Case Study: EMV and Distance Bounding

Tom Chothia

Introduction

- This lecture looks at the design of EMV card security
- Relay attacks are a security issue for payment cards.
- How to check distance attack in ProVerif.
- After the break: more case studies.

Mag Strip Encoding

- Track 1

%B<*Primary account number (PAN)*>**^**<*Name*>**^**<*Expiration date (YYMM)*><*Service code*><*Discretionary data*>**?**<*Check char*>

- Track 2 (different data encoding)

;<*Primary account number (PAN)*>**=**<*Expiration date (YYMM)*><*Service code*><*Discretionary data*>**?**<*Check char*>

Service code

First digit
- 1: International use
- 2: International use, use chip.
- 5: National use,
- 6: National use, use chip

Second digit
- 0: Normal
- 2: Contact issuer via online means
- 4: Contact issuer via online means except under bilateral agreement

Third digit
- 0: PIN required
- 1: No restrictions
- 2: Goods and services only (no cash)
- 3: ATM only, PIN required
- 4: Cash only
- 5: Goods and services only (no cash), PIN required
- 6: No restrictions, use PIN where feasible
- 7: Goods and services only (no cash), use PIN where feasible

VISA PIN Verification Value

- Some VISA cards stored the encrypted PIN in the card discretionary data:

$$PVV = 3\text{-}DES_k(PAN,PIN)$$

PVV stored on magstripe, "k" a secret key known to bank and ATMs.

Your thoughts?

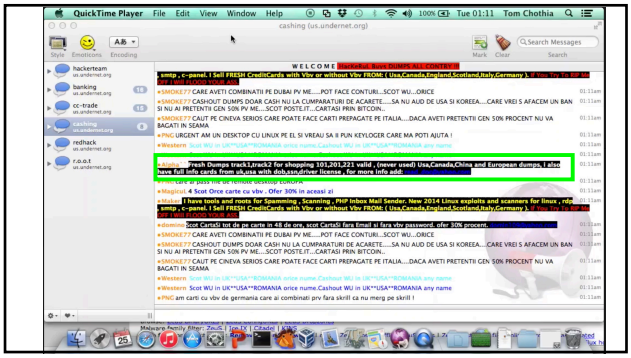
VISA PIN Verification Value

- Some VISA cards stored the encrypted PIN in the card discretionary data:

$$PVV = 3\text{-}DES_k(PAN,PIN)$$

PVV stored on magstripe, "k" a secret key known to bank and ATMs.

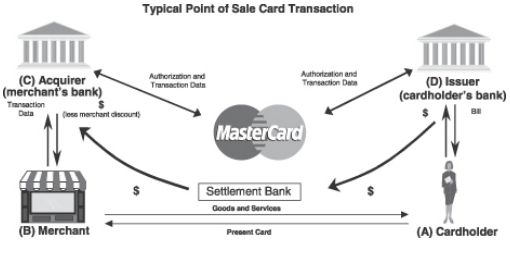
Protecting "k" is to difficult and it's compromise too dangerous.
This scheme is no longer used. Online PIN verification only.



Chip and PIN

The Specification (over 1600 pages)

- Reader has a CAs public key.
- Card has:
  - Symmetric key shared with bank
  - Certificate for a signing key.
- Static data signed by bank
  - CC no (PAN)., exp. date., etc.
- Card generates
  - A cryptogram (AC) to send to the bank as evidence of the transaction,
  - A signature (SDAD) that is checked by the bank.



https://www.sec.gov/Archives/edgar/data/1141391/000119312508034694/d10k.htm

Cardholder Verification

- Online PIN
  - Terminal sends PIN and PAN to bank to check
  - Card plays no part in this.
- Offline plaintext PIN
  - Terminal sends a "verify pin" command.
  - Card replies with success or failure.
  - 3 failed tries locks the card.
- Offline encrypted PIN
  - Card sends reader a nonce.
  - Terminal sends a "verify pin" and pin and nonce encrypted with the cards public key
  - Card decrypts this and replies with success or failure.
  - 3 failed tries locks the card.

Card Authentication Mechanism (CAM)

Static Data Authentication (SDA):

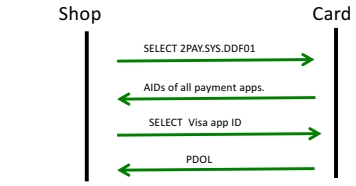
- All track data is on the card signed with RSA.
- A public key certificate for the card key is signed.
- *This includes the service code*

- Dynamic Data Authentication (DDA) adds:
  - Terminal sends nonce to card
  - Card will sign this nonce
  - Provide cert. for signing key.

Kernels

- Each credit card company can implement then own version of the protocol.
  - E.g. Visa contact vs Visa contactless vs MasterCard contact vs American Express vs …
- Same command set, different order and parameters.
- Each application as a unique AID that is selected at the start of the protocol.
  - E.g. Visa debit card app as AID "A0000000031010"

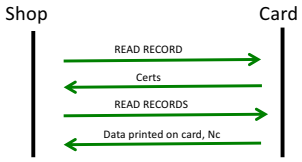
Protocol Set Up Stage



PDOL = Processing Options Data Object List

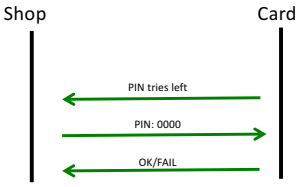
- list of data the reader must provide to the card.

Protocol set up

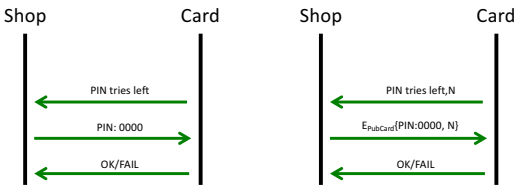


- Shop reader then checks the signature on the SDAD data.
- If this is correct it shop reader accepts the payment and sends the AC to the bank.
- The bank checks the AC and transfers the money.

