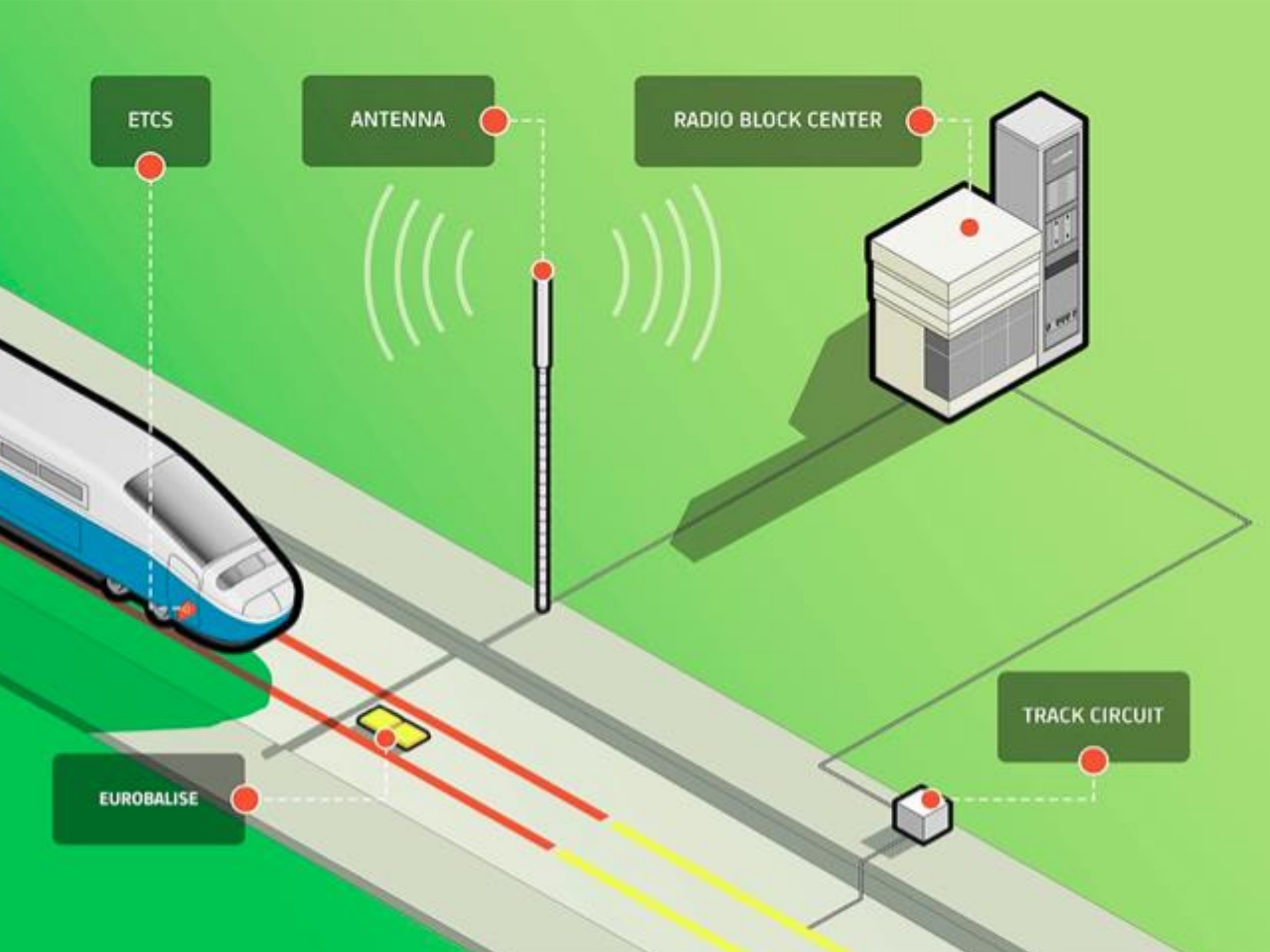
Protocol security

Mihai Ordean
Designing Secure Systems
University of Birmingham

Overview

- Device security
  - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
  - Is the code authenticated ? (i.e. has not been tampered with)
- Data security (in the cloud)
  - Is the stored data is accessible to everyone? (e.g. encrypted)
  - Is the stored data authenticated?
- Metadata security
  - What does metadata reveal about data?
  - Can we tamper the metadata?
- **Protocol security**
  - Is data in transit visible?
  - Can data in transit be tampered with?

European Rail Traffic Management System (ERTMS)


ETCS
ANTENNA
RADIO BLOCK CENTER
TRACK CIRCUIT
EUROBALISE

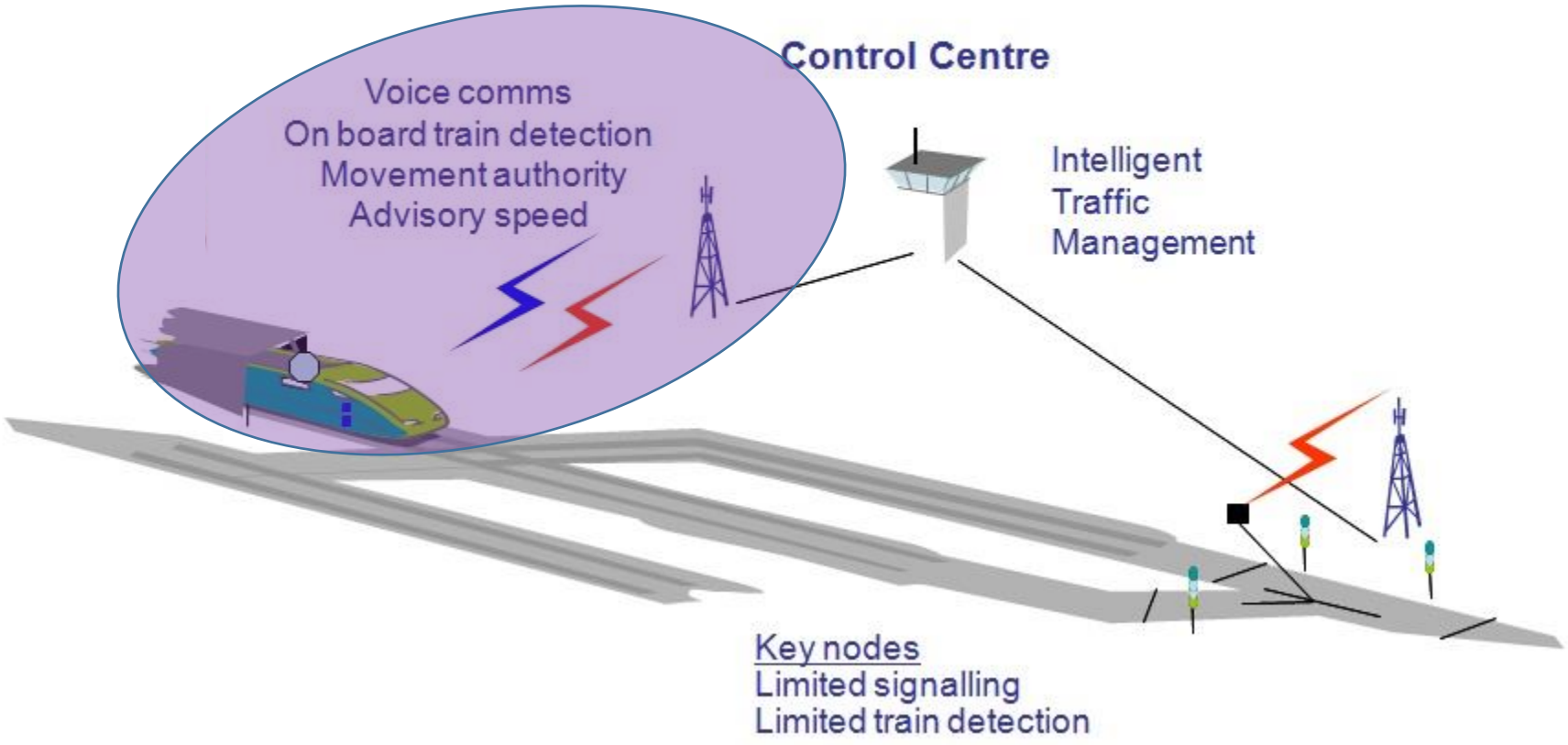
ERTMS Overview

The European Rail Traffic Management System (ERTMS) is a suite of protocols used to deliver next-generation train management and signalling.

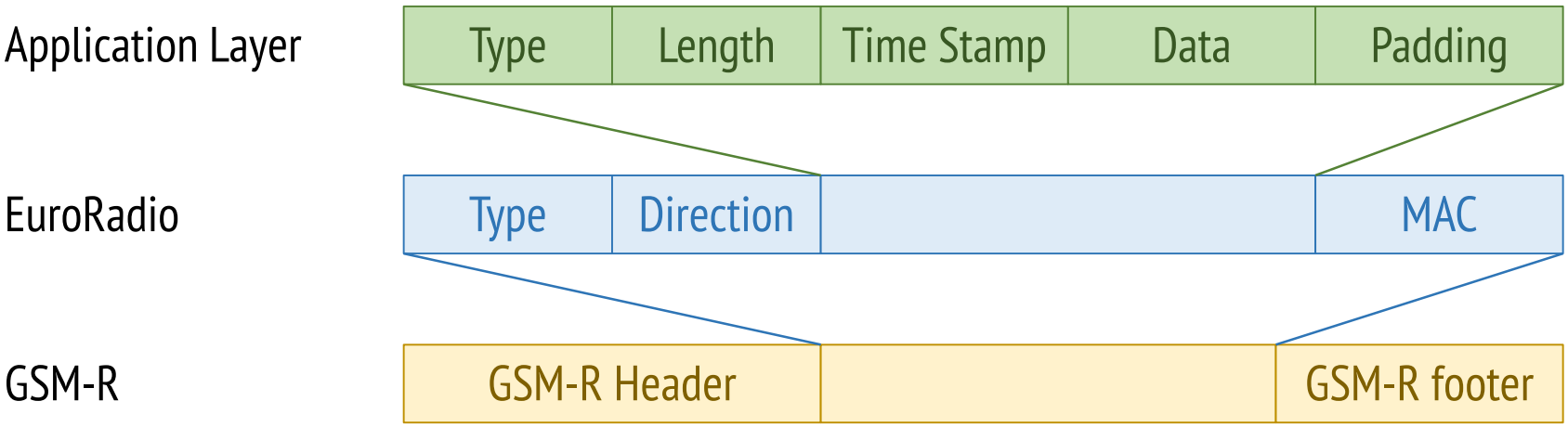
ERTMS components:

- GSM-R – encryption/physical interaction
- EuroRadio – message authentication
- Application Layer protocol - instructions

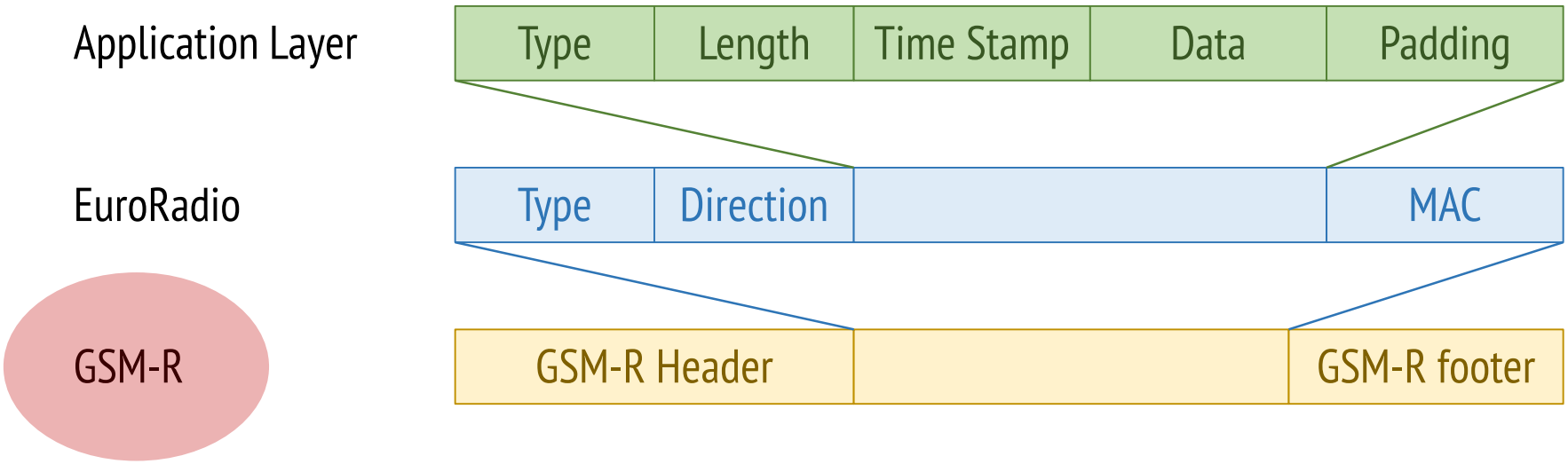
ERTMS Overview



ERTMS stack


Application Layer
Type
Length
Time Stamp
Data
Padding
EuroRadio
Type
Direction
MAC
GSM-R
GSM-R Header
GSM-R footer

ERTMS stack


Application Layer
Type
Length
Time Stamp
Data
Padding
EuroRadio
Type
Direction
MAC
GSM-R
GSM-R Header
GSM-R footer

GSM-R

- Provides data encryption on the ERTMS stack
- Based on the GSM Mobile Communications Standard

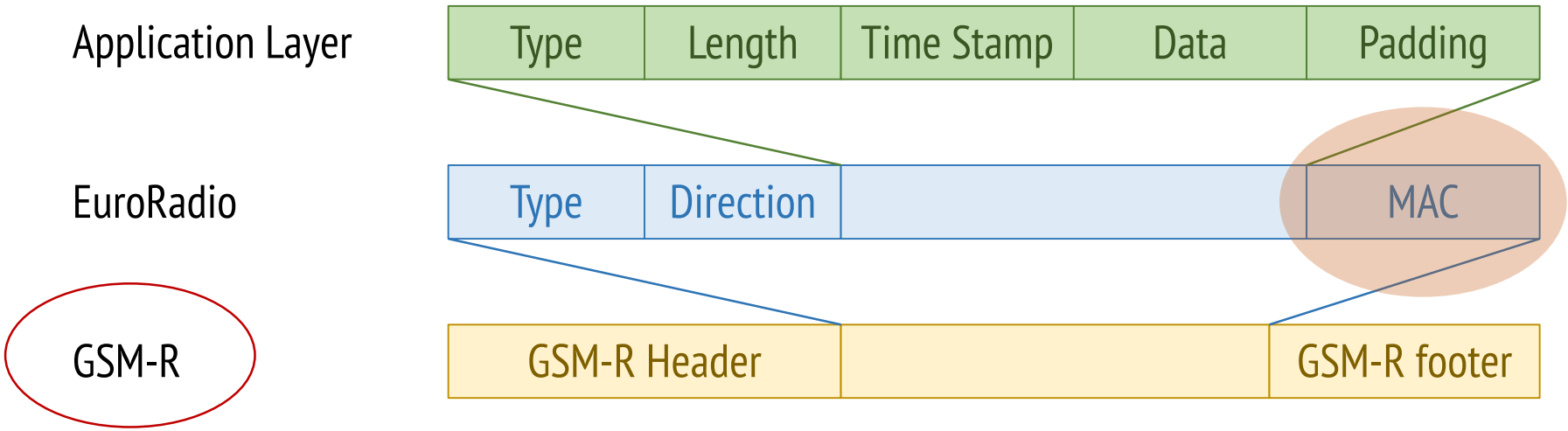
  (i.e. basically 2G) with:
  - different frequency ranges
  - rail-specific functionality (multi-party communication, emergency calling functionality, priority-based pre-emption, etc.)
- Crypto:
  - A5/1* a stream cipher based on (LFSRs)
  - A5/3 (optionally) a block cipher

---

* broken:

1. Elad Barkan, Eli Biham, Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. J. Cryptology 21(3): 392-429 (2008)
2. L. Karstensen. GSM A5/1 rainbow tables in Oslo, Norway. Available: https://lassekarstensen:wordpress:com/2013/08/08/gsm-a51-rainbow-tables-in-oslo-norway/, 2015.
3. https://www.ckn.io/blog/2016/01/25/gsm-sniffing-voice-traffic/
4. https://www.youtube.com/playlist?list=PLRovDyowOn5F_TFotx0n8A79ToZYD2lOv

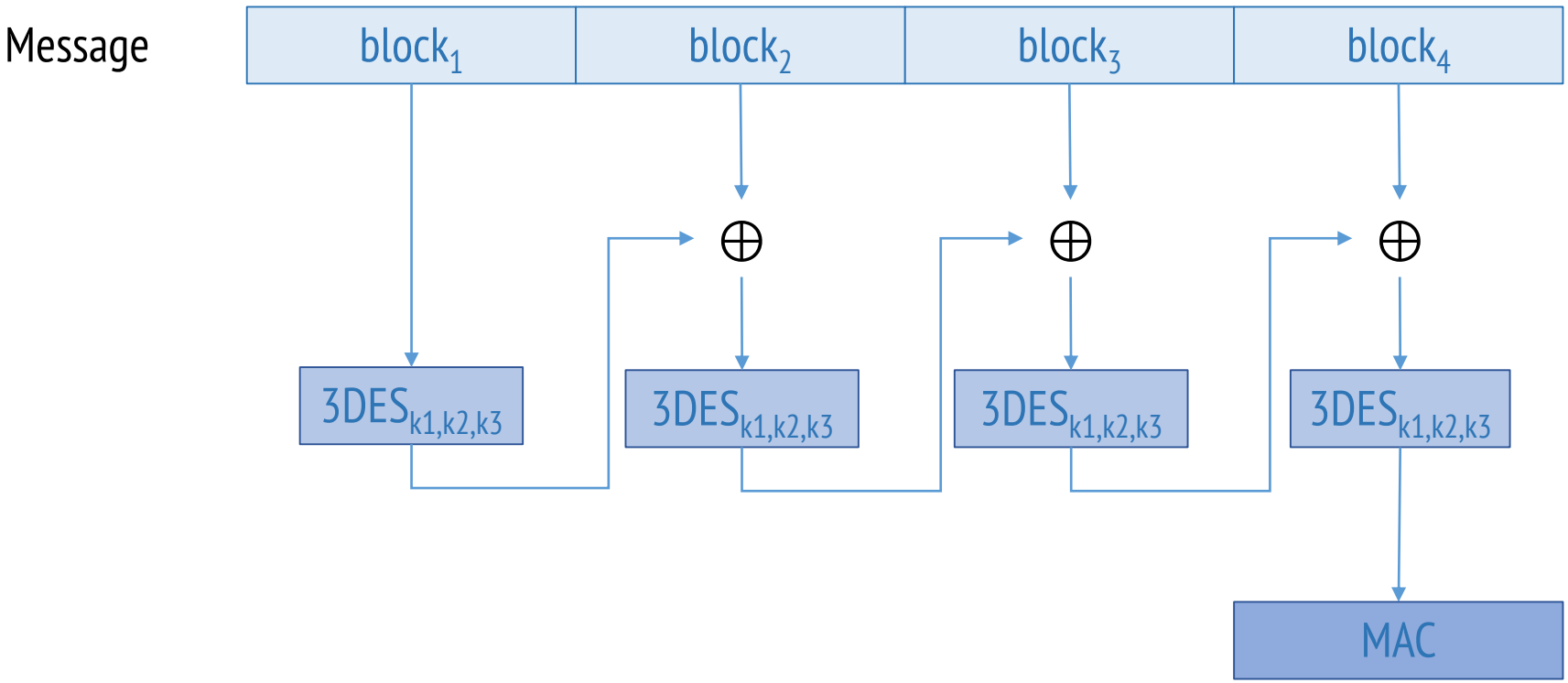
ERTMS stack


Application Layer
Type
Length
Time Stamp
Data
Padding
EuroRadio
Type
Direction
MAC
GSM-R
GSM-R Header
GSM-R footer

