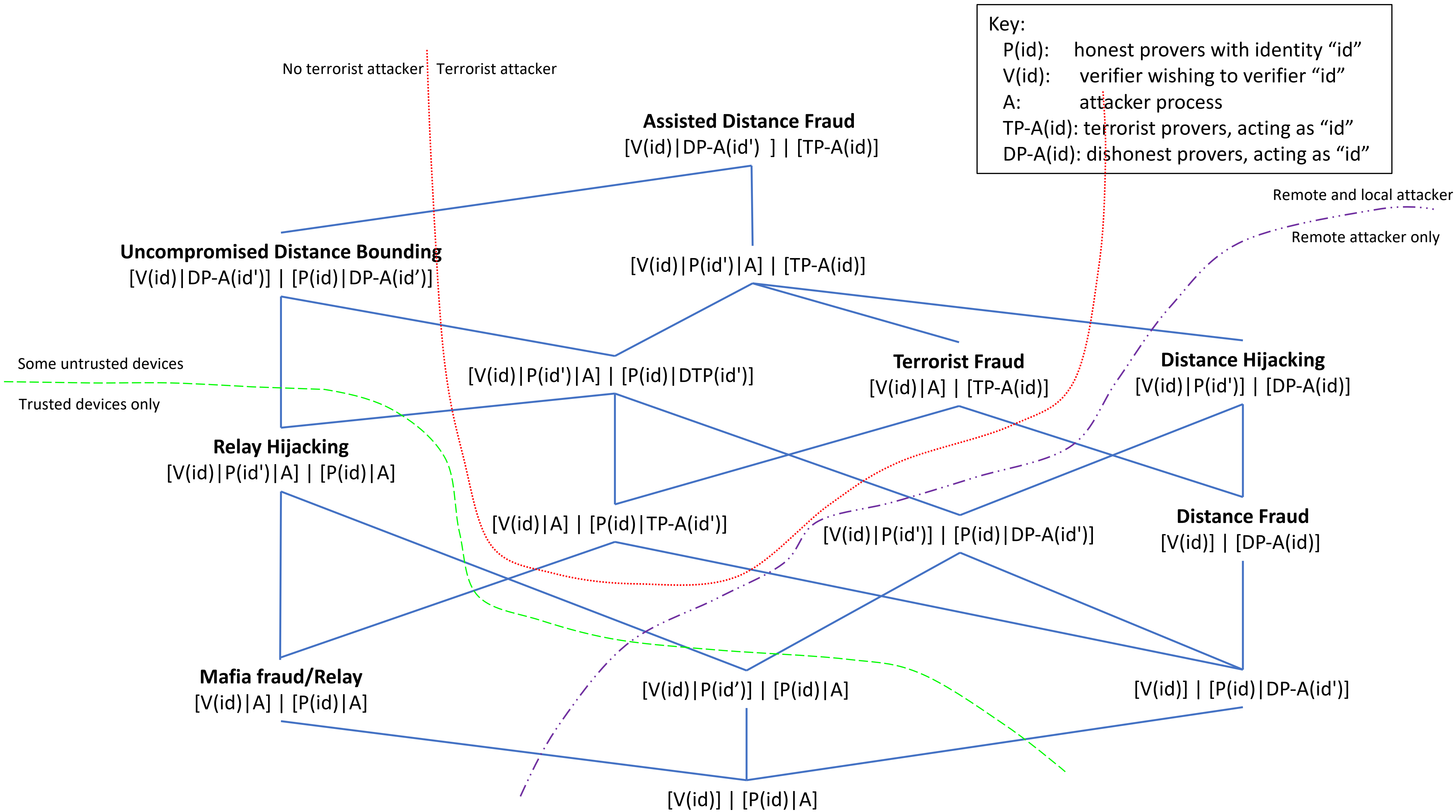
If the prover doesn’t time bound the verifier, and remote communication is possible (or uninteresting to us):

[V(id) | X | A] | [Y]   vs.   [V(id) | X | A] | [Y | A]