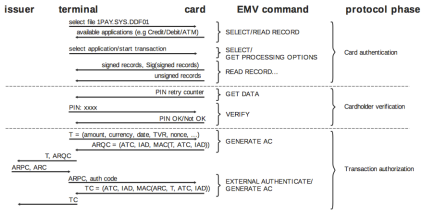
Card holder verification



Card holder verification



Chip & PIN (contact) protocol



Chip and PIN is Broken

Steven J. Murdoch, Saar Drimer, Ross Anderson, Mike Bond
University of Cambridge
Computer Laboratory
Cambridge, UK
http://www.cl.cam.ac.uk/users/{sjm217,sd410,rja14,mkb23}

***Abstract*—EMV is the dominant protocol used for smart card payments worldwide, with over 730 million cards in circulation. Known to bank customers as "Chip and PIN", it is used in Europe; it is being introduced in Canada; and there is pressure from banks to introduce it in the USA too. EMV secures credit and debit card transactions by authenticating both the card and the customer presenting it through a combination of cryptographic authentication codes, digital signatures, and the entry of a PIN. In this paper we describe and demonstrate a protocol flaw which allows criminals to use a genuine card to make a payment without knowing the card's PIN, and to remain undetected even when the merchant has an online connection to the banking network. The fraudster performs a man-in-the-middle attack to trick the terminal into believing the PIN verified correctly, while telling the card that no PIN was entered at all. The paper considers how the flaws arose, why they remained unknown despite EMV's wide deployment for the best part of a decade, and how they might be fixed. Because we have found and validated a practical attack against the core functionality of EMV, we conclude that the protocol is broken. This failure is significant in the field of protocol design, and also has important public policy implications, in light of growing reports of fraud on stolen EMV cards. Frequently, banks deny such fraud victims a refund, asserting that a card cannot be used without the correct PIN, and**

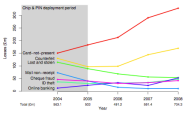
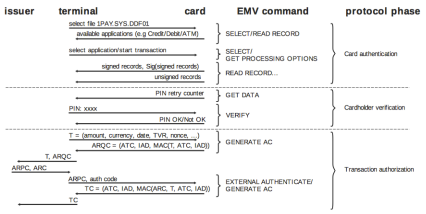


Figure 1. Fraud statistics on UK-issued cards [6]

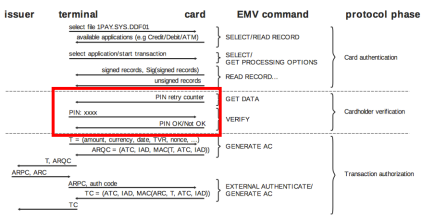
a symmetric key shared between the payment card and the bank that issued the card to the customer (the issuer).

EMV was heavily promoted under the "Chip and PIN" brand during its national rollout in the UK. The technology was advertised as a solution to increasing card fraud: a chip

Chip & PIN (contact) protocol



Chip & PIN (contact) protocol



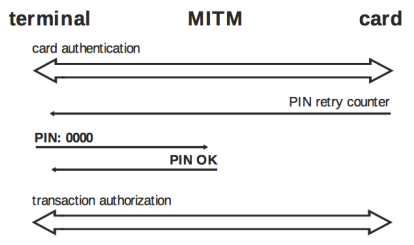
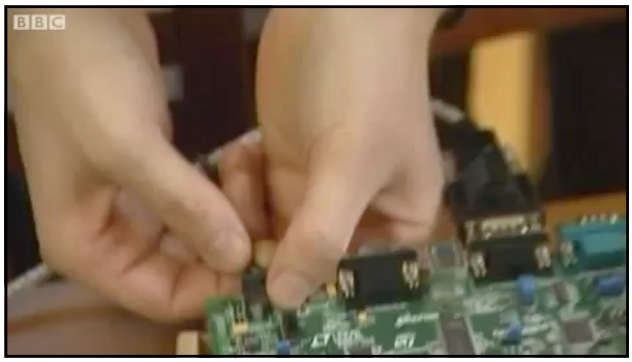


Figure 3. The man-in-the-middle suppresses the PIN Verify command to the card, and tells the terminal that the PIN has been verified correctly. A complete transaction is detailed in Appendix A.



PIN numbers

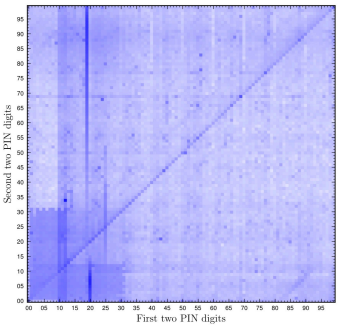
- PIN numbers aren't as secure as you might think.

A birthday present every eleven wallets? The security of customer-chosen banking PINs

Joseph Bonneau, Sören Preibusch, Ross Anderson

Computer Laboratory
University of Cambridge
{jcb82,sdp36,rja14}@cl.cam.ac.uk

Abstract. We provide the first published estimates of the difficulty of guessing a human-chosen 4-digit PIN. We begin with two large sets of 4-digit sequences chosen outside banking for online passwords and smartphone unlock-codes. We use a regression model to identify a small number of dominant factors influencing user choice. Using this model and a survey of over 1,100 banking customers, we estimate the distribution of banking PINs as well as the frequency of security-relevant behaviour such as sharing and reusing PINs. We find that guessing PINs based on the victims' birthday, which nearly all users carry documentation of, will enable a competent thief to gain use of an ATM card once for every 11–18 stolen wallets, depending on whether banks prohibit weak PINs such as 1234. The lesson for cardholders is to never use one's date of birth as



Contactless EMV

- Contactless just removes the card verification step and run over NFC.
- This lets MasterCard/Visa get a 1.5% cut of all the small transaction they were missing.
- MasterCard/Visas will use some of this money to refund loss due to people using stolen cards.

Relay Cost Bounding for Contactless EMV Payments

Tom Chothia[1], Flavio D. Garcia[1], Joeri de Ruiter[2], Jordi van den Breekel[3], and Matthew Thompson[1]

[1] School of Computer Science, University of Birmingham, UK
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen
[3] Department of Mathematics and Computer Science, Technical University Eindhoven

Abstract. This paper looks at relay attacks against contactless payment cards, which could be used to wirelessly pickpocket money from victims. We discuss the two leading contactless EMV payment protocols (Visa's payWave and MasterCard's PayPass). Stopping a relay attack against cards using these protocols is hard: either the overhead of the communication is low compared to the (cryptographic) computation by the card or the messages can be cached before they are requested by the terminal. We propose a solution that fits within the EMV Contactless specification to make a payment protocol that is resistant to relay attacks from commercial off-the-shelf devices, such as mobile phones. This solution does not require significant changes to the cards and can easily be added to existing terminals. To prove that our protocol really does stop relay attacks, we develop a new method of automatically checking defences against relay attacks using the applied pi-calculus and the tool ProVerif.