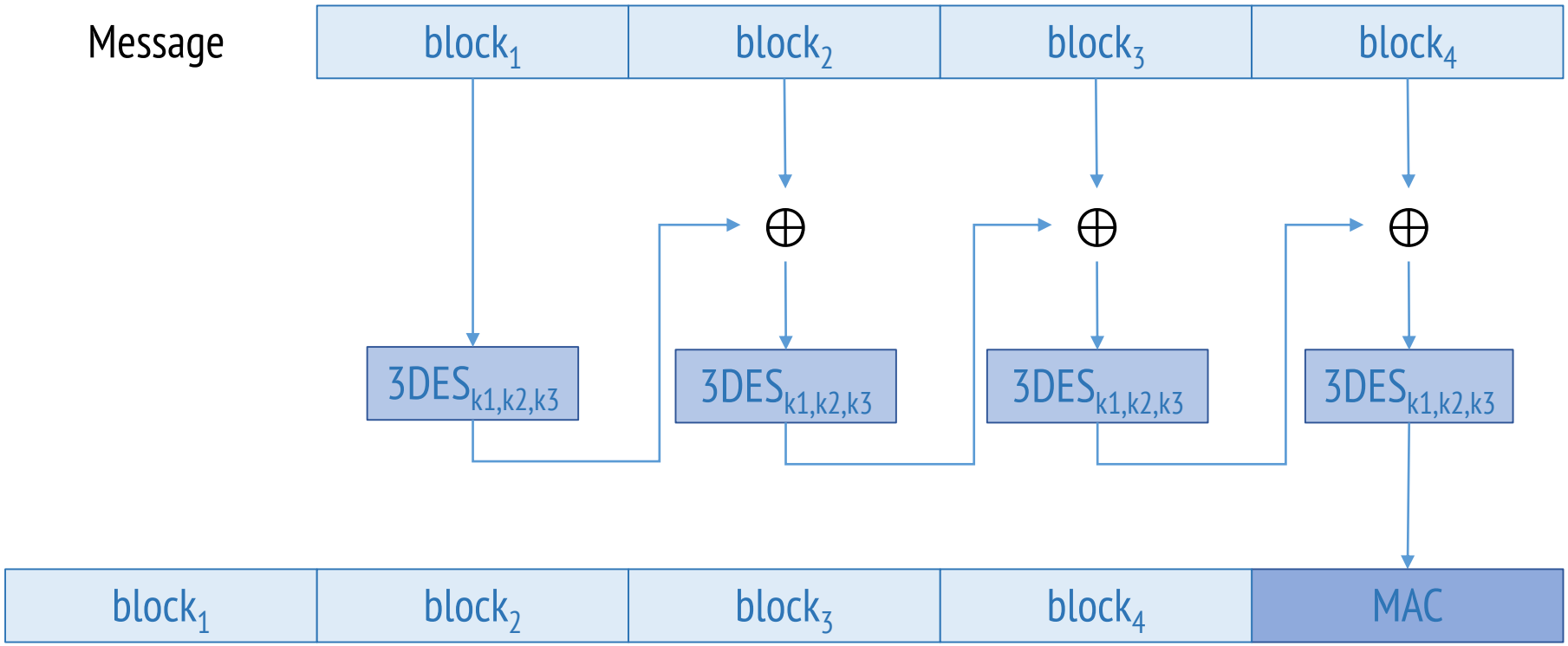
EuroRadio

- Provides authentication for messages on the upper layers
- Based on the ISO 9797-1 MAC Algorithm 3:
  - A CBC circuit which uses a combination of DES and 3DES
  - ISO 9797 padding, i.e. 0s are used as padding until data becomes a multiple of the block size
- Supports priority:
  - Normal priority: messages have a MAC
  - High priority: messages do not require a MAC (e.g. emergency stop messages)

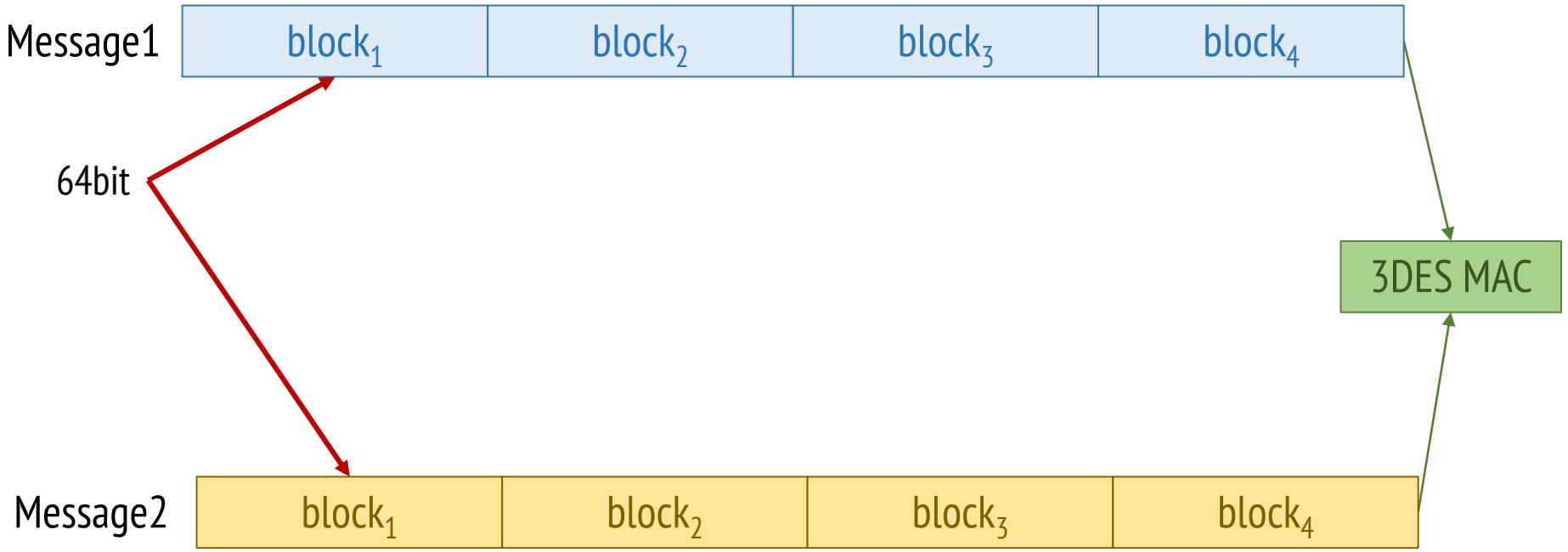
3DES-CBC-MAC


Message
block1
block2
block3
block4
3DESk1,k2,k3
3DESk1,k2,k3
3DESk1,k2,k3
3DESk1,k2,k3
MAC

3DES-CBC-MAC

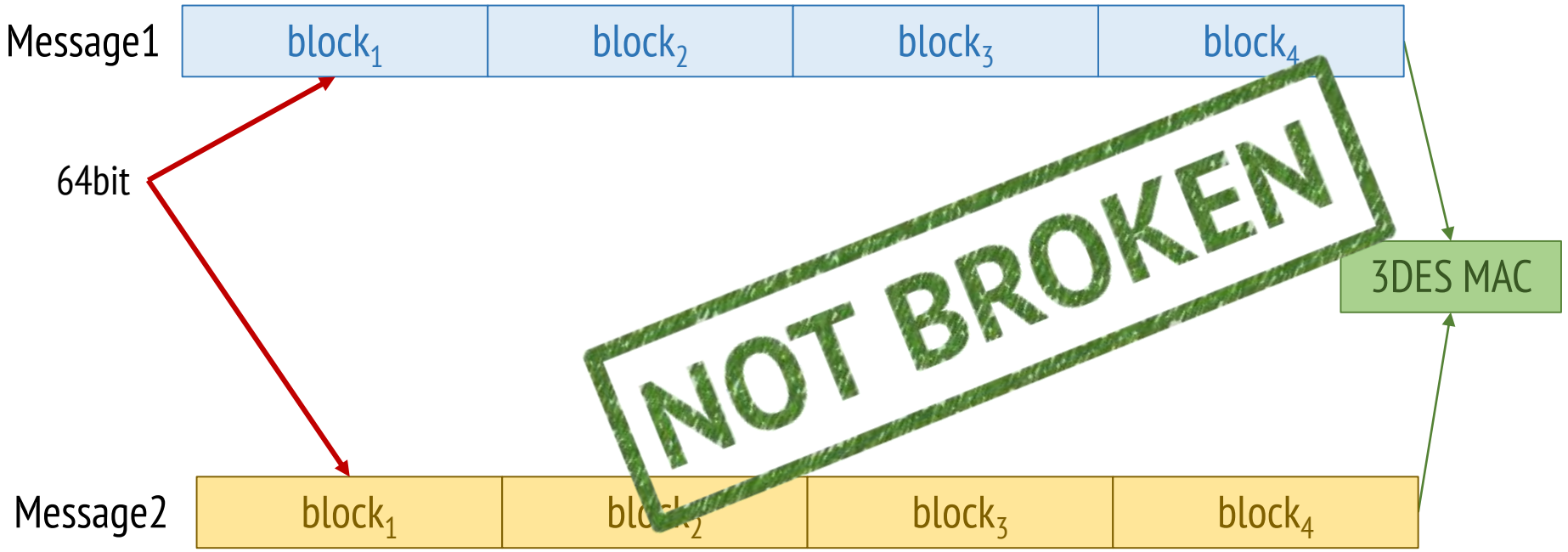

Message
block1
block2
block3
block4
3DESk1,k2,k3
3DESk1,k2,k3
3DESk1,k2,k3
3DESk1,k2,k3
block1
block2
block3
block4
MAC

Collisions in ciphers with small block sizes



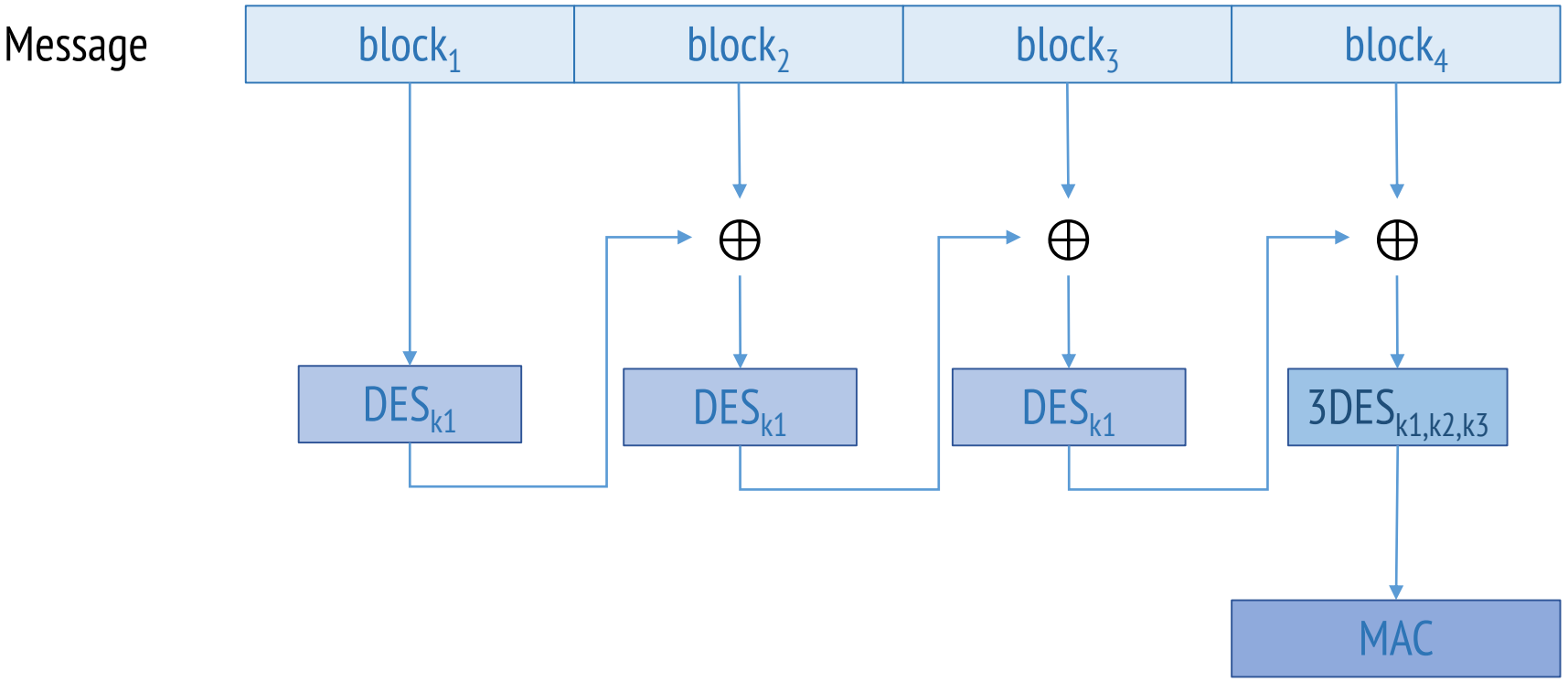
1. B. Preneel and P. C. van Oorschot. Key recovery attack on ANSI X9. 19 retail MAC. Electronics Letters, 1996
2. H. Handschuh and B. Preneel. Minding your MAC algorithms. *Information Security Bulletin*, 2004.
3. Bhargavan, Karthikeyan, and Gaëtan Leurent. "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

Collisions in ciphers with small block sizes

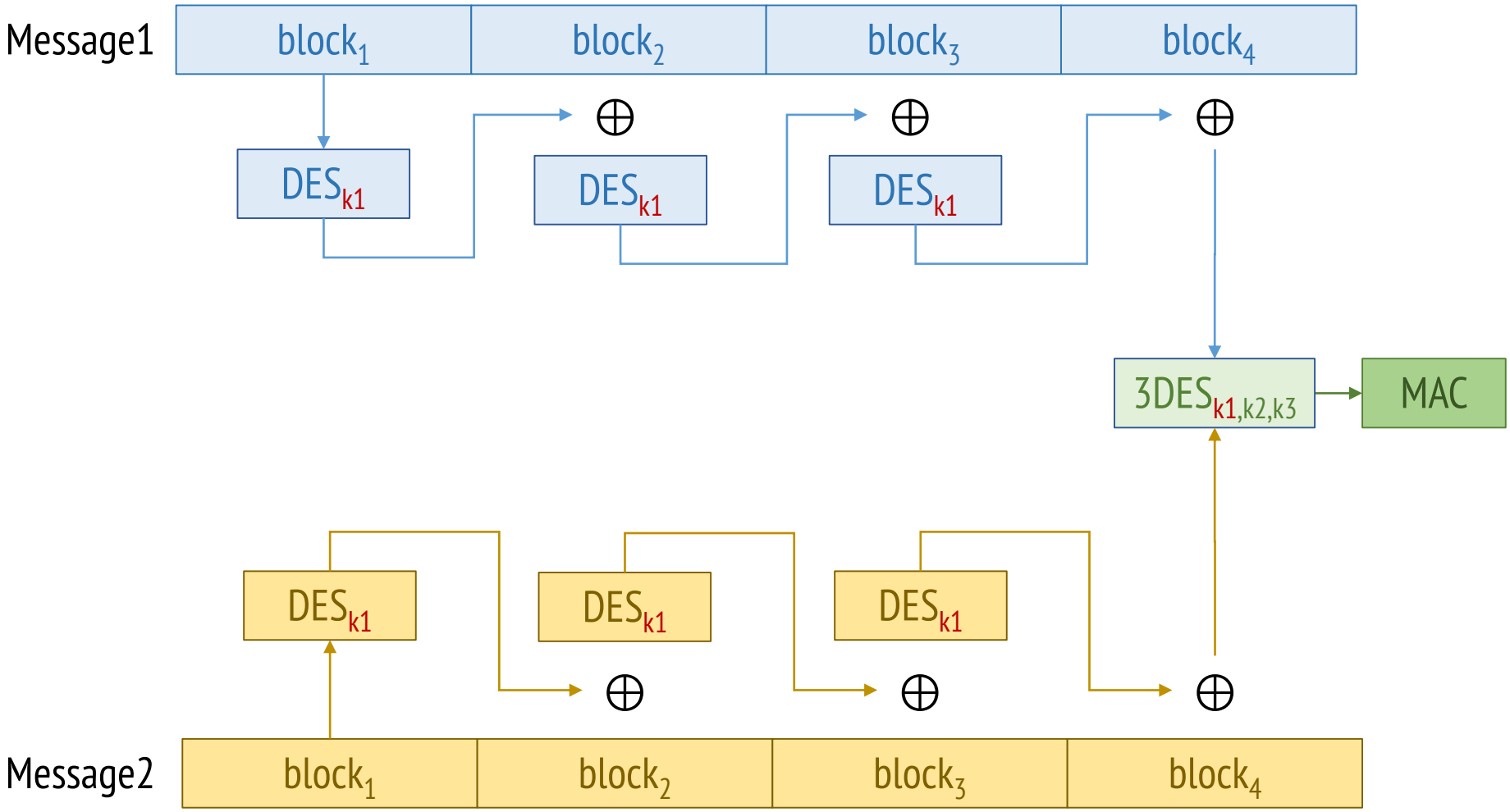


1. B. Preneel and P. C. van Oorschot. Key recovery attack on ANSI X9. 19 retail MAC. Electronics Letters, 1996
2. H. Handschuh and B. Preneel. Minding your MAC algorithms. *Information Security Bulletin*, 2004.
3. Bhargavan, Karthikeyan, and Gaëtan Leurent. "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN."
4. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