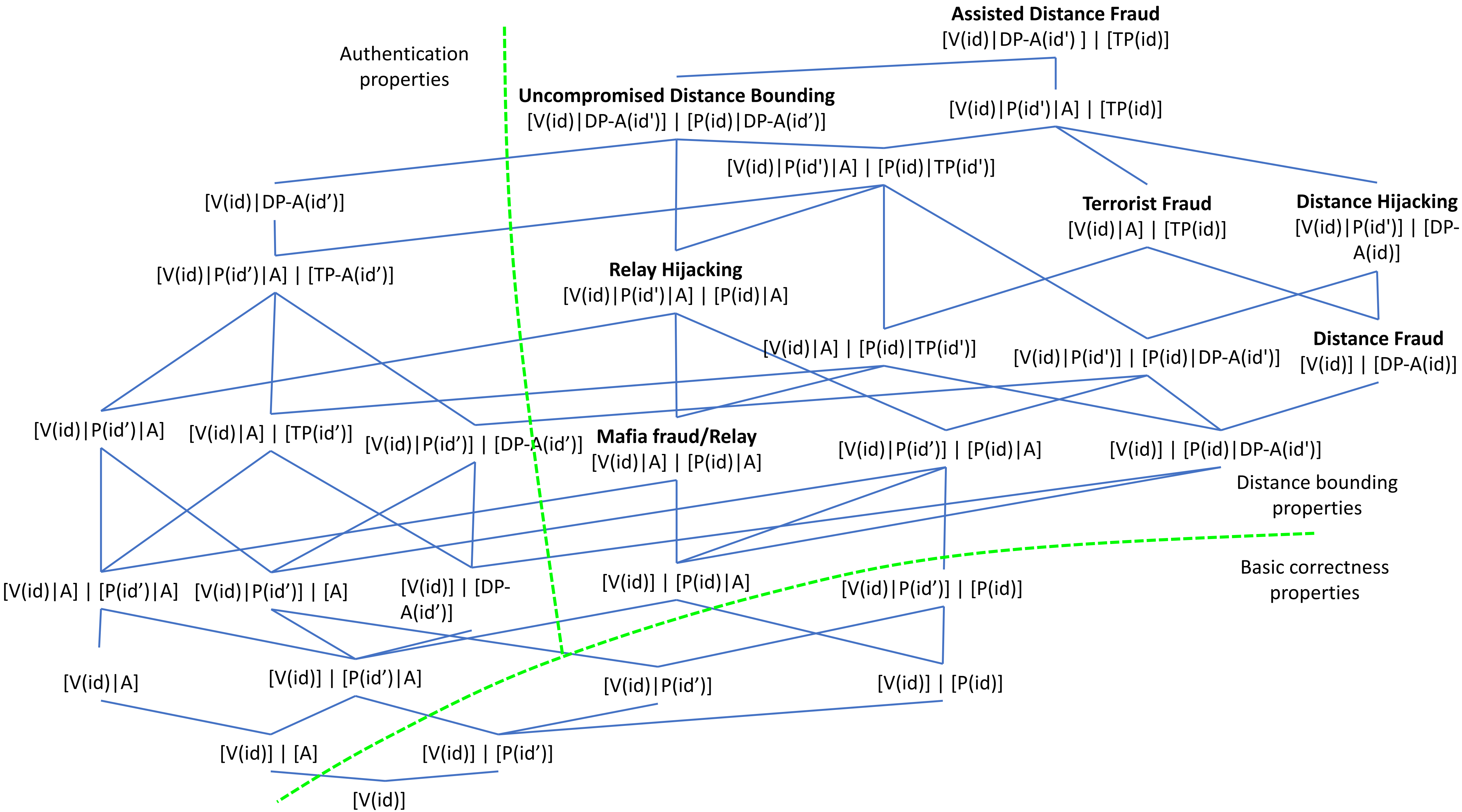
With no local attacker TP-A & DP-A will normally have the same power:

[V(id) | X] | [Y | TP-A(x)] vs [V(id) | X] | [Y | DP-A(x)]


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/β-secure
[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]



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
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]



Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP(id)]
Authentication properties
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id')]
[V(id)|P(id')|A] | [TP(id)]
[V(id)|P(id')|A] | [P(id)|TP(id')]
[V(id)|DP-A(id')]
Terrorist Fraud
[V(id)|A] | [TP(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
[V(id)|P(id')|A] | [TP-A(id')]
Relay Hijacking
[V(id)|P(id')|A] | [P(id)|A]
[V(id)|A] | [P(id)|TP(id')]
[V(id)|P(id')] | [P(id)|DP-A(id')]
Distance Fraud
[V(id)] | [DP-A(id)]
[V(id)|P(id')|A]
[V(id)|A] | [TP(id')]
[V(id)|P(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')]
Distance bounding properties
[V(id)|A] | [P(id')|A]
[V(id)|P(id')] | [A]
[V(id)] | [DP-A(id')]
[V(id)] | [P(id)|A]
[V(id)|P(id')] | [P(id)]
Basic correctness properties
[V(id)|A]
[V(id)] | [P(id')|A]
[V(id)|P(id')]
[V(id)] | [P(id)]
[V(id)] | [A]
[V(id)] | [P(id')]
[V(id)]

Checking Mobility

- What about a prover than is in range and then moves away?

  We can check this as a injective correspondence

E.g. **Relay Hijacking:** [V(id)|P*(id)|A] | [P(id)|A]

where P*(id) as P(x), but starts with `event(start(id))`.

- We then check if:

  `event(verified(id)) =>` a unique `event(start(id))`.

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.

Papers

Financial Crypto 2015:

- PaySafe, idea and example of checking in the applied pi-calculus, automated checked of relay attacks.

- Current draft paper:
  - DB extensions, Hierarchy, MasterCard & NXP protocols. Automatic checking of all attacks.

Conclusion

- We build a model of distance bounding in which we abstract away from exact times:
  - All local communication and operations can be performed within the time bound
  - All communication with remote locations will take longer then the time bound.
- We enumerate and order possible symbolic DB properties
  - And link them to particular attacker models.
- Examples from MasterCard and NXP
- Links to computational model?