1 Introduction

EMV is the most widely used standard for payments using smart cards [13]. The EMV Contactless specification has been introduced to support contactless smart cards [14]. For every payment provider a different variation of the specification exists. MasterCard

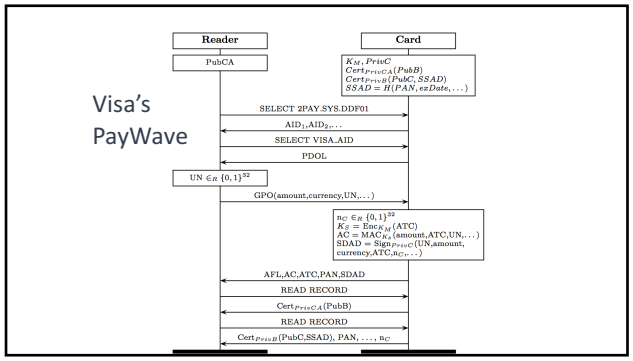
ISO 14443: Level 1 messages

Reader:	52	WUPA (wake up)
Tag:	44 03	ATQA (Respond)
Reader	93 20	SELECT
Tag:	88 04 34 74 cc	UID (tags unique ID)
Reader	93 70 88 04 34 74 cc	SELECT card via UID

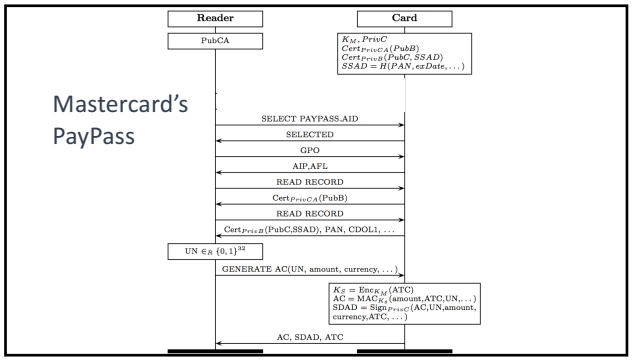
If there is more than one card present the reader picks a UID at random.

Messages include checksums (not shown). If messages from different cards collide, the reader sends SELECT again.

Visa's PayWave



Mastercard's PayPass



Protocol model

$$
\begin{array}{ll}
\textit{Reader} = \bar{c}\langle\text{SELECT,PAYSYSDDF}\rangle. & \textit{Card} = c(=\text{SELECT},=\text{PAYSYSDDF}).\\
\quad c(=\text{AID}). & \quad \bar{c}\langle\text{AID}\rangle.\\
\quad \bar{c}\langle\text{SELECT,aid}\rangle. & \quad c(=\text{SELECT},=\text{AID}).\\
\quad c(=\text{PDOL}). & \quad \nu\, n_C.\bar{c}\langle\text{PDOL}\rangle.\\
\quad \nu\, n_R.\bar{c}\langle\text{GPO,amt},n_R\rangle. & \quad c(=\text{GPO,amt}',n_R').\\
\quad c(n_C',\text{atc}',\text{PAN}'). & \quad \bar{c}\langle n_C,\text{atc,PAN}\rangle.\\
\quad \bar{c}\langle\text{GENERATE AC}\rangle. & \quad c(=\text{GENERATE AC}).\\
\quad c(\text{sdad}',\text{ac}'). & \quad \text{let } \text{mac}_K = \mathit{genkey}(\text{atc,bank}_K) \text{ in}\\
\quad \bar{c}\langle\text{READ RECORD}\rangle.c(\mathit{ssad}'). & \quad \text{let ac} = \mathit{mac}(\langle\text{amt}',n_R',\text{atc}\rangle,\text{mac}_K) \text{ in}\\
\quad \bar{c}\langle\text{READ RECORD}\rangle.c(\text{cert}'). & \quad \text{let sdad} =\\
\quad \text{let cardPub}_K' = & \qquad \mathit{sign}(\langle n_R,n_C,\text{amt,atc,ac}\rangle,\text{card}_K) \text{ in}\\
\qquad \mathit{check}(\text{cert}',\mathit{pk}(\text{bank}_K)) & \quad \bar{c}\langle\mathit{sdad}\rangle.\\
\quad \text{if } \mathit{check}(\text{sdad}',\text{cardPub}_K') = & \quad c(=\text{READ RECORD}).\\
\qquad \langle n_R,n_C',\text{amt,atc}',\text{ac}'\rangle & \quad \bar{c}\langle\mathit{sign}(\langle\text{PAN,expDate}\rangle,\text{bank}_K)\rangle.\\
\quad \bar{c}\langle\text{readerAccepts}\rangle & \quad c(=\text{READ RECORD}).\bar{c}\langle\mathit{cert}\rangle
\end{array}
$$

$$
\begin{array}{l}
\textit{System} = \ \nu\,\text{bank}_K.(\bar{c}\langle\mathit{pk}(\text{bank}_K)\rangle \mid\ !\nu\,\text{amount}.!\textit{Reader}\\
\quad \mid\ !(\nu\,\text{PAN}.\nu\,\text{expDate}.\nu\,\text{card}_K.\text{let cert} = \mathit{sign}(\mathit{pk}(\text{card}_K),\text{bank}_K) \text{ in } !\nu\,\text{atc}.!\textit{Card}))
\end{array}
$$

Modelling of contactless EMV

- See proverif models
- Mastercard is secure
- Visa has a free lunch attack

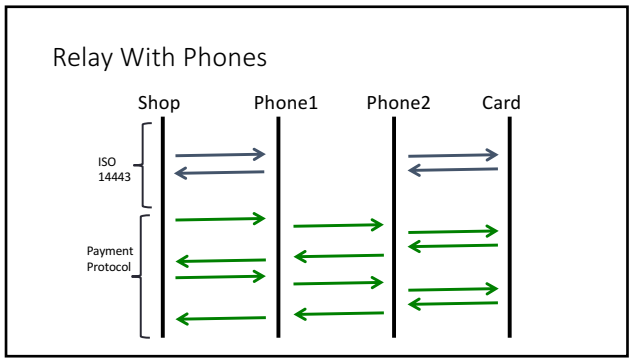
Relay Cost Bounding for Contactless EMV Payments

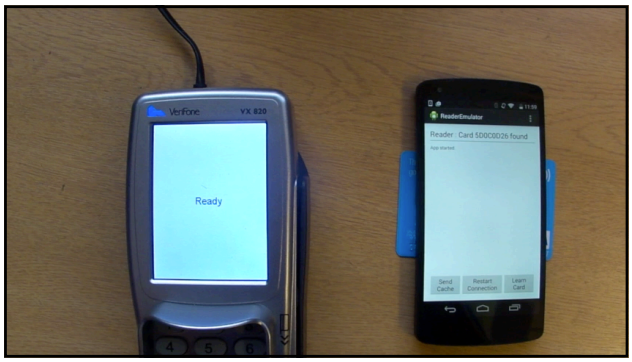
Tom Chothia[1], Flavio D. Garcia[1], Joeri de Ruiter[2], Jordi van den Breekel[3], and Matthew Thompson[1]