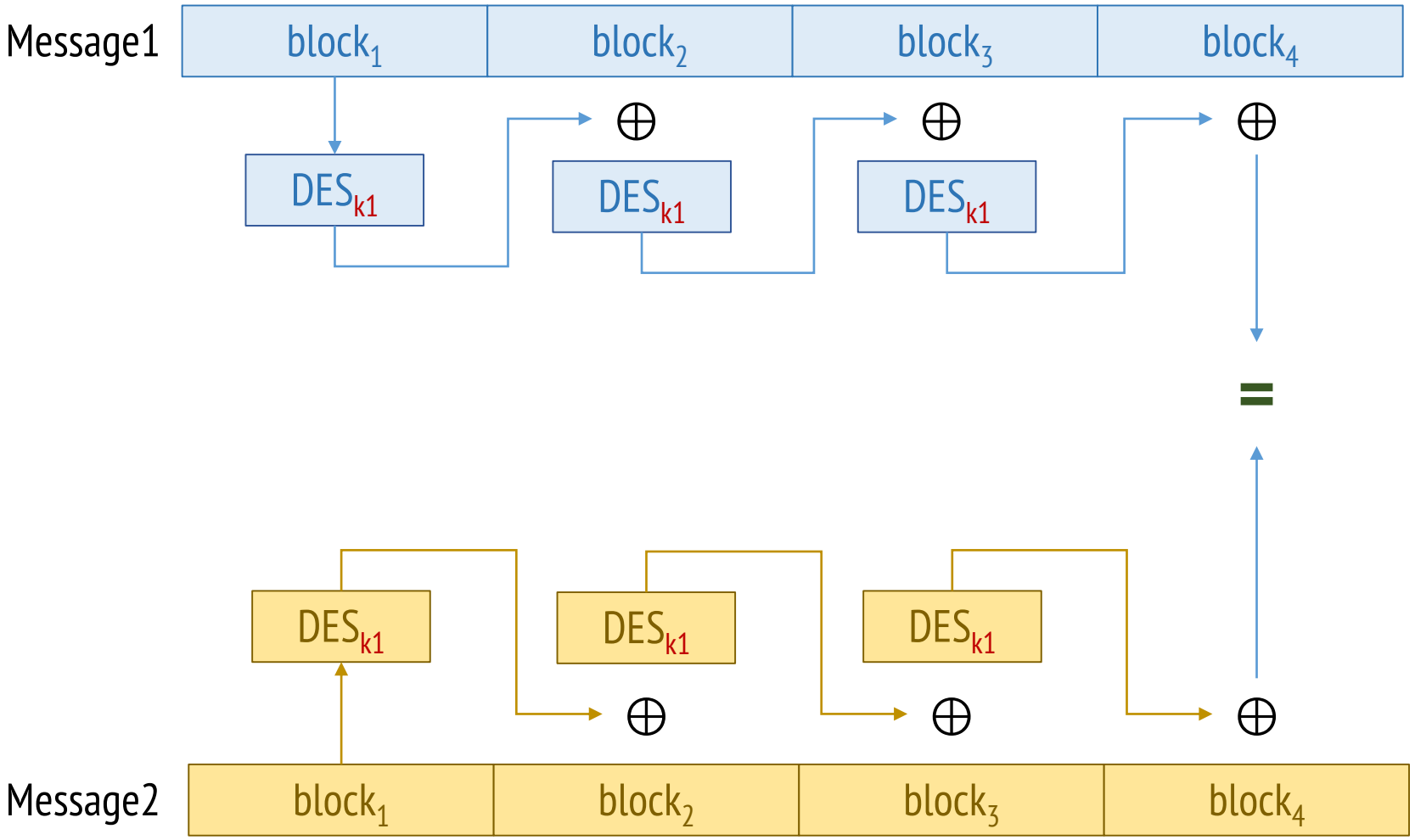
EuroRadio MAC



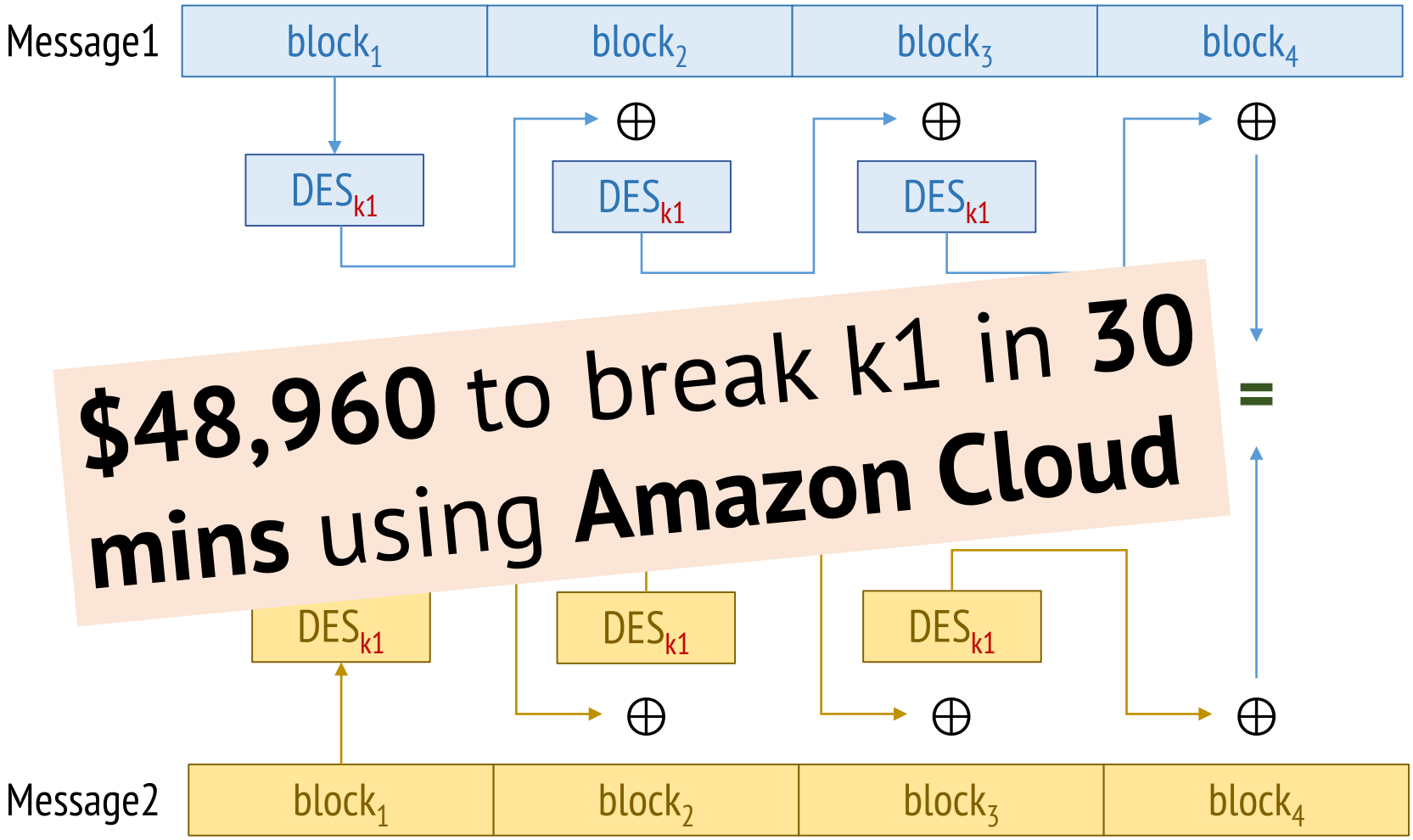
EuroRadio DES key recovery (when a collision happens)



DES key recovery (when a collision happens)



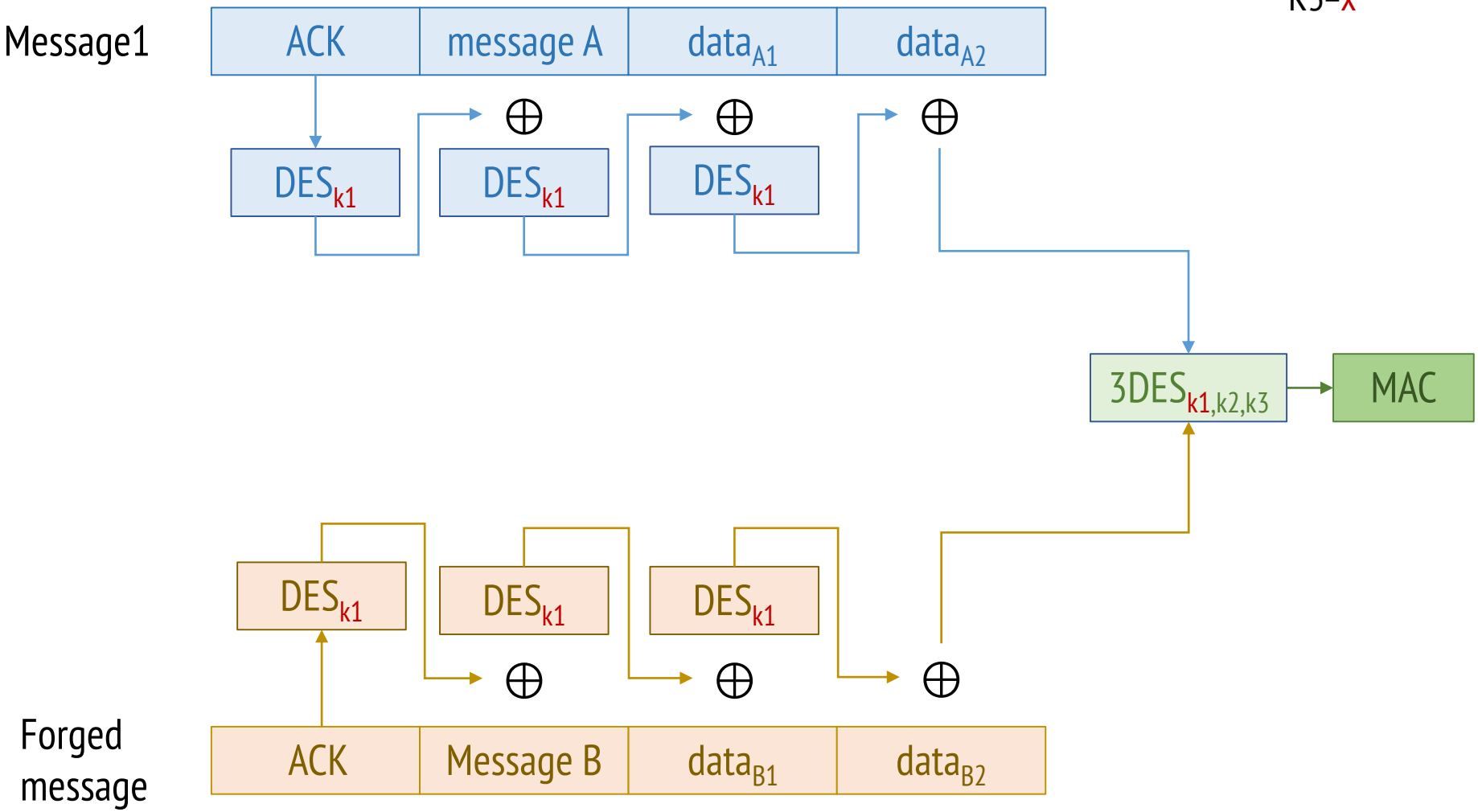
DES key recovery (when a collision happens)



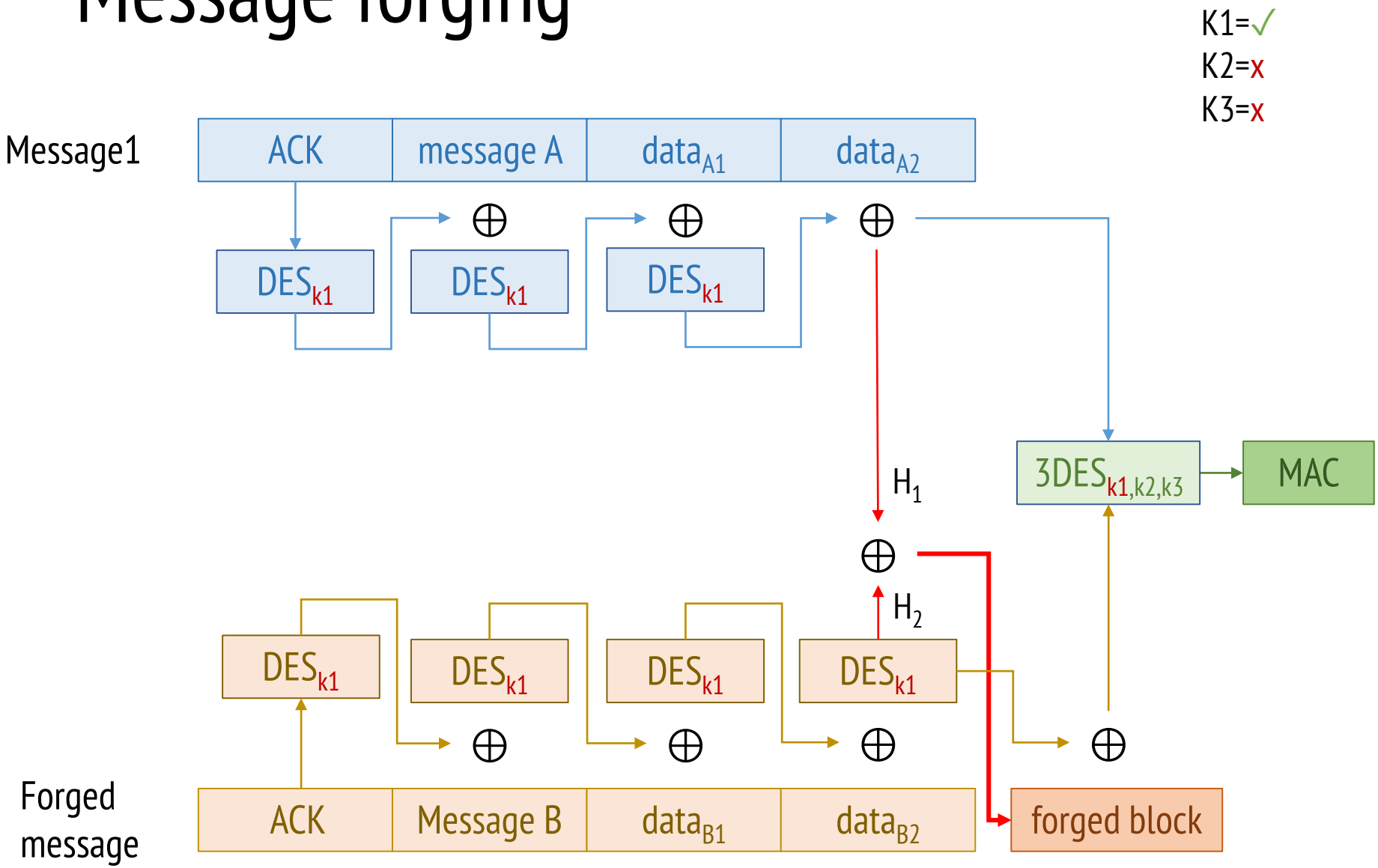
$48,960 to break k1 in **30 mins** using **Amazon Cloud**

Message forging

K1=✓
K2=x
K3=x

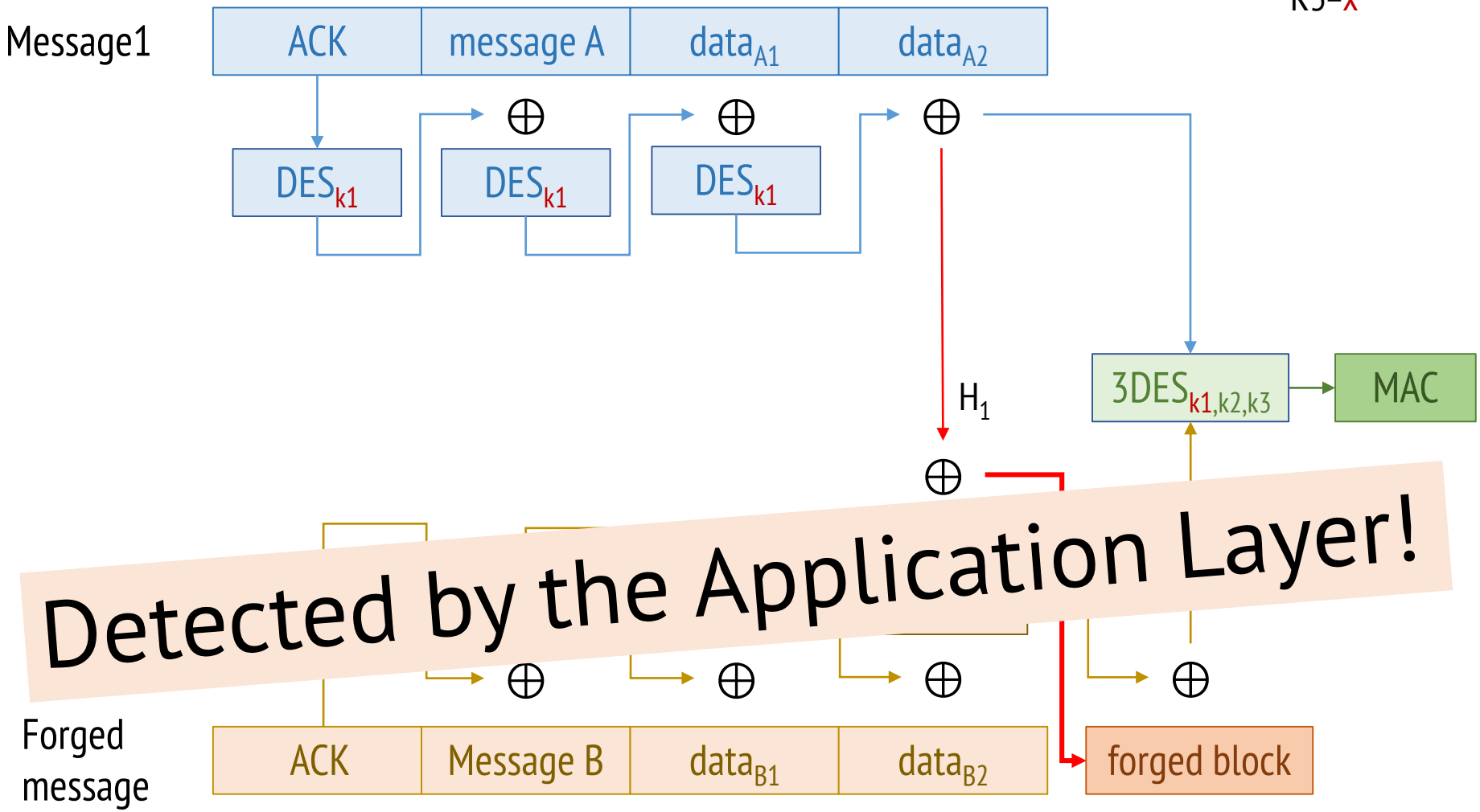


Message forging



Message forging

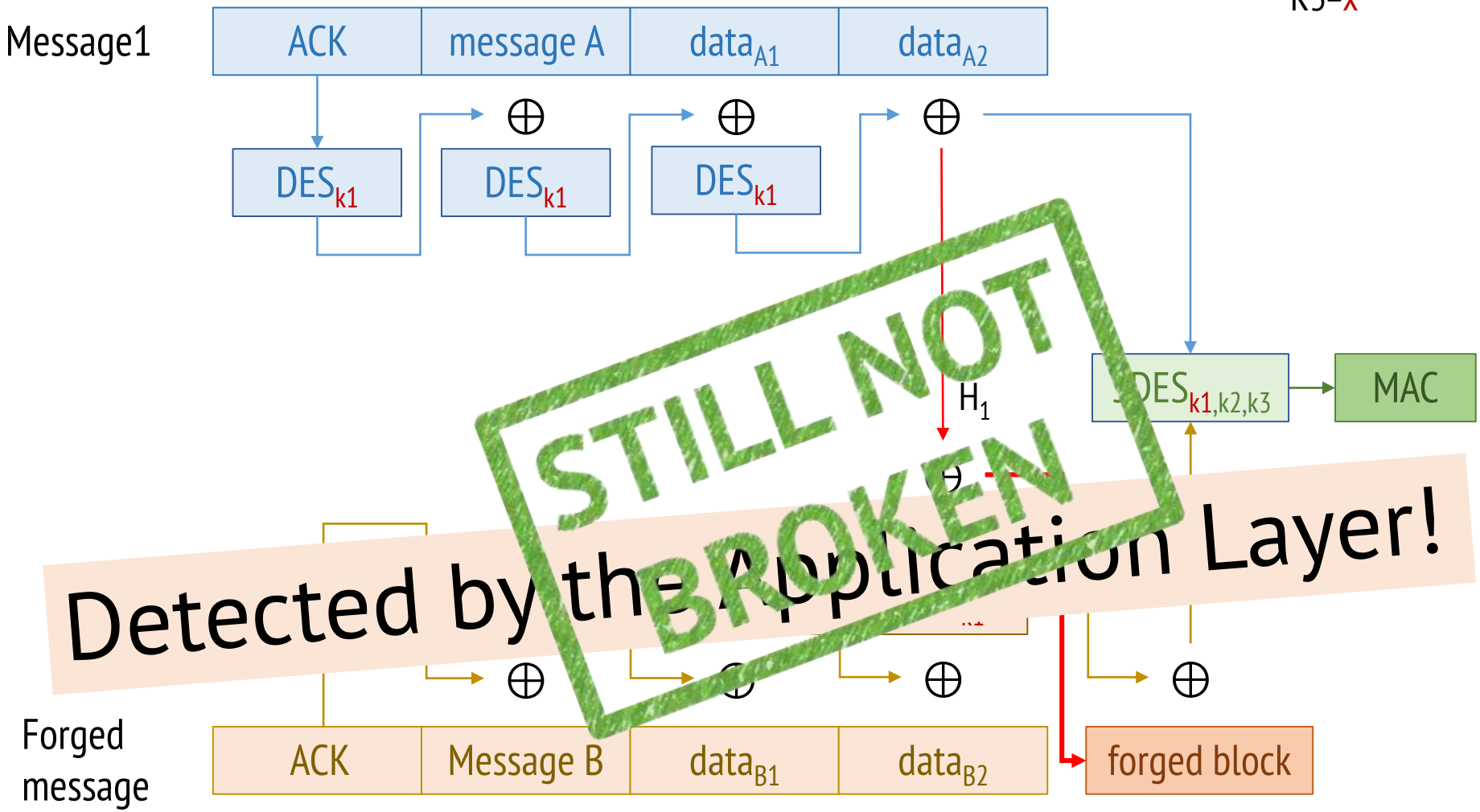
K1=✓
K2=x
K3=x



Detected by the Application Layer!

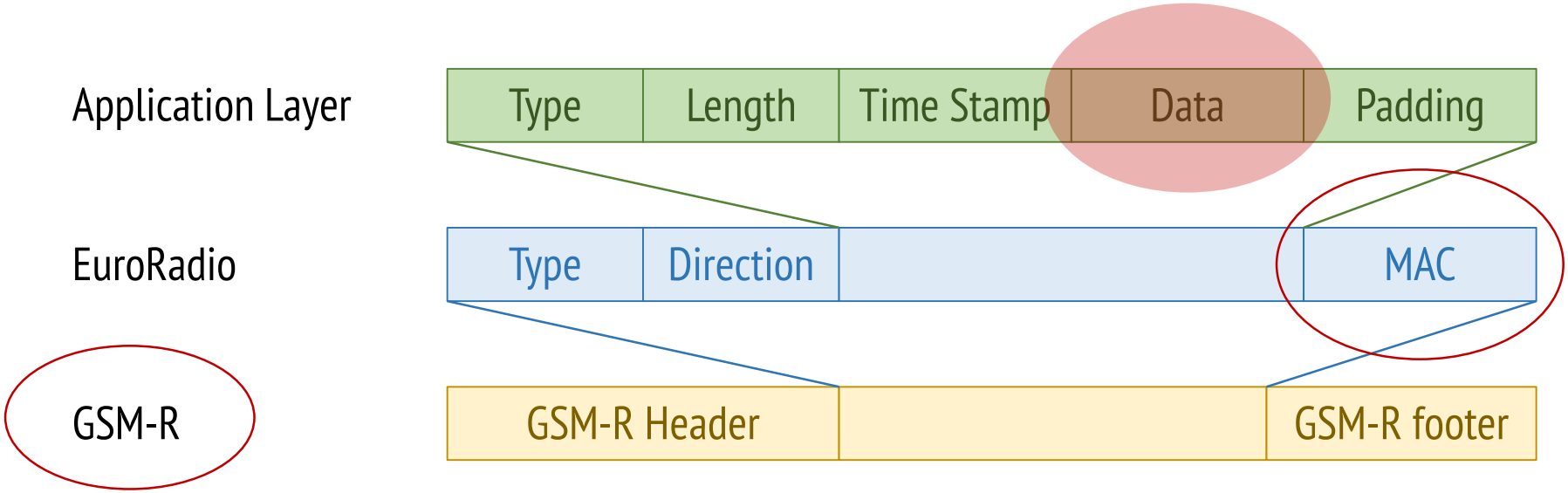
Message forging

K1=✓
K2=x
K3=x



ERTMS stack

K1=✓
K2=x
K3=x

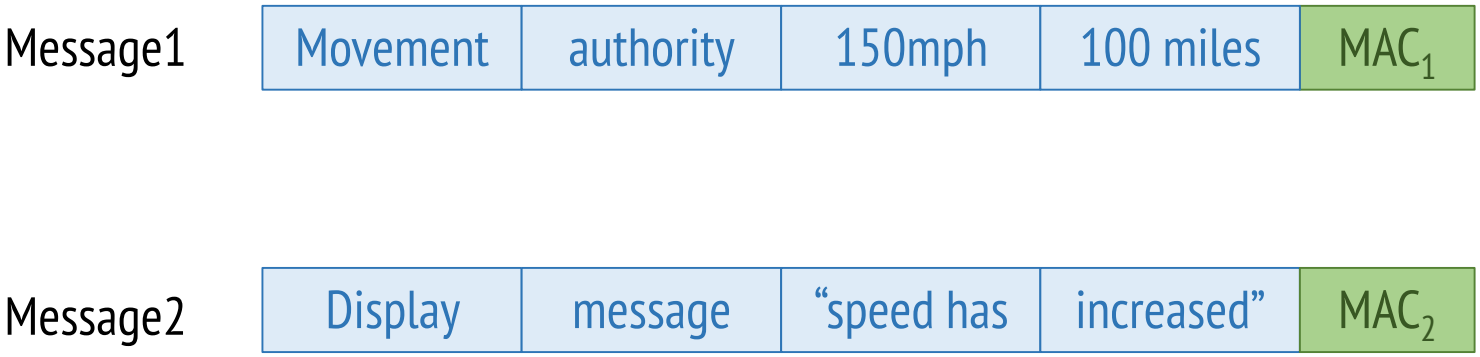


ERTMS - Application Layer

- Transmits train control messages and signalling
- Messages can be of multiple types
  - Movement authorities
  - Display message
  - Acknowledgment message

Application layer

K1=✓
K2=x
K3=x



Application layer

K1=✓
K2=x
K3=x

Message1	Movement	authority	150mph	100 miles	MAC_1
Message2	Display	message	“speed has	increased”	MAC_2

Message1 Message2

Movement	authority	150mph	100miles	Display	message	“speed has	increased”	MAC_3

Application layer

K1=✓
K2=x
K3=x

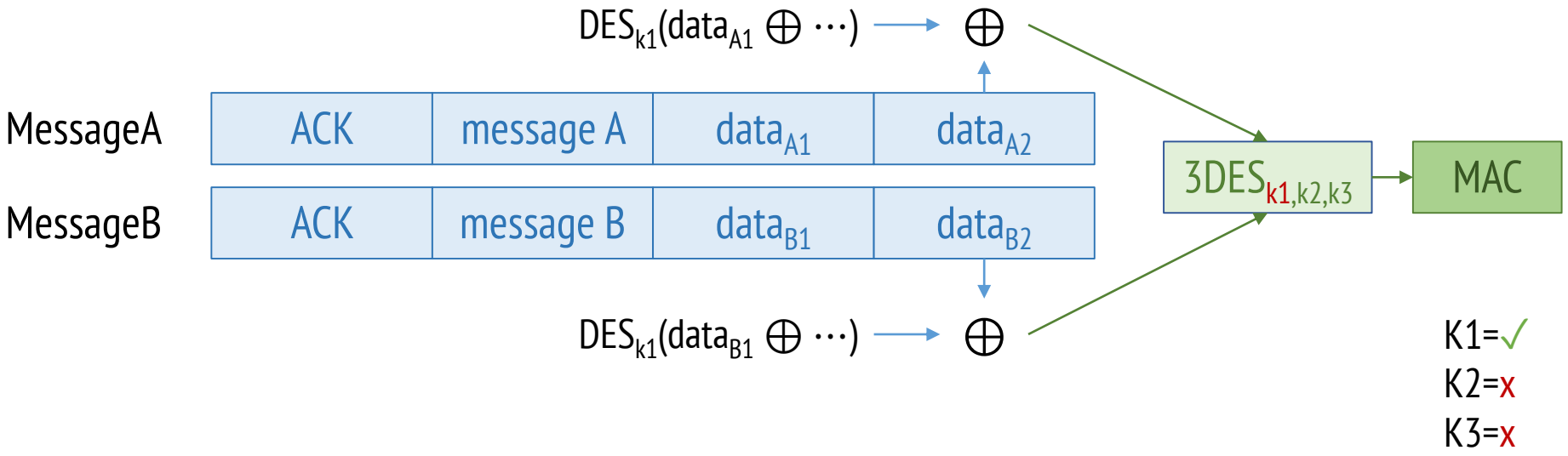
Message1	Movement	authority	150mph	100 miles	MAC_1
Message2	Display	message	“speed has	increased”	MAC_2

Display message accepts Unicode characters!