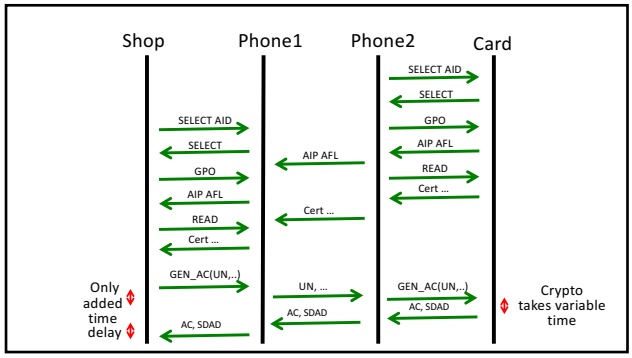
[1] School of Computer Science, University of Birmingham, UK
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen
[3] Department of Mathematics and Computer Science, Technical University Eindhoven

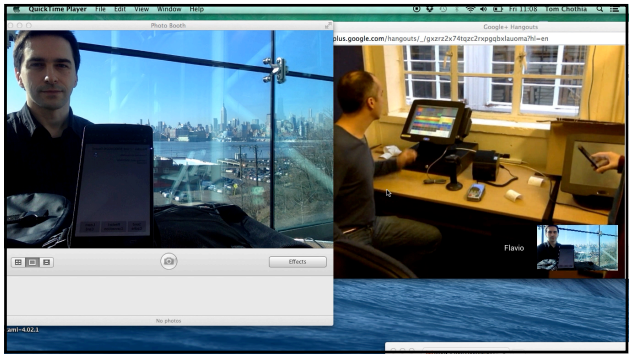
Abstract. This paper looks at relay attacks against contactless payment cards, which could be used to wirelessly pickpocket money from victims. We discuss the two leading contactless EMV payment protocols (Visa's payWave and MasterCard's PayPass). Stopping a relay attack against cards using these protocols is hard: either the overhead of the communication is low compared to the (cryptographic) computation by the card or the messages can be cached before they are requested by the terminal. We propose a solution that fits within the EMV Contactless specification to make a payment protocol that is resistant to relay attacks from commercial off-the-shelf devices, such as mobile phones. This solution does not require significant changes to the cards and can easily be added to existing terminals. To prove that our protocol really does stop relay attacks, we develop a new method of automatically checking defences against relay attacks using the applied pi-calculus and the tool ProVerif.

Relay With Phones

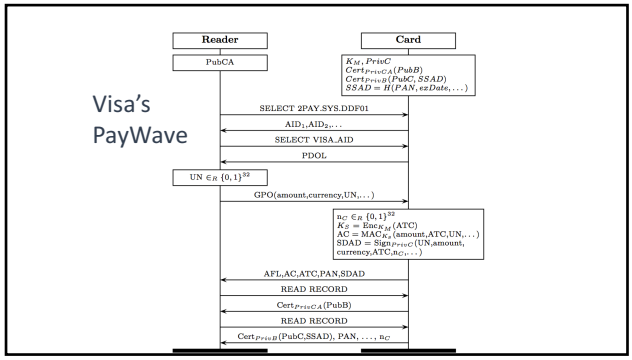




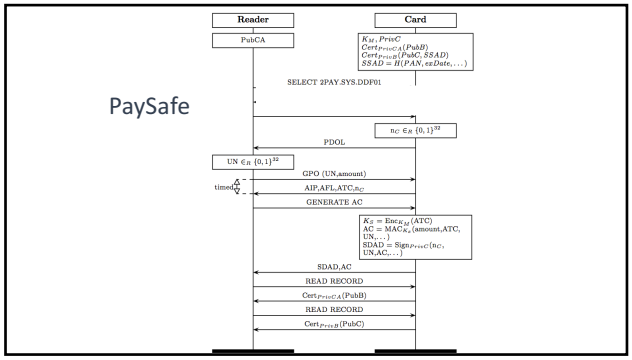




Visa's PayWave



PaySafe



Key Observation

- The attackers can do anything they want before or after the time-bound step.
- Attackers can reply to the time-bound step with their own message or a replayed message.
- The attacker does not have time to
  - look at the time-bound step,
  - and then send a message to the card
  - and then reply to the reader.

This is equivalent to saying that the attacker cannot talk to the card during the time bound step.

Key Observation

- In our formal model, we lock the card during the time-bound step.
  - It cannot communicate with the attacker or the reader.
- If the attacker can find a sequence of actions that allow the reader to successfully terminate, then there is a relay attack.
- If the reader cannot terminate then the protocol is safe from relay attacks.

Locking the Card Process Using Phases

- Phases enforce order on processes:
  - e.g. 2:P | 2:a(x).3:Q | 3:R
- To model relays we use three phases: 0,1 & 2.
- The reader, attacker and card can all act in phase 0 & 2
- The attacker can act in phase 0,1 & 2
- The reader moves to phase 1 before sending its time action & moves to phase 2 when it gets the reply.

Locking the Card Process Using Phases

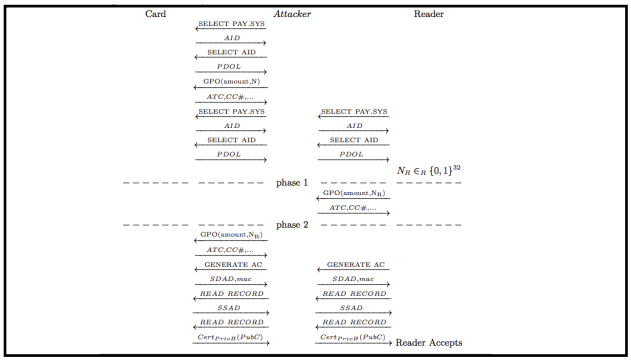
- Cards and readers must be able to jump from phase 1 to 3 at any point

$$phasesSet(P) = \{C[2{:}M(x).P'] : P = C[M(x).P']\}$$
$$phases(P) = !P_1 \mid !P_2 \mid \cdots \mid !P_n \quad where \{P_1, \ldots, P_n\} = phasesSet(P)$$

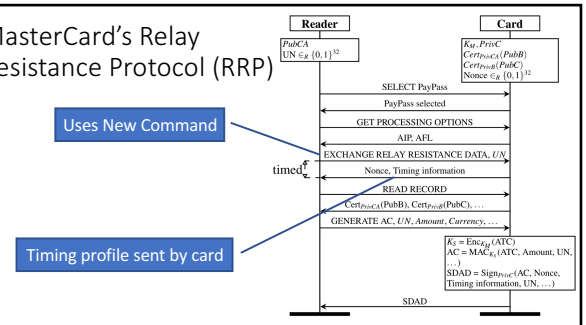
$TestReader = \ldots$
$\quad c(=\text{PDOL}).\nu\, n_R.$
$\quad 1{:}\overline{c}\langle \text{GPO,amt}, n_R\rangle.$
$\quad c(n'_C, \text{atc}', \text{ccNo}').$
$\quad 2{:}\overline{c}\langle \text{GENERATE AC}\rangle.$
$\quad c(\text{sdad}', \text{ac}').$
$\quad \ldots$
$\quad \text{if } check(\text{scad}', \text{cardPub}'_K) = (n_R, n'_C, \text{amt}, \text{atc}', \text{ac}')$
$\quad \overline{c}\langle \text{phaseReaderAccepts}\rangle$