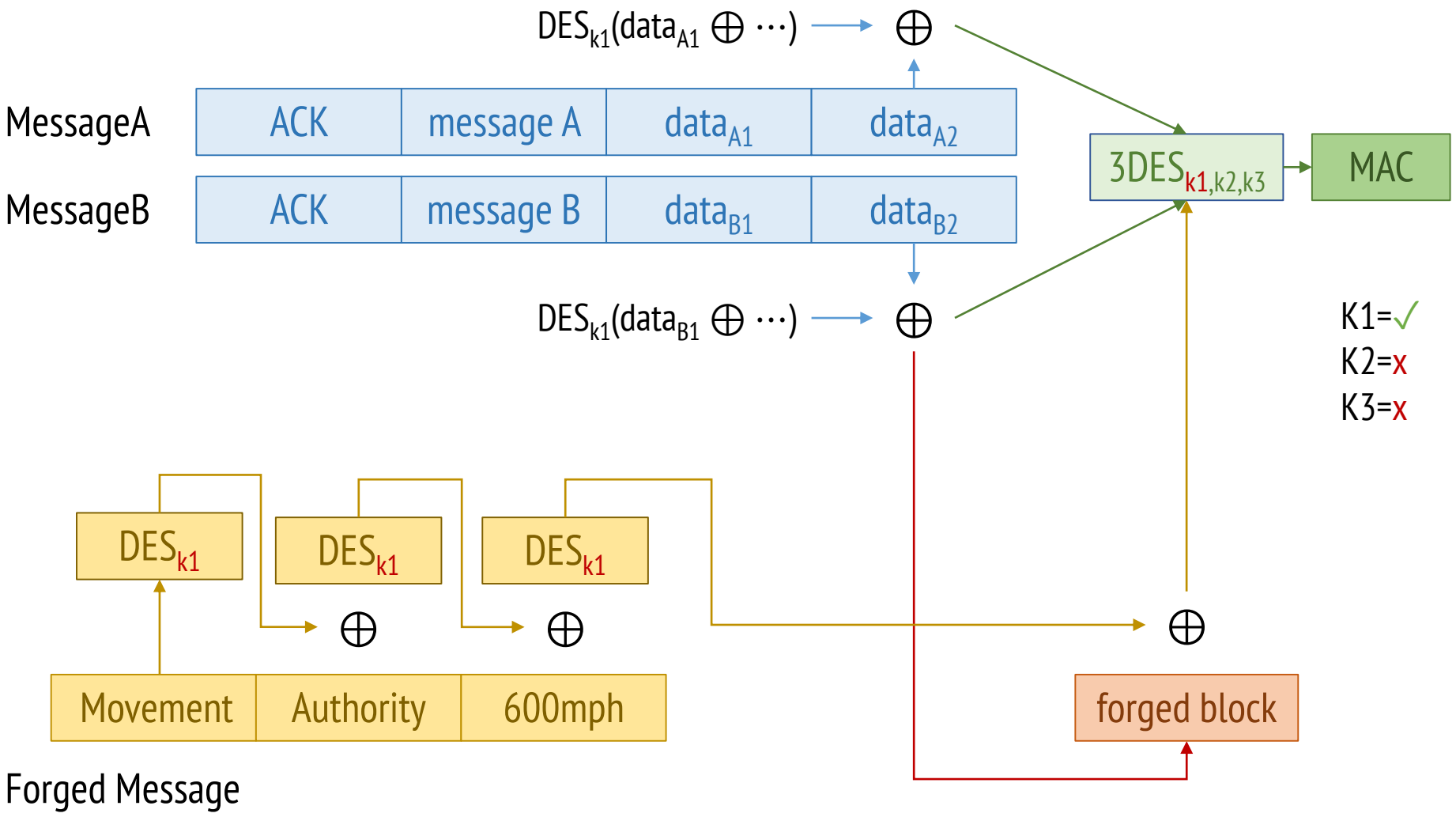
Message1

Movement	authority	150mph	100miles	Display	message	“speed has	increased”	MAC_3

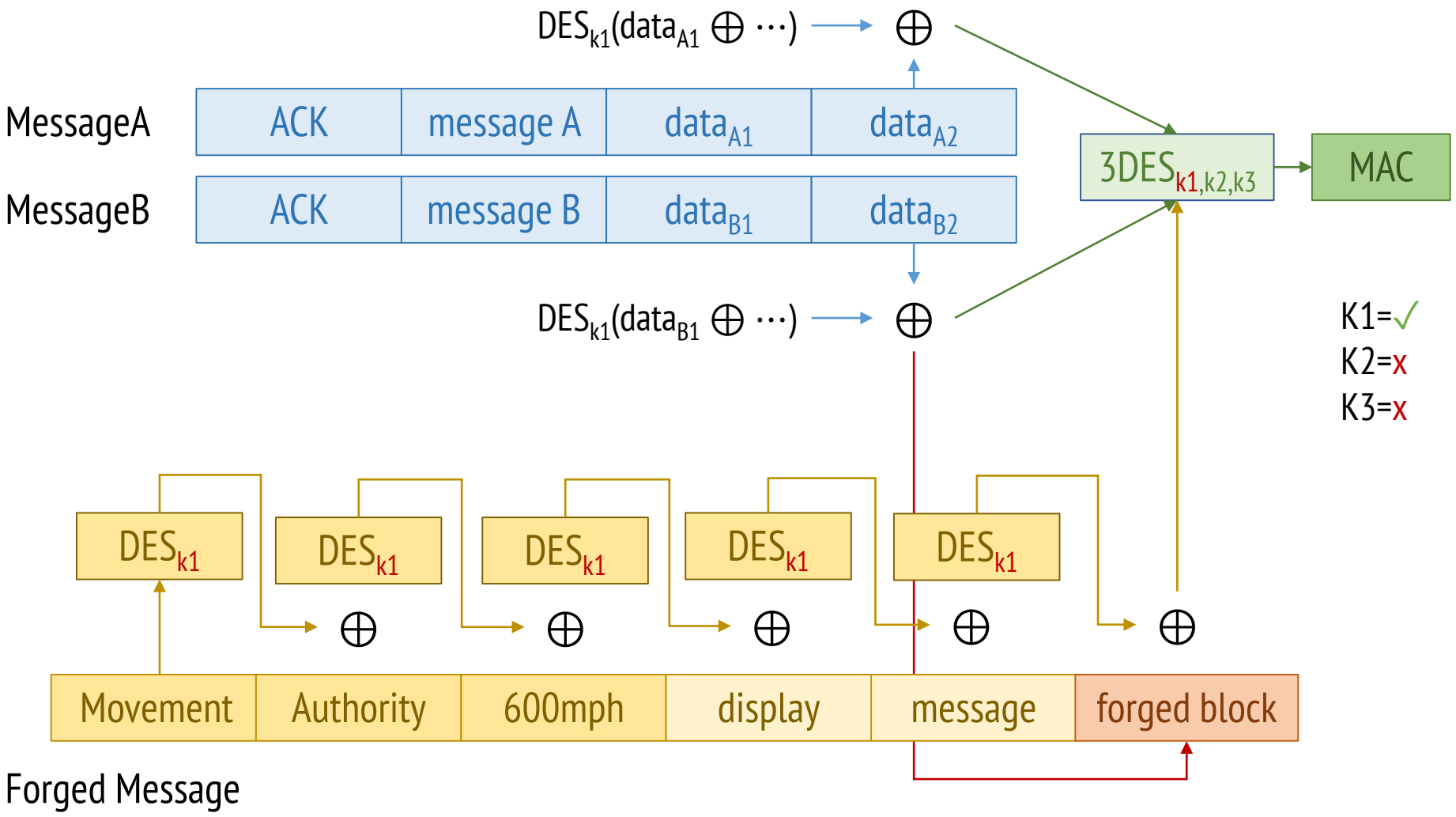
Leveraging collisions



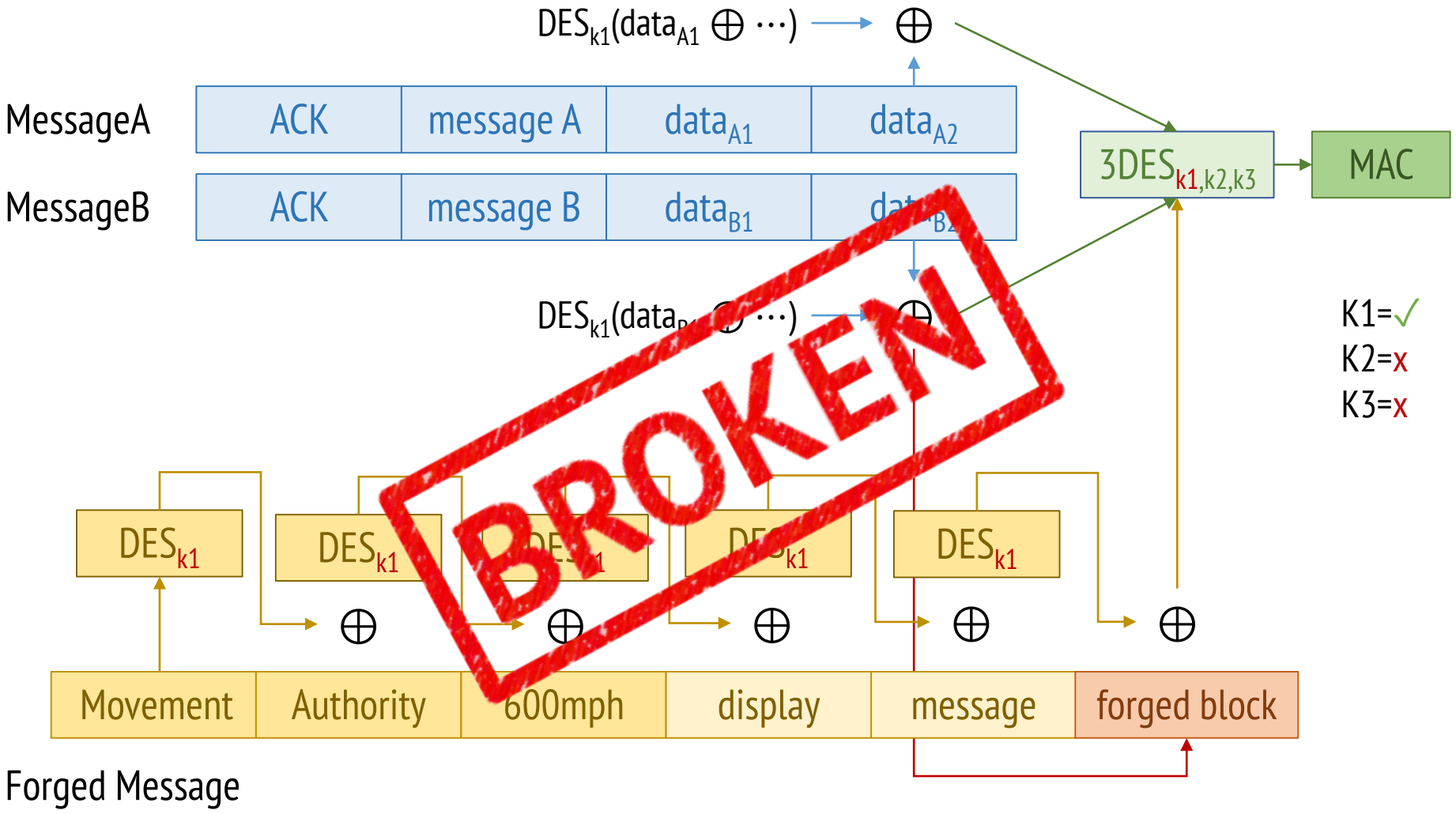
DES key recovery



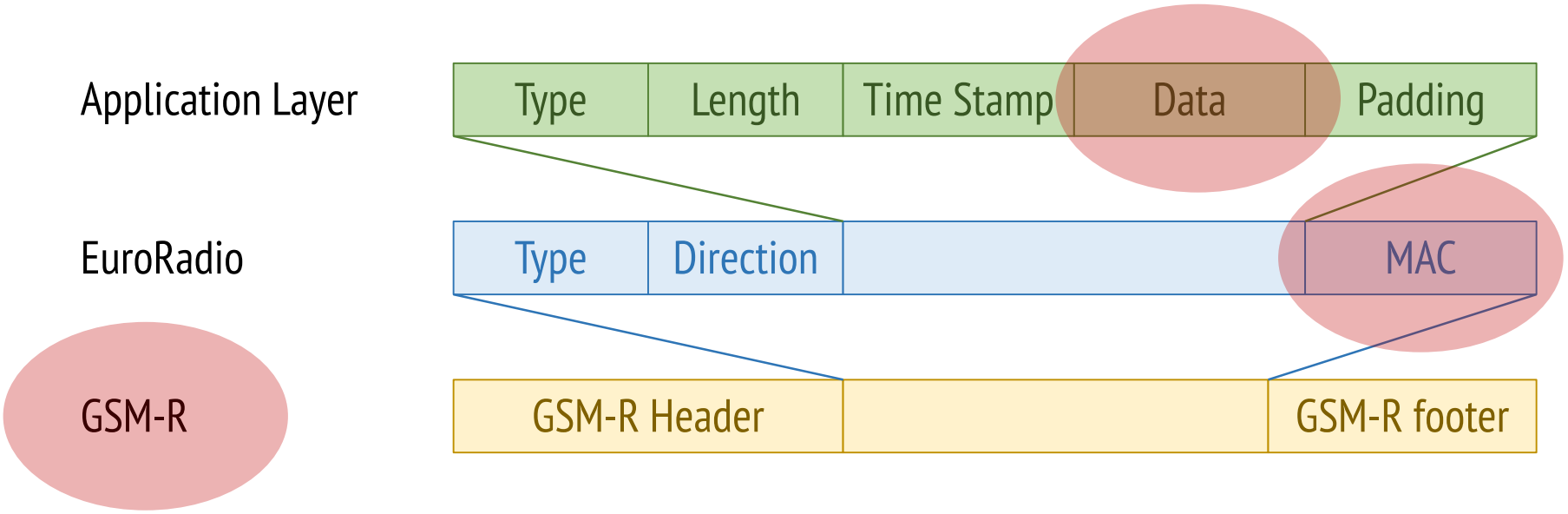
Message concatenation



Message concatenation



ERTMS stack vulnerabilities



ACK message collision

Two acknowledgement messages:

0012000020A9203A2105E048000062105DFD0000000000

MAC: 80B7557F31566DBB

0012000020A9203AAE360078000006AE360000000000000

MAC: 80B7557F31566DBB

Forged movement authority

Variable	Length (bits)	Value	Description
NID_PACKET	8	0000 1111	Level 2/3 movement authority (only RBC)
Q_DIR	2	10	Both directions
L_PACKET	13	0 0000 0111 0001	113 bits
Q_SCALE	2	10	10 m
V_LOA	7	111 1000	600 km/h
T_LOA	10	11 1111 1111	Unlimited
N_ITER	5	0 0000	0 iterations
L_ENDSECTION	15	111 1111 1111 1111	327670 meter
Q_SECTIONTIMER	1	0	No section timer information
Q_ENDTIMER	1	0	No end section timer information
Q_DANGERPOINT	1	0	No danger point information
Q_OVERLAP	1	1	Overlap information to follow
D_STARTOL	15	000 0000 0000 0000	0 meter
T_OL	10	00 0000 0000	0 sec
D_OL	15	000 0000 0000 0000	0 meter
V_RELEASEOL	7	111 1110	Use onboard calculated release speed

Forged display message

Variable	Length (bits)	Value	Description
NID_PACKET	8	0100 1000	Packets for sending plain text messages
Q_DIR	2	00	Reverse
L_PACKET	13	0 0000 1101 1100	220 bits
Q_SCALE	2	10	10 m
Q_TEXTCLASS	2	00	Auxiliary
Q_TEXTDISPLAY	1	0	no, as soon until events fulfilled
D_TEXTDISPLAY	15	111 1111 1111 1110	327660 Meter
M_MODETEXTDISPLAY	4	1001	System failure
M_LEVELTEXTDISPLAY	3	000	Level 0
L_TEXTDISPLAY	15	000 0000 0000 0000	0 Meter
T_TEXTDISPLAY	10	00 0000 0000	0 sec
M_MODETEXTDISPLAY	4	1001	System failure
M_LEVELTEXTDISPLAY	3	000	Level 0
Q_TEXTCONFIRM	2	00	No confirmation required
L_TEXT	8	0001 0000	16 Chars
X_TEXT	128	...	Text messsage...

Encoded messages example

ACK M1: 00120000020A9203A2105E048000006 2105DFD0000000000

MAC: 80B7557F31566DBB

ACK M2: 00120000020A9203AAE360078000006AE360000000000000

MAC: 80B7557F31566DBB

FORGED MSG: continue at 600km/h; display “Z|1MB\%<w*RRf)8n/”

030CD3C677A100000021F01C651FF809C4080000000007E4801
B90FFFD2000000120105A7C314D42253C772A52526629386E2F

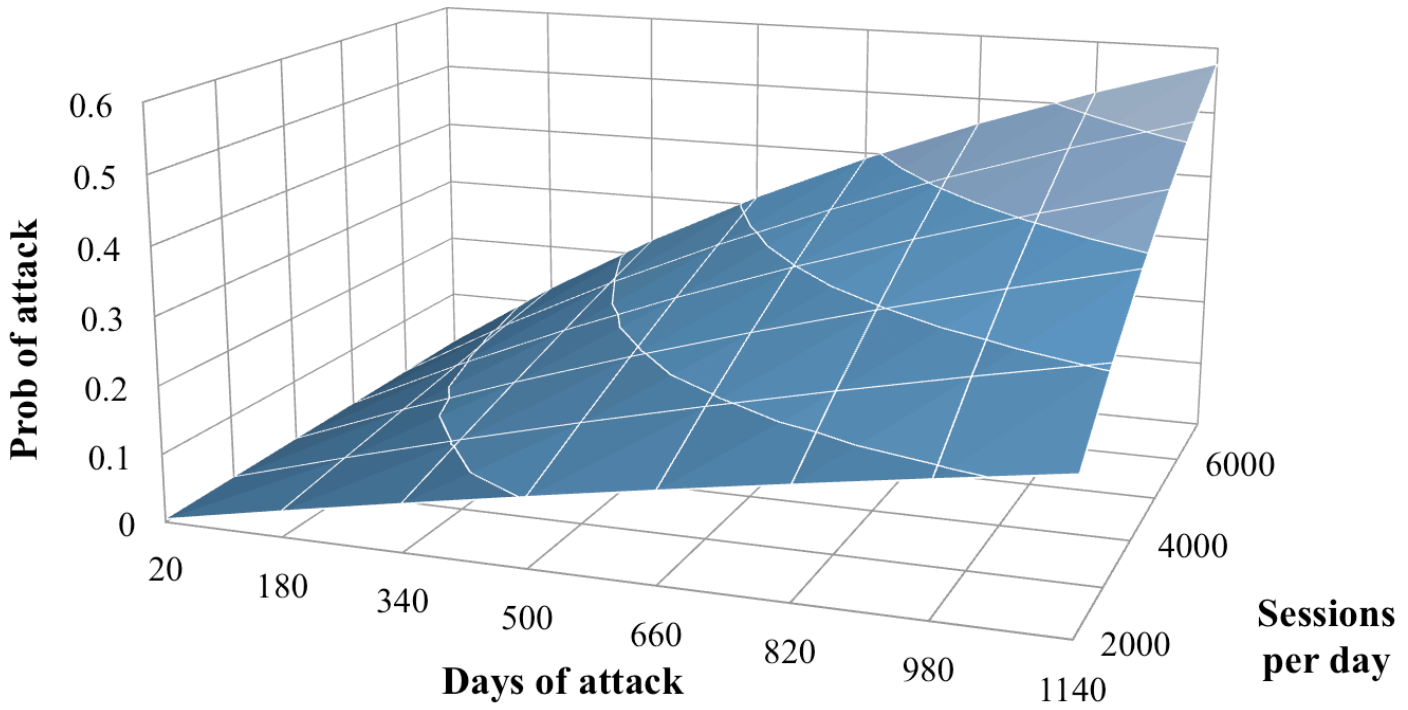
MAC: 80B7557F31566DBB

Impact

1. The attack depends on the ability to discover a collision (assuming that the key used by DES can be brute-forced).

2. Cipher collisions depends on the ability to capture the right amount of traffic.

Impact



$$P_{collision} = 1 - \prod_{i=1}^{M-1}\left(1 - \frac{i}{N}\right)^{S} \approx 1 - e^{\frac{-M(M-1)}{2N}\cdot S}, N = 2^{64}$$

Assumptions

- Message collision chance: 1%
- Average message length 32 bytes
- GSM-R speed 10Kbps (14Kbps max)
- UK rail network:
  - 4000 trains per day
  - 10h sessions

Data capture

- 1% chance of collision requires ~600,000,000 messages.
  - *32 byte messages, 10 Kbps bandwidth*
- Safe limit for a EuroRadio session: 19 GB.
- This would require a single session lasting 22 days!
- No threat to current trains.

Data capture

- 1% chance of collision requires ~600,000,000 messages.
  - *32 byte messages, 10 Kbps bandwidth*
- If we could monitor next generation UK rail backbone(s).
  - 4000 trains per day, 10 hour sessions.
- 1% chance of attack in 45 days. 50% chance ~ 8 years
  - This *might* be a problem.

ERTMS conclusion

- Defence in depth did not help, there were issues on every protocol layer.
- The specification fails to meet safety standards.
- Poorly designed: ERTMS is the next gen in rail technology, but has been designed with obsolete ciphers.

Payment protocols

Payment protocols

- PayWave(Visa)/PayPass(MasterCart) contactless protocols
- ApplePay protocol

Purpose of a payment protocol

1. Provide evidence to the shop reader that the card is genuine.

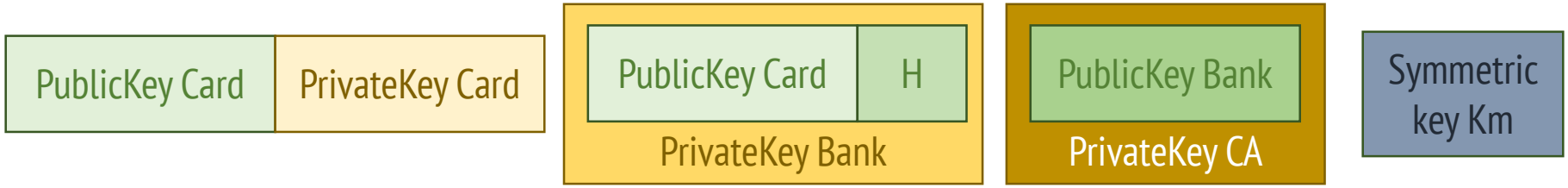
2. Give the shop reader a cryptogram (i.e., AC) which it can sent to the bank as proof that a payment is due.

PayWave/PayPass protocols

- Based on the contact-based EMV standard
- 7 different payment protocol variations
  Visa's PayWave and MasterCard's PayPass are the most popular
- Not compatible with each other, but use similar commands

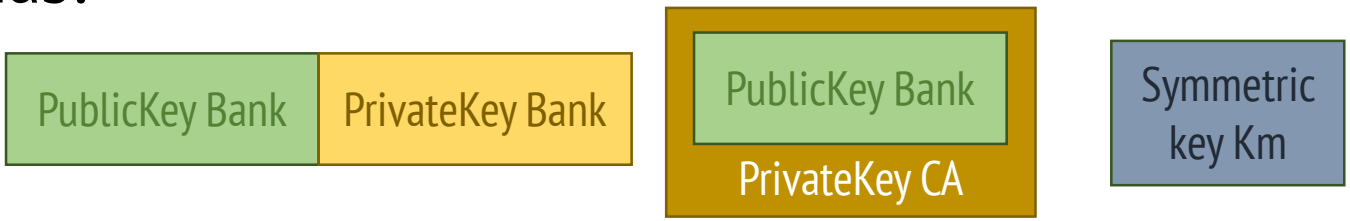
Cryptographic elements of the protocol

Card has:

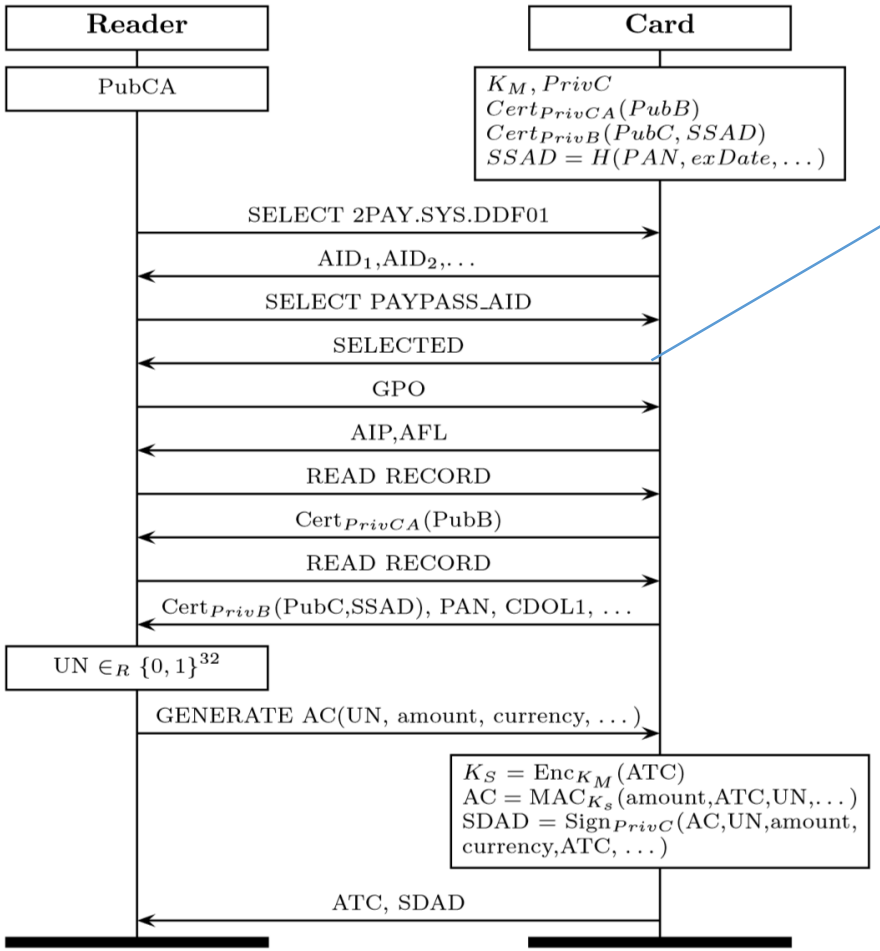


Card reader has:

Banks has:



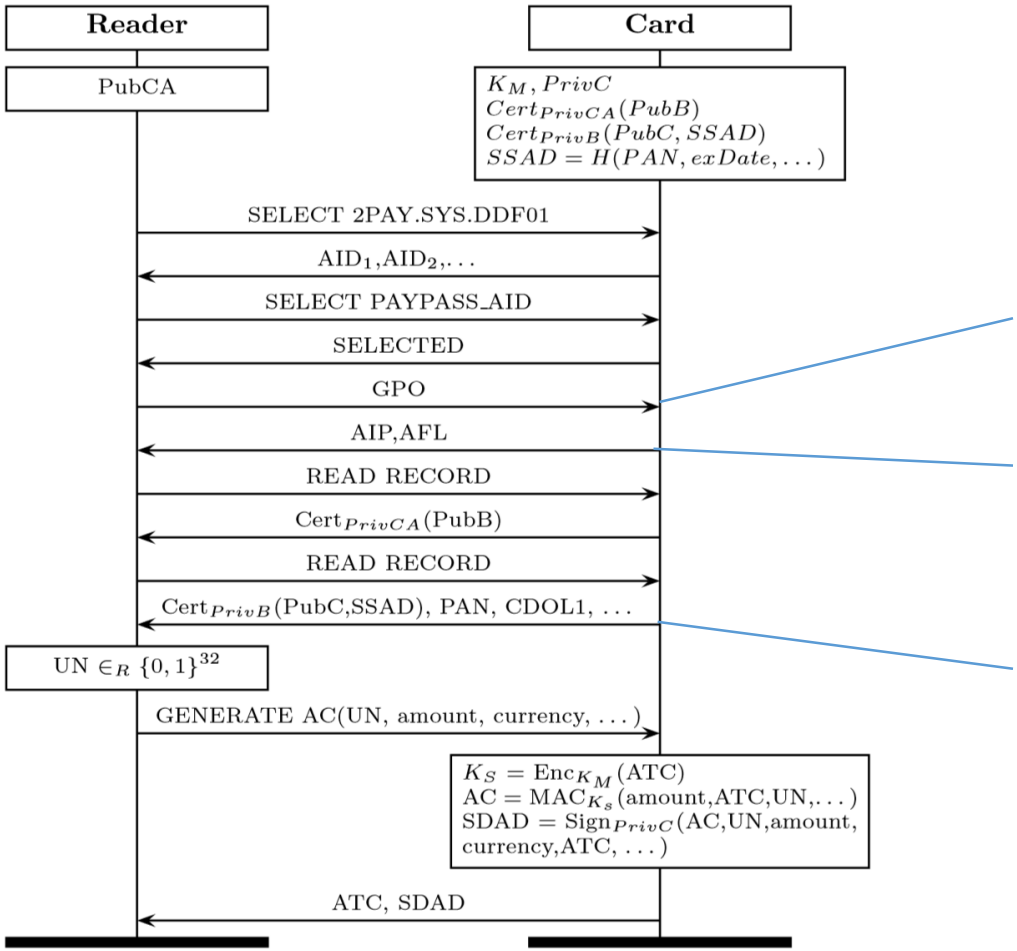
MasterCard PayPass protocol



1. Shop: select payment application
2. Card: Present supported applications identities per protocol
3. Shop: select protocol
4. Acknowledge

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

MasterCard PayPass protocol



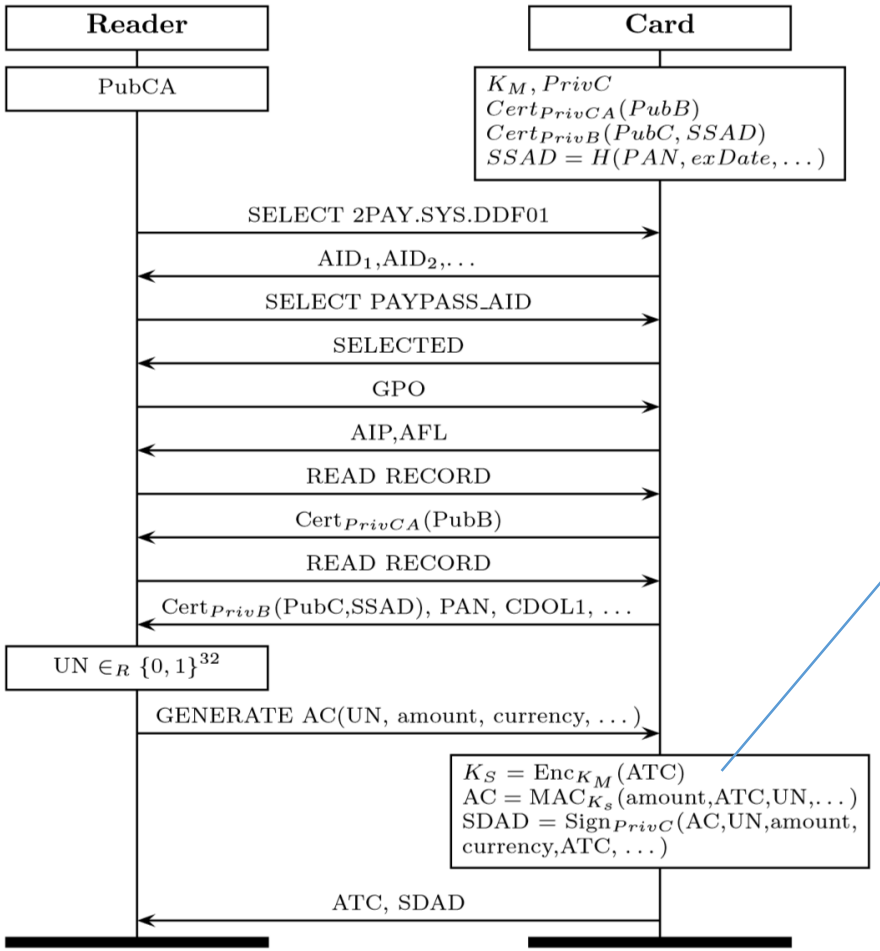
Req: Get processing options (GPO)

Rply: Supported functions (AIP) and their location on card (AFL)

Reader requests fields form the Card. Number of steps is indicated in AFL.

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

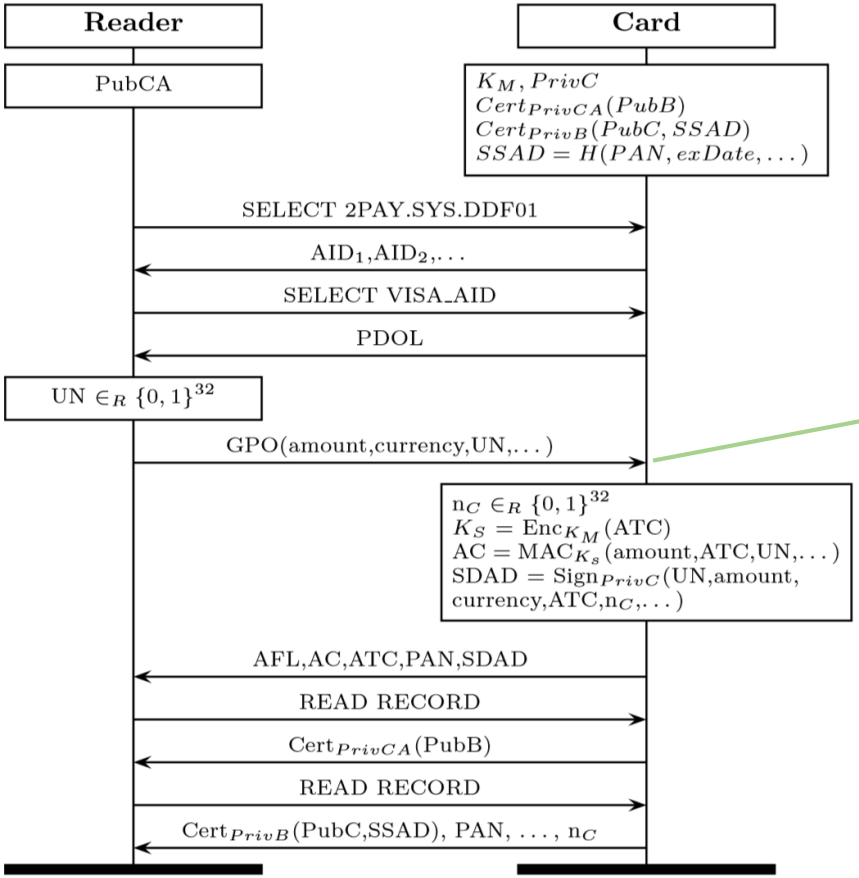
MasterCard PayPass protocol



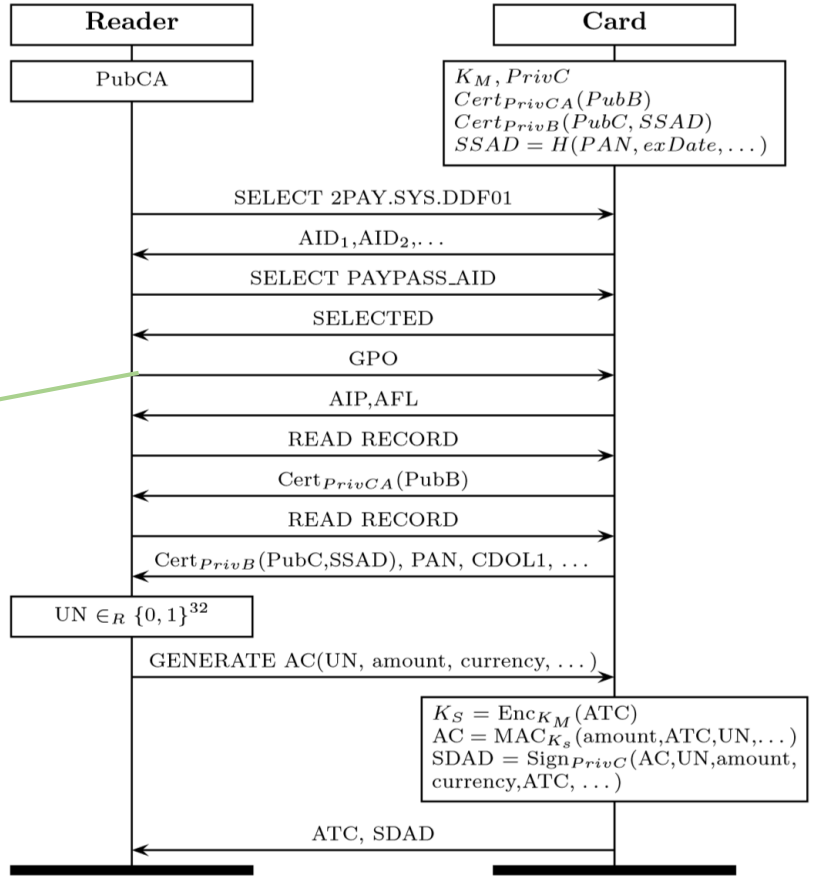
R: Generate nonce
Request cryptogram (AC)
C: Generate Ks from **Km and ATC**
Generate cryptogram for bank using Ks (AC)
Generate verifiable msg. for the Reader (SDAD)

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Visa PayWave protocol



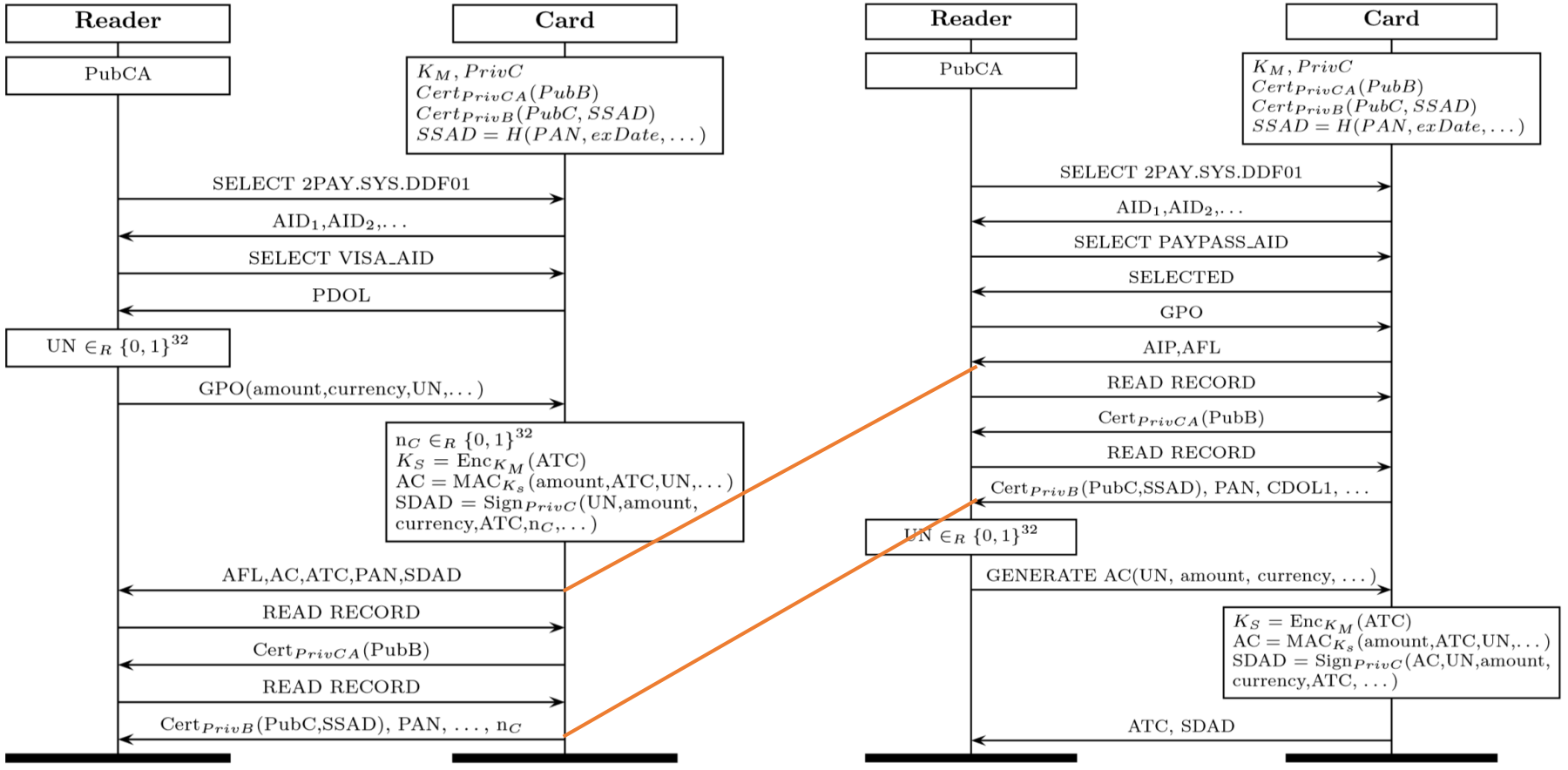
Visa payWave



MasterCard PayPass

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Visa PayWave protocol

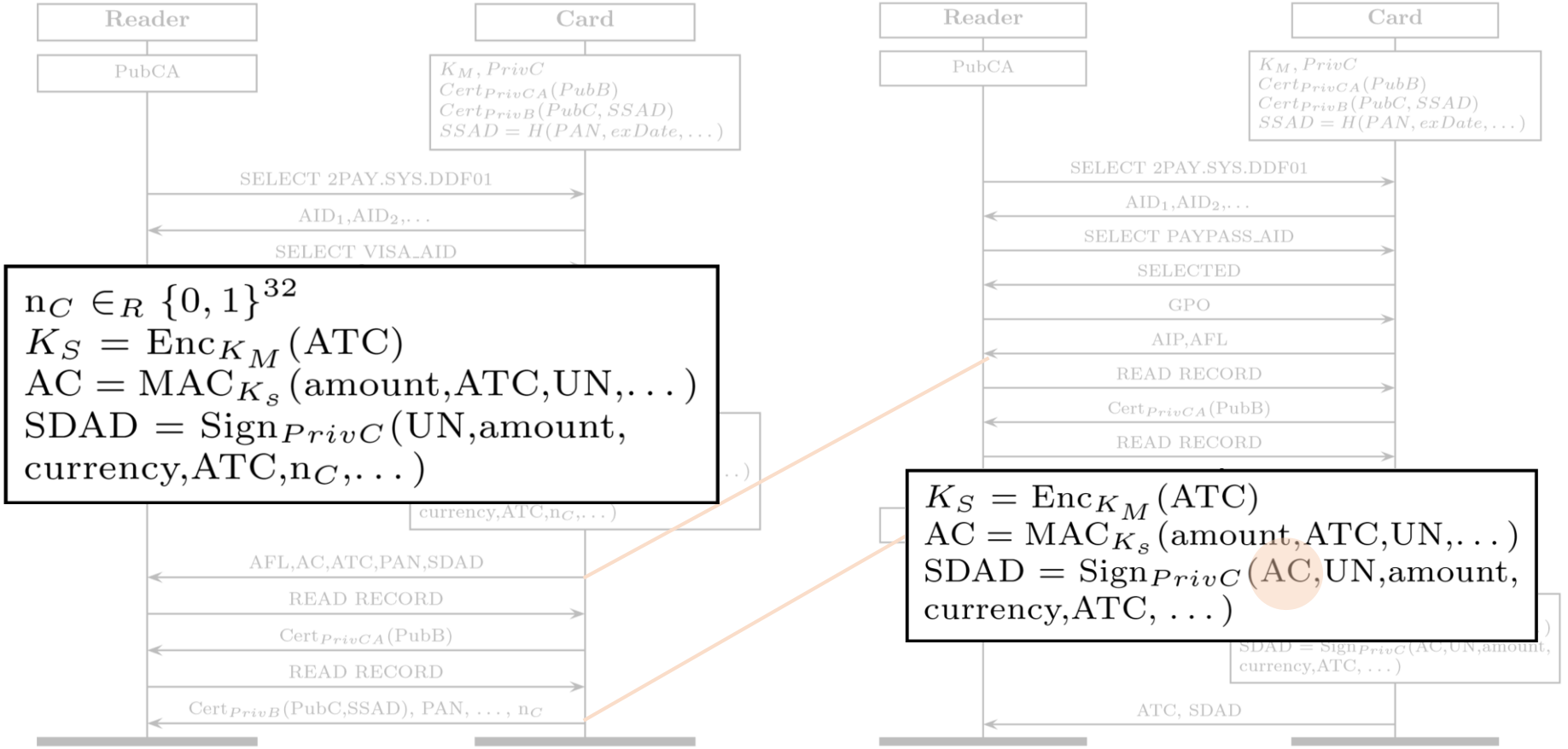


Visa payWave

MasterCard PayPass

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Visa PayWave protocol

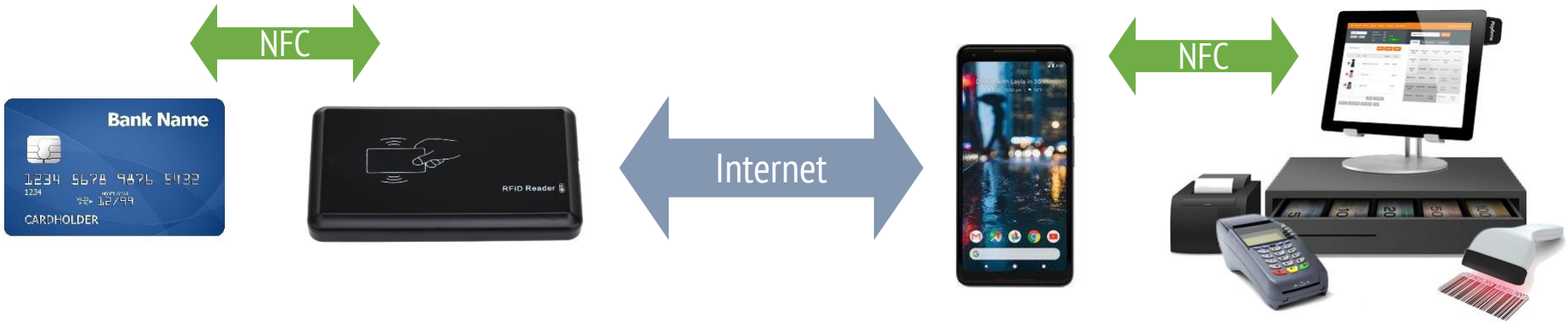


Visa payWave

MasterCard PayPass

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Relay attacks against EMV Contactless Smart Cards