$SystemP = \nu\, \text{bank}_K.(\overline{c}\langle pk(\text{bank}_K)\rangle$
$\quad \mid \nu\, \text{amount}.TestReader$
$\quad \mid !\nu\, \text{amount}.Readers$
$\quad \mid !(\nu\, \text{ccNo}.\nu\, \text{expDate}.\nu\, \text{card}_K.$
$\quad\quad \text{let cert} = sign(pk(\text{card}_K), \text{bank}_K)$
$\quad\quad \text{in } !\nu\, \text{atc}.Cards))$

where:
$Cards = phases(Card)$
$Readers = phases(Reader)$



MasterCard's Relay Resistance Protocol (RRP)



Uses New Command

Timing profile sent by card

Modelling and Analysis of a Hierarchy of Distance Bounding Attacks

Tom Chothia
University of Birmingham
Birmingham, UK

Joeri de Ruiter
Radboud University
Nijmegen, The Netherlands

Ben Smyth
University of Luxembourg,
Luxembourg

Abstract

We present an extension of the applied pi-calculus that can be used to model distance bounding protocols. A range of different security properties have been suggested for distance bounding protocols; we show how these can be encoded in our model and prove a partial order between them. We also relate the different security properties to particular attacker models. In doing so, we identify a new property, which we call *uncompromised distance bounding*, that captures the attacker model for protecting devices such as contactless payment cards or car entry systems, which assumes that the prover being tested has not been compromised, though other provers may have been. We show how to compile our new calculus into the applied pi-calculus so that protocols can be

A distance bounding attack occurs when a verifier is deceived into believing they are co-located with a prover, when they are not. Attackers may relay, replay and alter messages, as well as trying to predict or preempt timed challenges. Some distance bounding protocols also aim to defend against a "dishonest prover" attacker , i.e., an attacker that knows all of the secret values of a normal prover, but will misuse them to try to trick a verifier. Other attacker models consider a weaker "terrorist prover," i.e., a dishonest prover that will not reveal its long term keys. The literature on symbolic verification of distance bounding protocols includes five different types of attacks, each of which uses some combination of basic, unprivileged attackers, dishonest prover attackers, and terrorist fraud attackers. We describe these

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 ]
```

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

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

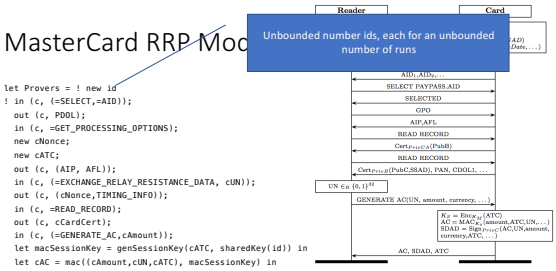
$[\text{ in c(x).P | out c<n>.Q }]_r$ -> $[\text{ P\{n/x\} | Q }]_r$

$[\text{out c<n>.Q}]_r$ | $[\text{P}]_0$ -> $[\text{Q}]_r$ | $[\text{out c<n>|P}]_0$

$[\text{out c<n>.Q }]_r$ -> $[\text{ out c<n> | Q }]_r$

MasterCard RRP Mod

Unbounded number ids, each for an unbounded number of runs



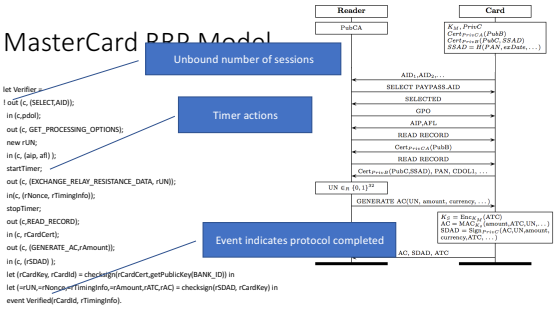

```
let Provers = ! new id
! in (c, (=SELECT,=AID));
  out (c, PDOL);
  in (c, =GET_PROCESSING_OPTIONS);
  new cNonce;
  new cATC;
  out (c, (AIP, AFL));
  in (c, (=EXCHANGE_RELAY_RESISTANCE_DATA, cUN));
  out (c, (cNonce,TIMING_INFO));
  in (c, =READ_RECORD);
  out (c, cCardCert);
  in (c, (=GENERATE_AC,cAmount));
  let macSessionKey = genSessionKey(cATC, sharedKey(id)) in
  let cAC = mac((cAmount,cUN,cATC), macSessionKey) in
  let cSDAD = sign((cUN,cNonce,TIMING_INFO,cAmount,cATC,cAC), getPrivateKey(idP)) in
  out (c, (cSDAD) ).
```

MasterCard RRP Model

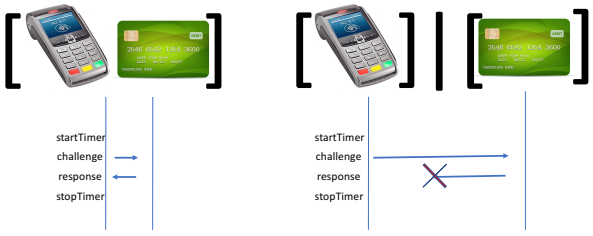
Unbound number of sessions

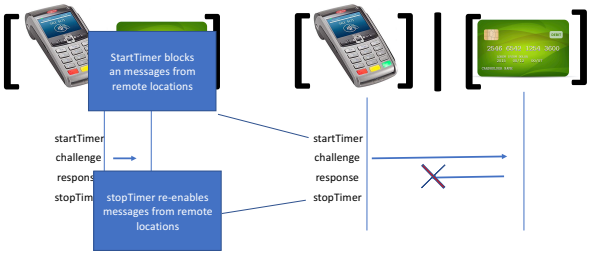
Timer actions

Event indicates protocol completed