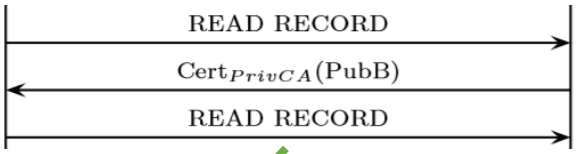


Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

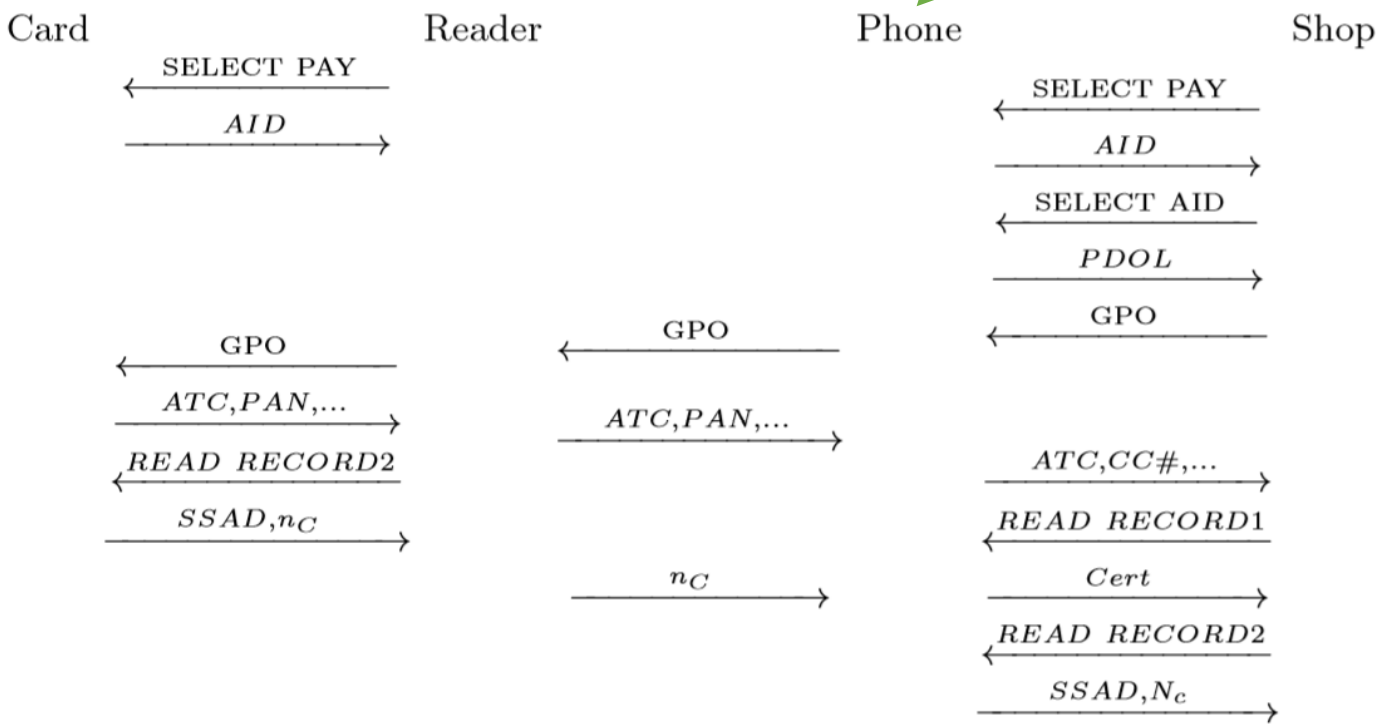
Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Relay attacks

1. Cache static data from card



2. Relay data to/from card



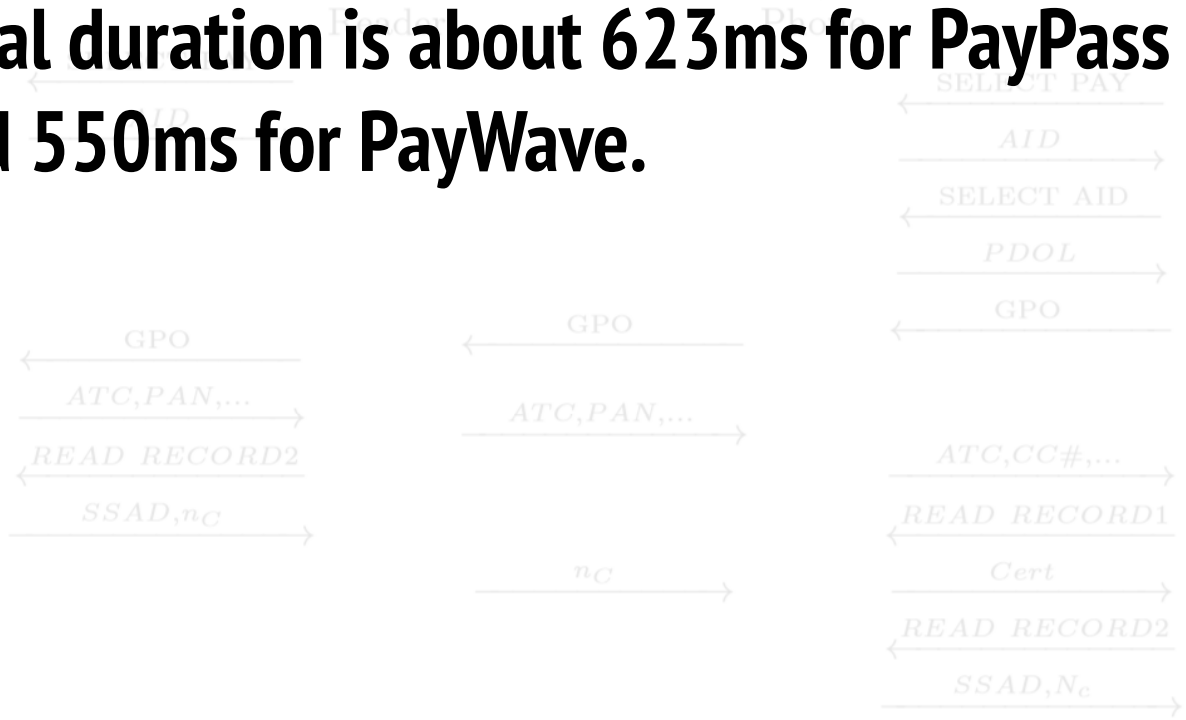
Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Relay attacks

1. Cache static data from card

2. Relay data to/from card

Total duration is about 623ms for PayPass and 550ms for PayWave.



Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Fixing the protocol

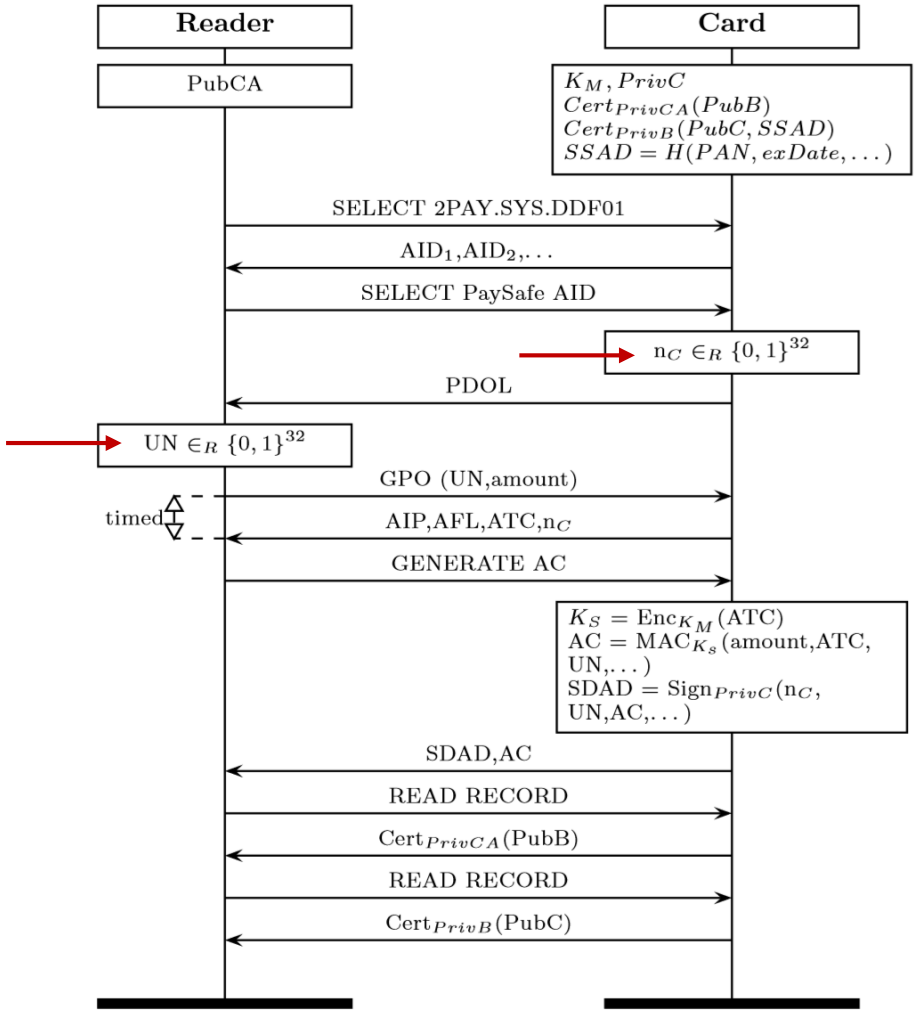
Challenges

1. The relay is very fast.
2. Cards and card positioning on the reader vary a lot and these affect protocol running time.
3. Find the right message exchange to measure.

Solution

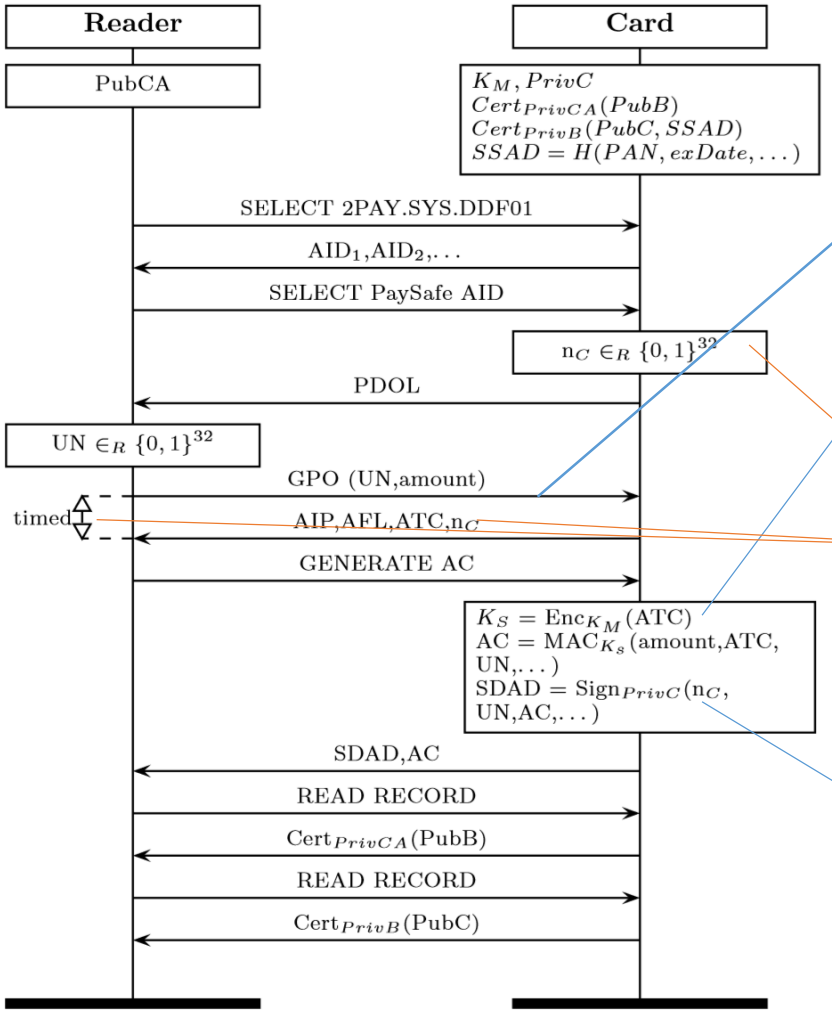
Split the challenge and response command from the generation of the signed authentication (SDAD) and the cryptogram (AC).

Fix: The PaySafe protocol



Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Fix: The PaySafe protocol



Provide the readers nonce in the GPO message.

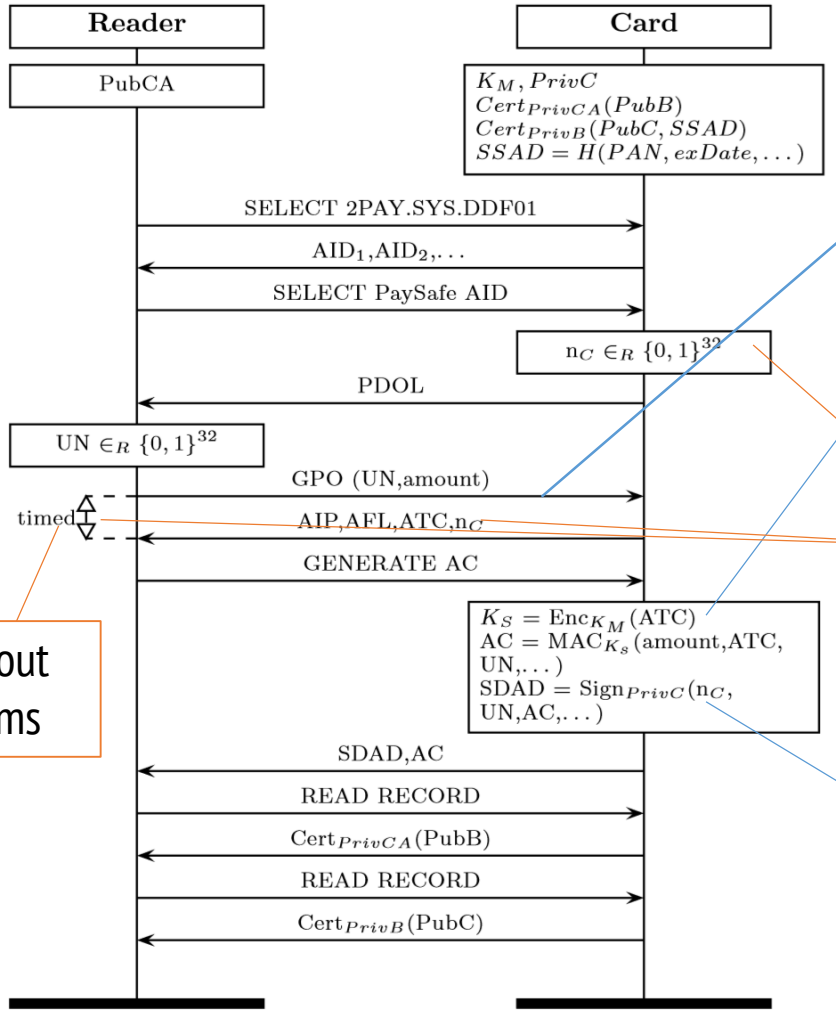
Delay the generation of the AC and SDAD until after the Generate AC is received.

Offset card nonce generation and sending. This enables consistent measurements of the time between when the GPO message is sent and then the reply is received.

Include both nonces an the AC in the signed message

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Fix: The PaySafe protocol



About 80ms

Provide the readers nonce in the GPO message.

Delay the generation of the AC and SDAD until after the Generate AC is received.

Offset card nonce generation and sending. This enables consistent measurements of the time between when the GPO message is sent and then the reply is received.

Include both nonces an the AC in the signed message

Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel and Matthew Thompson. **Relay Cost Bounding for Contactless EMV Payments.** In *19th Financial Cryptography and Data Security (FC 2015).* Lecture Notes in Computer Science, Vol. 7459, pages 189-206. 2015

Relay attacks against EMV

- Relay attacks work against PayWave and PayPass.
- Financial incentives are small (maximum transaction amount is ~30GBP).

- Time bounds the protocol to prevent relay attacks.
- Ensures that no caching was done by including card and reader nonces.
- Includes the cryptogram in the signed message to allow the “shop” to verify the transaction.

Mobile pay protocols

Components

- Secure element (SE)
- NFC controller
- Secure Enclave (SEP, TrustZone, etc.)
- Wallet App
- pay Servers

Secure element (SE)

“**Secure Element (SE)** is a tamper-resistant platform capable of securely hosting applications and their confidential and cryptographic data in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities.” GlobalPlatform

Secure element (SE)

SE in mobile payments.

During a mobile payment the SE is used to emulate a contactless card using industry standard protocols.

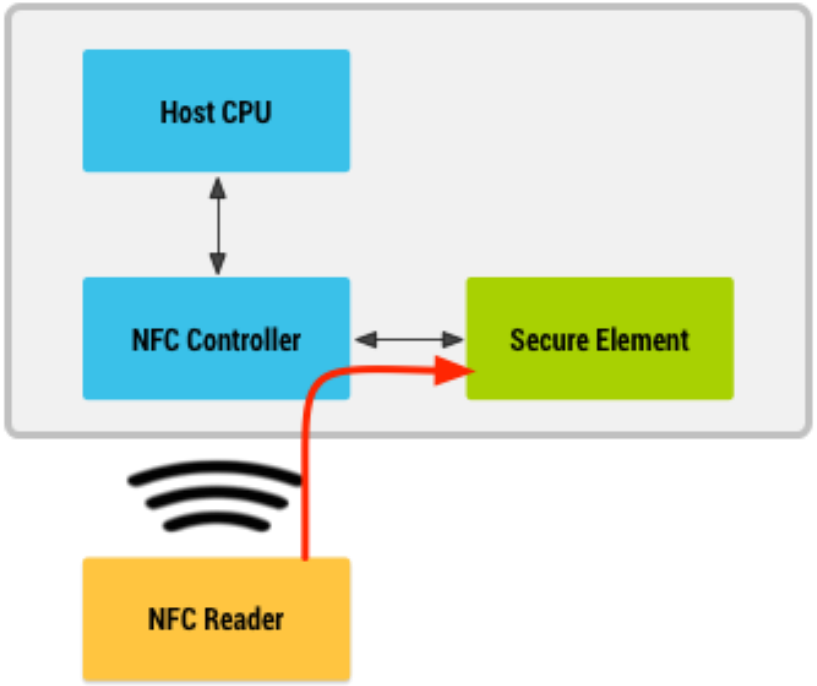
The SE can be:

Embedded in the phone

Embedded in the SIM card

A cloud based service (Google HCE)

Simple protocol (Google Wallet v1.0)

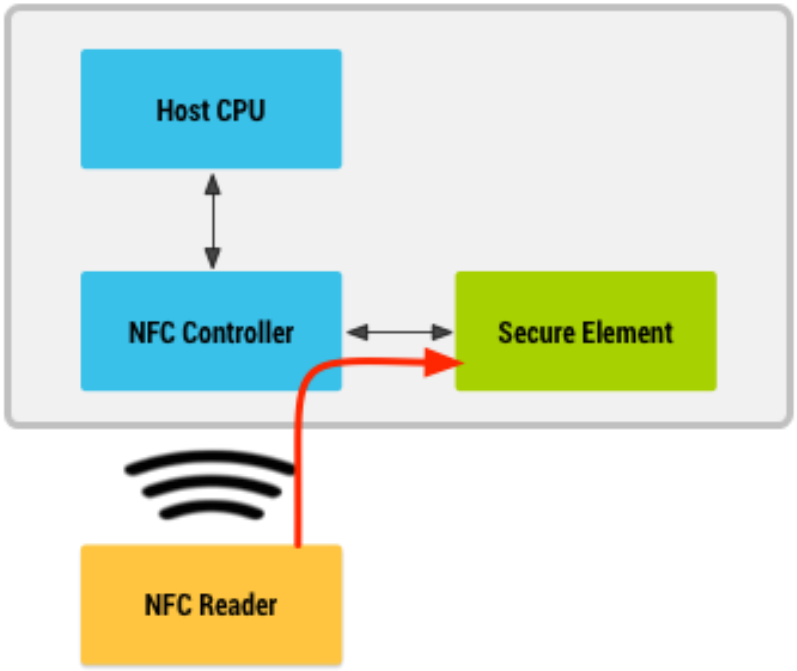


1. Store card details (i.e. PAN) in the SE.

2. Use the NFC enabled device in card emulation mode to make payments.

http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/

Simple protocol (Google Wallet v1.0)



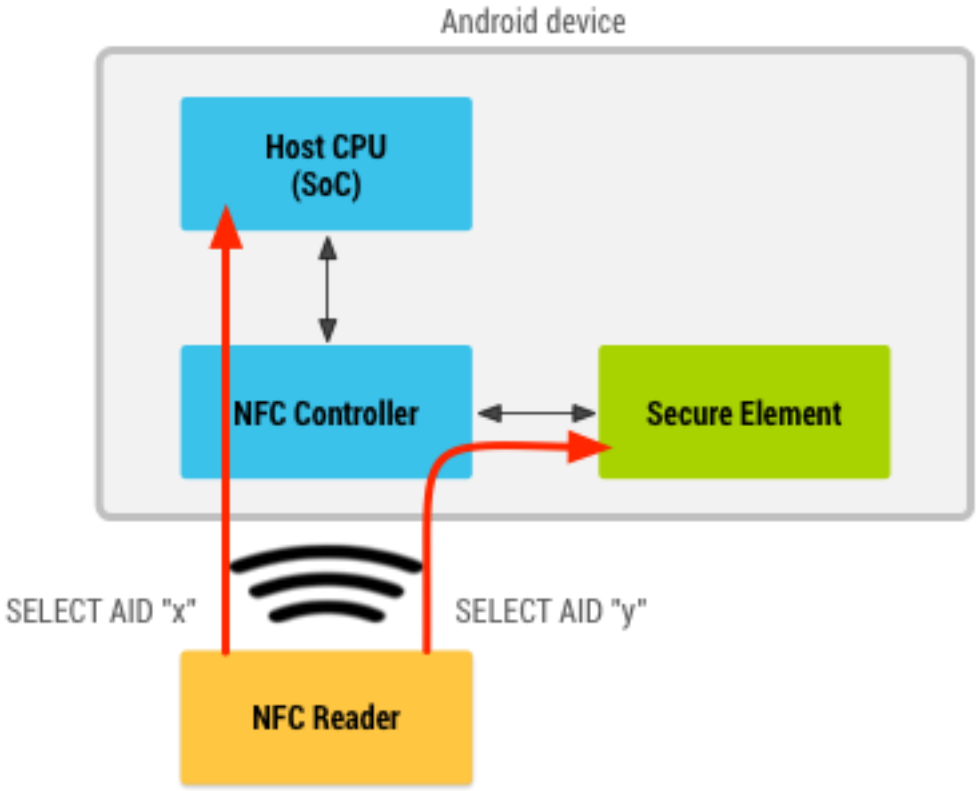
1. Store card details (i.e. PAN) in the SE.

2. Use the NFC enabled device in card emulation mode to make payments.

Didn't work out well for Google because network operators decided to support their own "wallets" (Softcard) and blocked access to the SE.

http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/

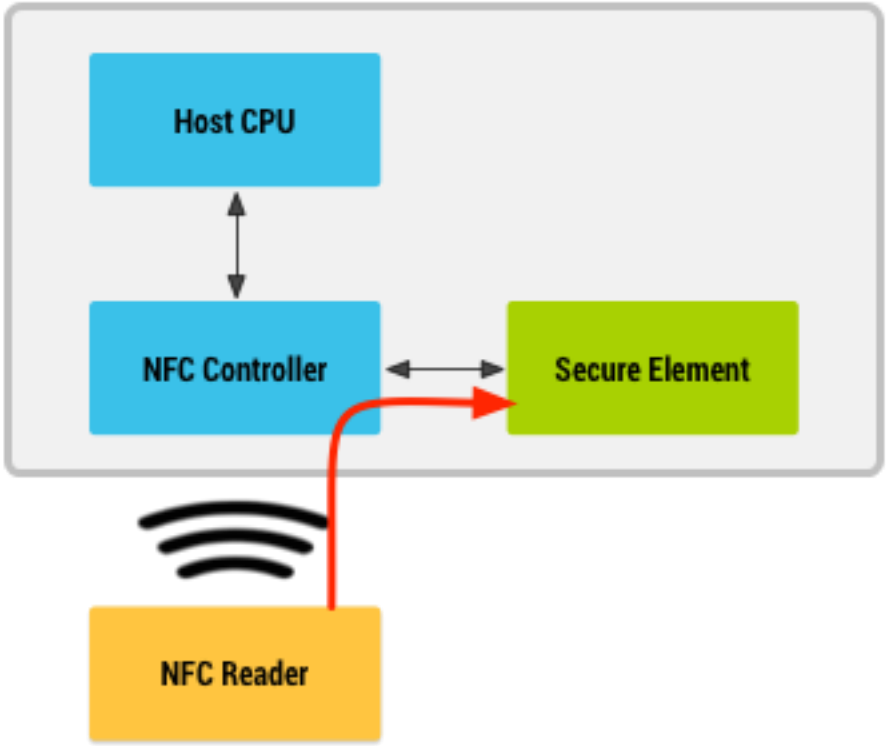
Host-based card emulation protocol (Google Wallet v3.0)



1. Use the NFC enabled device in card emulation mode to make a payment.
2. NFC communicates with the Host OS to request a virtual card number.
3. The Host OS forwards the transaction to the Google cloud.
4. Virtual card number is replaced with PAN and authorized with the Bank.

http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/

Apple Pay protocol



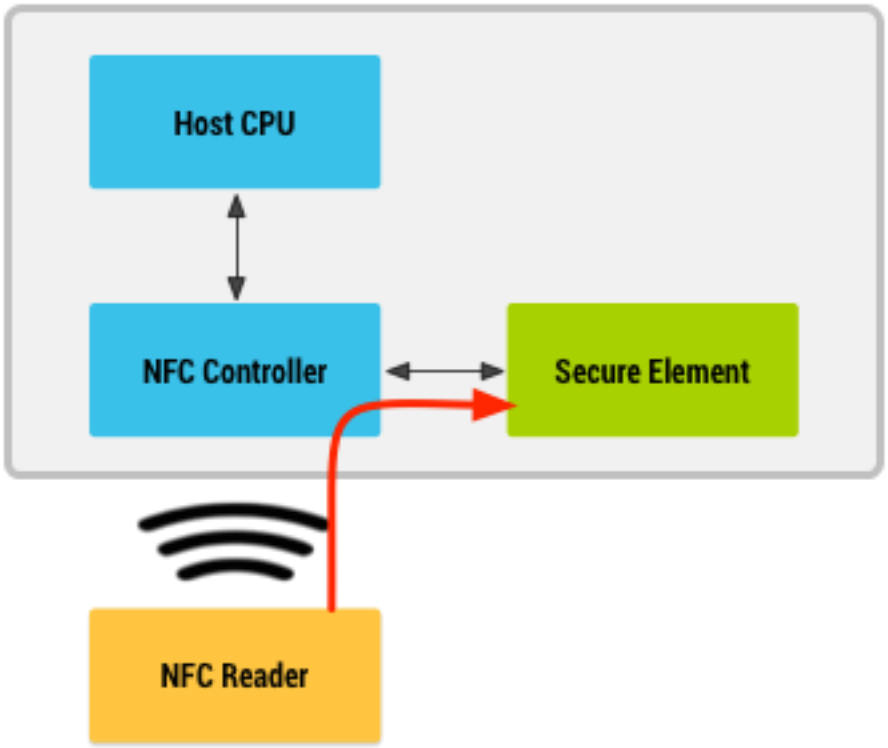
Similar to Google Wave 1.0:

1. Store card details (i.e. PAN) in the SE

2. Use the NFC enabled device in card emulation mode to make payments

http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/

Apple Pay protocol



Similar to Google Wave 1.0:

1. ~~Store card details (i.e. PAN) in the SE.~~
   Store a EMVco Token.

2. Use the NFC enabled device in card emulation mode to make payments

http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/

EMVco Token

- A **token** is fake credit card number that looks and feels like a credit card number.
- The **tokenization** and the **de-tokenization** of a PAN are usually handled by the **Acquiring Bank**.
- In the **EMVCo.** tokenization standard the **de-tokenization** is performed by the **payment network** (e.g. MasterCard, Visa).

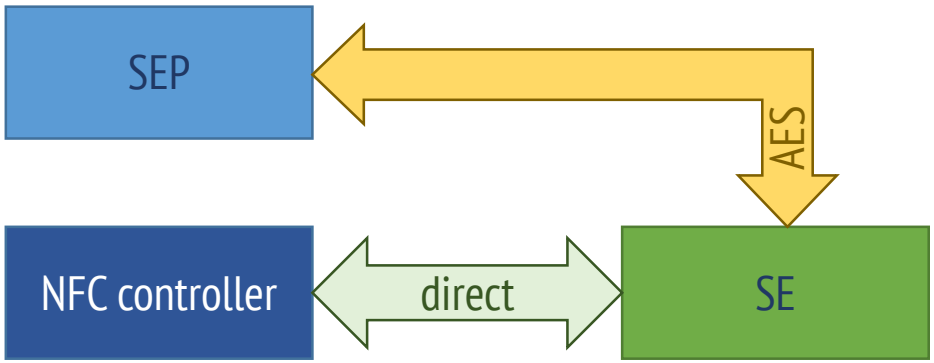
Provisioning a token

Adding a card to Apple Pay:

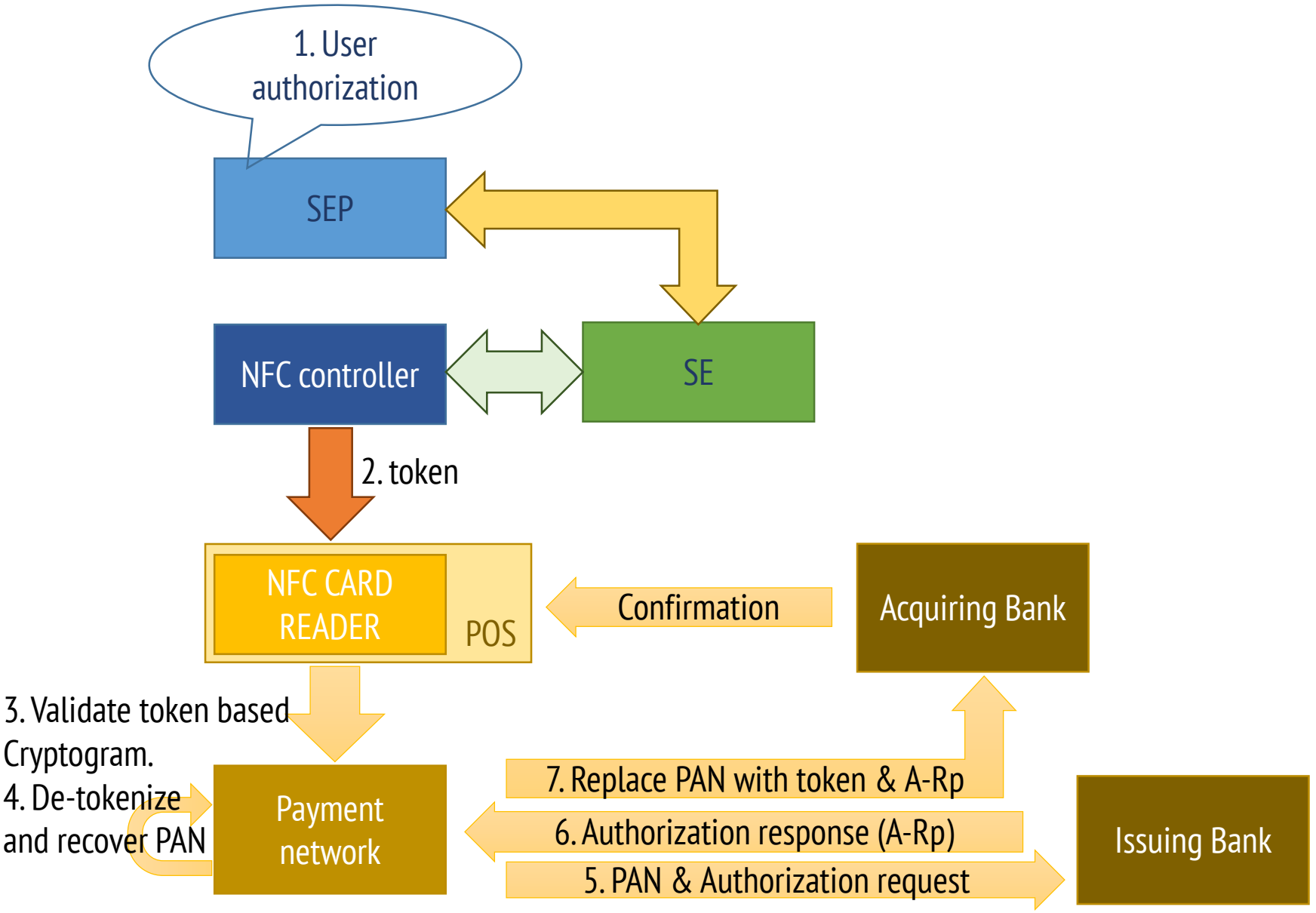
1. Submit PAN details to Apple Pay servers.
2. Apple Pay (i.e. the token requestor) contacts the payment network (i.e. the token service provider) and asks for a token.
3. The payment network contacts the card issuer and performs the card verification.
4. After a successful verification the payment network generates a token and sends it to Apple to provision the SE.

Apple Pay SE and SEP

- The SE communicates with the SEP over a serial interface
- SE and SEP are not directly connected, but have a pre-shared AES key provisioned during the manufacturing process.
- The key is based on the SEP UID and the SE UID
- The SE is tied to an authorization random (AR) generated in SEP.



The payment network



Erasing Cards

- The SE is tied with to an Authorization Random value inside the SEP.
- On receipt of a new Authorization Random the SE marks all previously added cards as deleted
- Cards added in the SE can only be used if the SE is presented with the same Authorization Random that was used during enrolment.
- The SEP can invalidate AR when password is disabled, device restored to default settings, etc.

Apple Pay design principles

- Promote privacy:
  - SE only knows tokens
  - Apple Pay only knows tokens after enrolment
- Assume secrets are not safe
  - SEP can revoke all keys in SE
  - SE can be compromised so only store EMVCo Tokens
- Defend in depth
  - PAN -> EMVCo Token -> SE -> SEP -> user authorization