```
let Verifier =
! out (c, (SELECT,AID));
  in (c,pdol);
  out (c, GET_PROCESSING_OPTIONS);
  new rUN;
  in (c, (aip, afl) );
  startTimer;
  out (c, (EXCHANGE_RELAY_RESISTANCE_DATA, rUN));
  in(c, (rNonce, rTimingInfo));
  stopTimer;
  out (c,READ_RECORD);
  in (c, rCardCert);
  out (c, (GENERATE_AC,rAmount));
  in (c, (rSDAD) );
  let (rCardKey, rCardId) = checksign(rCardCert,getPublicKey(BANK_ID)) in
  let (=rUN,=rNonce,=rTimingInfo,=rAmount,rATC,rAC) = checksign(rSDAD, rCardKey) in
  event Verified(rCardId, rTimingInfo).
```





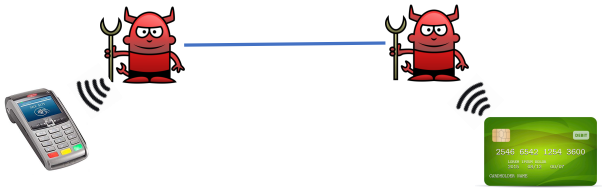
Some Questions

- 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?

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

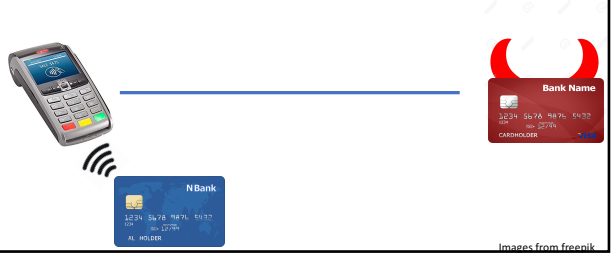
Relay/Mafia Fraud: attackers relay and interfere with messages



Images from freepik

Distance Fraud: remote dishonest prover tricks the verifier

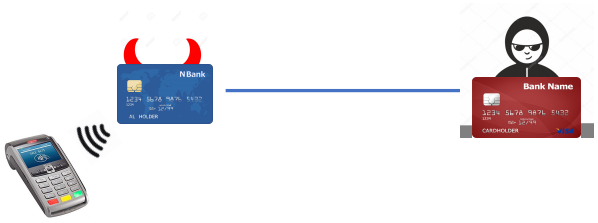
Distance Hijacking: remote dishonest prover uses a local honest prover



Images from freepik

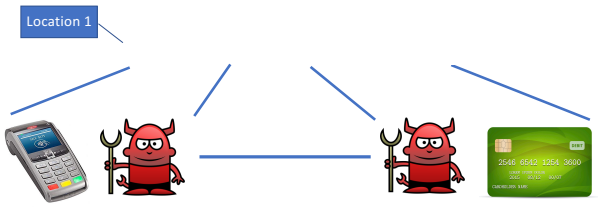
Terrorist Fraud: A remote dishonest prover* and local attacker

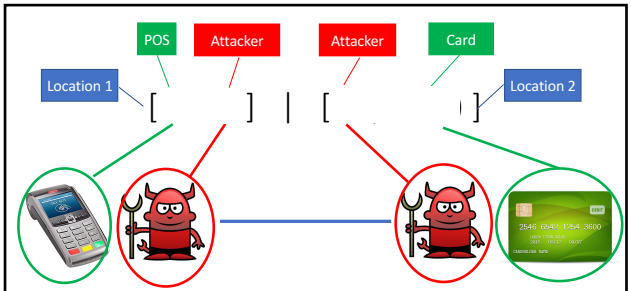
Assisted Distance Fraud: remote dishonest prover* and local dishonest prover



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





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 Dis** 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

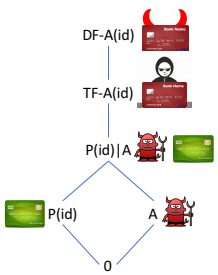
- Arbitrary number of provers — P(id)
- Verifier looking for one of "id" — V(id)
- A Dolev–Yao attacker — A
- Other provers — P(id')

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.

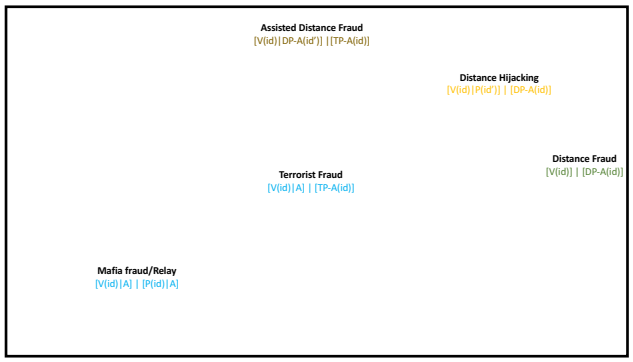


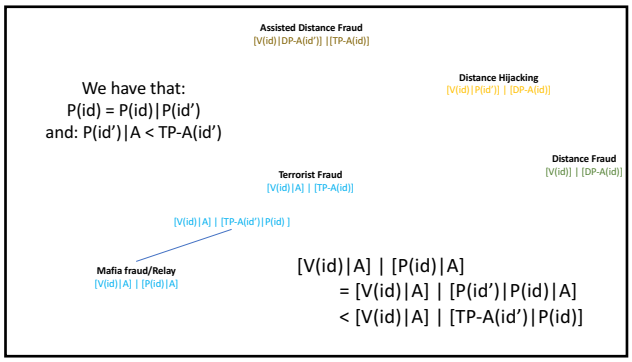
Equalities between processes

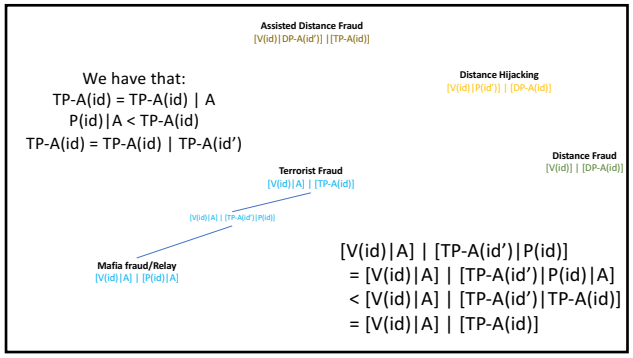
[V(id)|A] | [P(id)|A]

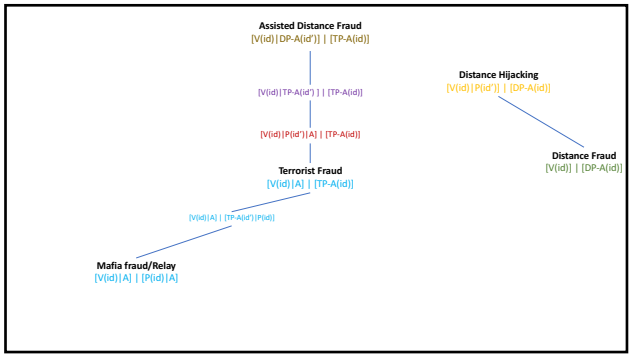
=

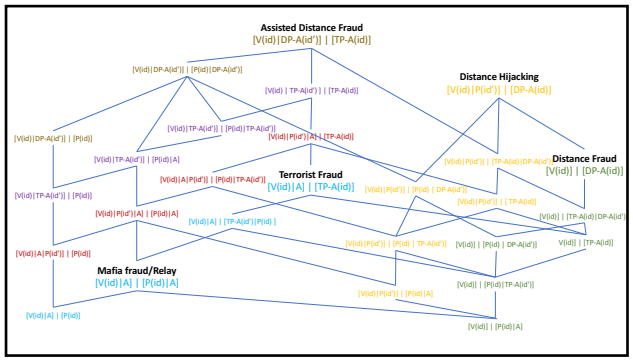
[V(id)|A] | [P(id)|A|P(id')]

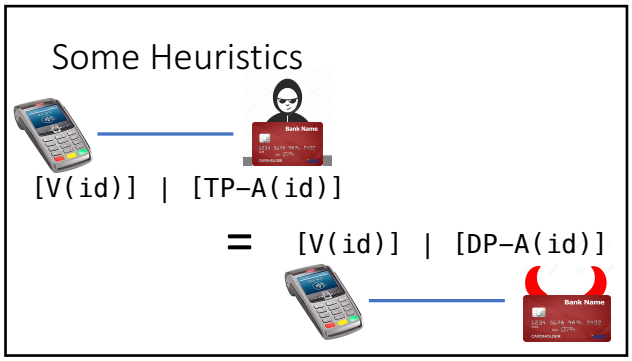

Assisted Distance Fraud
[V(id)|DP-A(id')] |[TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]

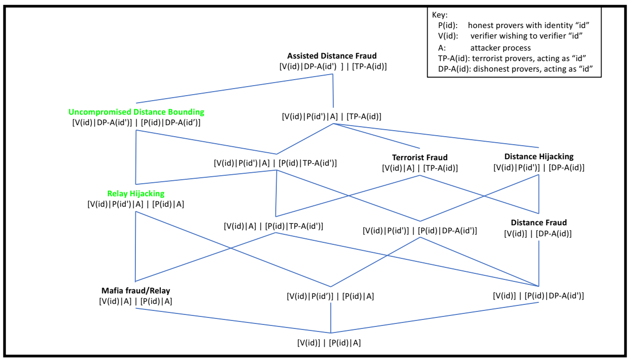

Assisted Distance Fraud
[V(id)|DP-A(id')] |[TP-A(id)]
We have that:
P(id) = P(id)|P(id')
and: P(id')|A < TP-A(id')
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
[V(id)|A] | [TP-A(id')|P(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)|A] | [P(id)|A]
= [V(id)|A] | [P(id')|P(id)|A]
< [V(id)|A] | [TP-A(id')|P(id)]

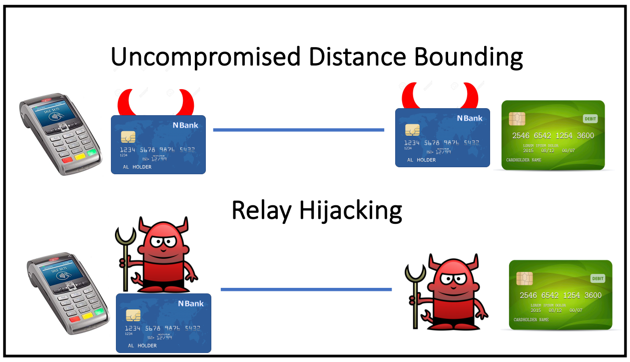

Assisted Distance Fraud
[V(id)|DP-A(id')] |[TP-A(id)]
We have that:
TP-A(id) = TP-A(id) | A
P(id)|A < TP-A(id)
TP-A(id) = TP-A(id) | TP-A(id')
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
[V(id)|A] | [TP-A(id')|P(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]
[V(id)|A] | [P(id)|A]
= [V(id)|A] | [TP-A(id')|P(id)|A]
< [V(id)|A] | [TP-A(id')|TP-A(id)]
= [V(id)|A] | [TP-A(id)]

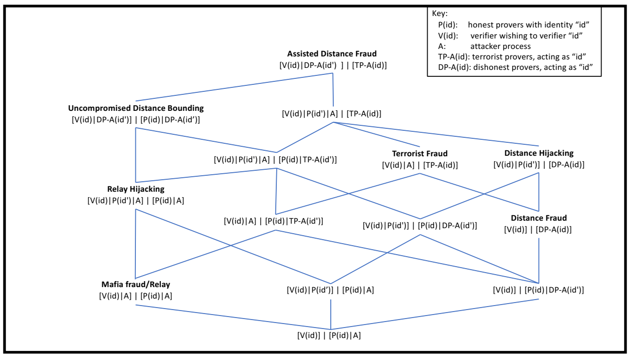

Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
[V(id)|TP-A(id')] | [TP-A(id)]
[V(id)|P(id')|A] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
[V(id)|A] | [TP-A(id')|P(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]

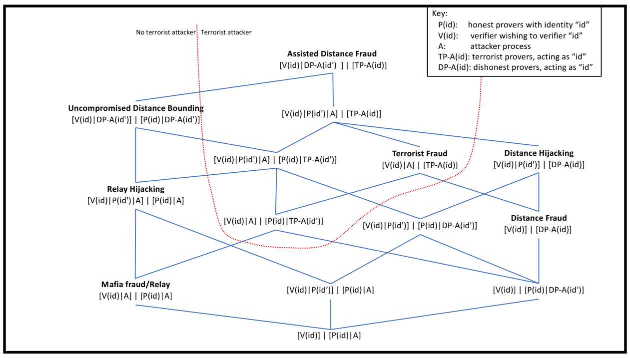

Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Distance Hijacking
[V(id)|P(id')] | [DP-A(id)]
Distance Fraud
[V(id)] | [DP-A(id)]
Terrorist Fraud
[V(id)|A] | [TP-A(id)]
Mafia fraud/Relay
[V(id)|A] | [P(id)|A]

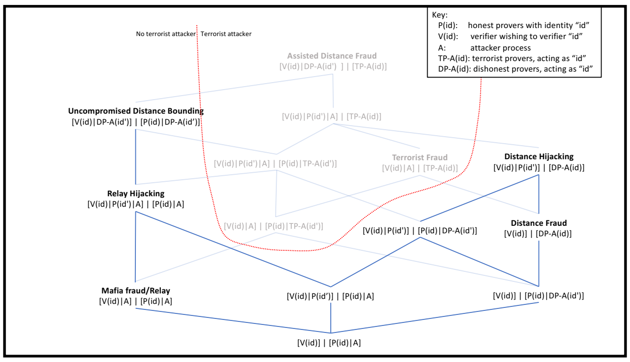

Some Heuristics
[V(id)] | [TP–A(id)]
=
[V(id)] | [DP–A(id)]
Bank Name
Bank Name

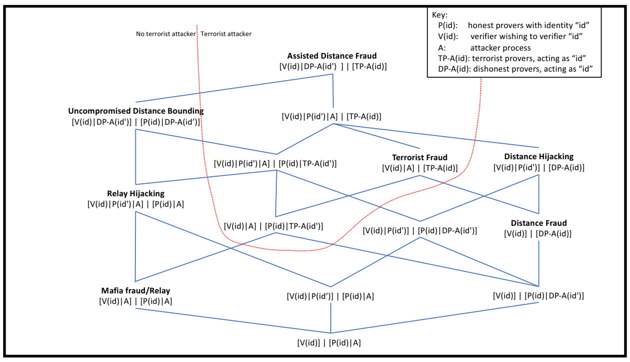

Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id')]
[V(id)|P(id')|A] | [TP-A(id)]
[V(id)|P(id')|A] | [P(id)|TP-A(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]

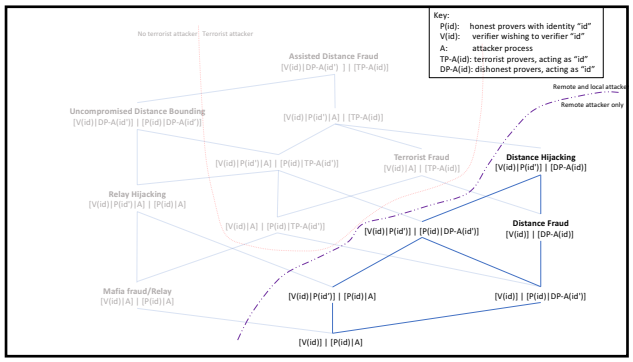

Uncompromised Distance Bounding
Relay Hijacking

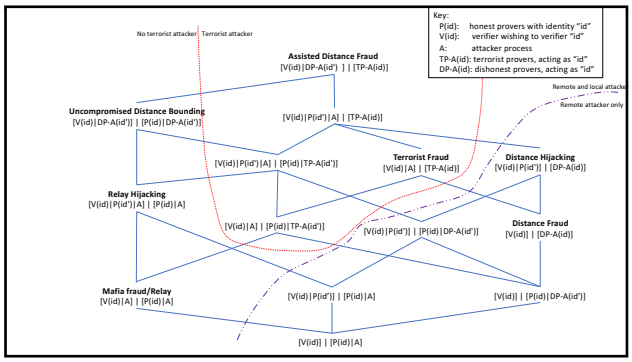

Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised Distance Bounding
[V(id)|DP-A(id')] | [P(id)|DP-A(id')]
[V(id)|P(id')|A] | [TP-A(id)]
[V(id)|P(id')|A] | [P(id)|TP-A(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]

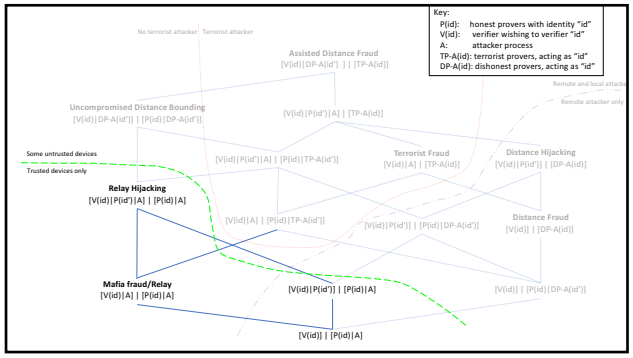

No terrorist attacker
Terrorist attacker
Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised 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)|TP-A(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]

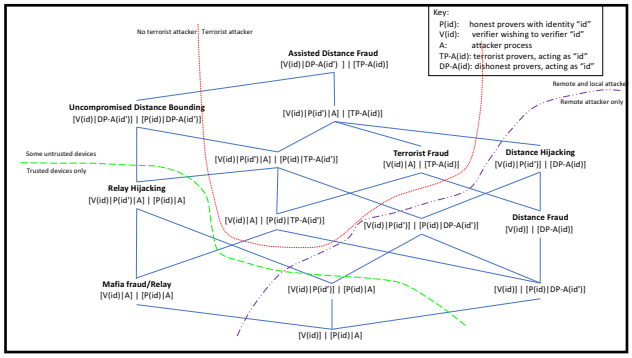

No terrorist attacker
Terrorist attacker
Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised 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)|TP-A(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]

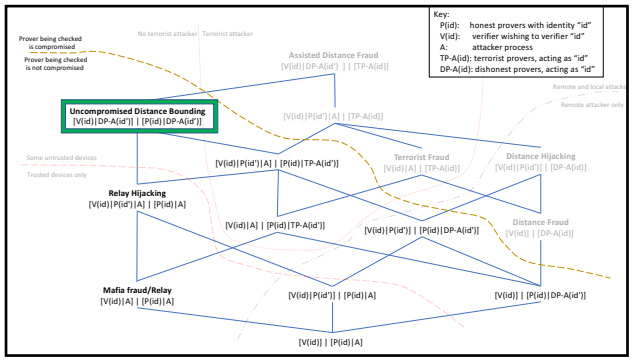

No terrorist attacker
Terrorist attacker
Key:
P(id): honest provers with identity "id"
V(id): verifier wishing to verifier "id"
A: attacker process
TP-A(id): terrorist provers, acting as "id"
DP-A(id): dishonest provers, acting as "id"
Assisted Distance Fraud
[V(id)|DP-A(id')] | [TP-A(id)]
Uncompromised 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)|TP-A(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.

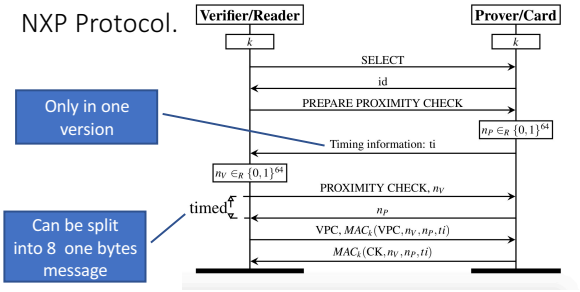
Process at the same location as the verifier can act in all phases
Process at a different location can only act in Phase 1 and Phase 2.

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.



	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