Security Engineering

Banking and Payment Security I

Bank security

- Bulk compromise: spooks, crooks, whistleblowers
- Targeted attacks: e.g. against celebs
- Insider fraud: use internal controls
- Now: how do you stop external fraud against payment systems?
- Goal: understand payments, cybercrime

Rapid history of banking

- Gold and silver merchants would keep your assets in their vault, for a fee
- Later: you just had a balance with them
- Add interest: deposits, loans, mortgages
- Marco Polo sees paper money in China
- Add trade and foreign payments, creating a paper-based payment system

Paper-based payments

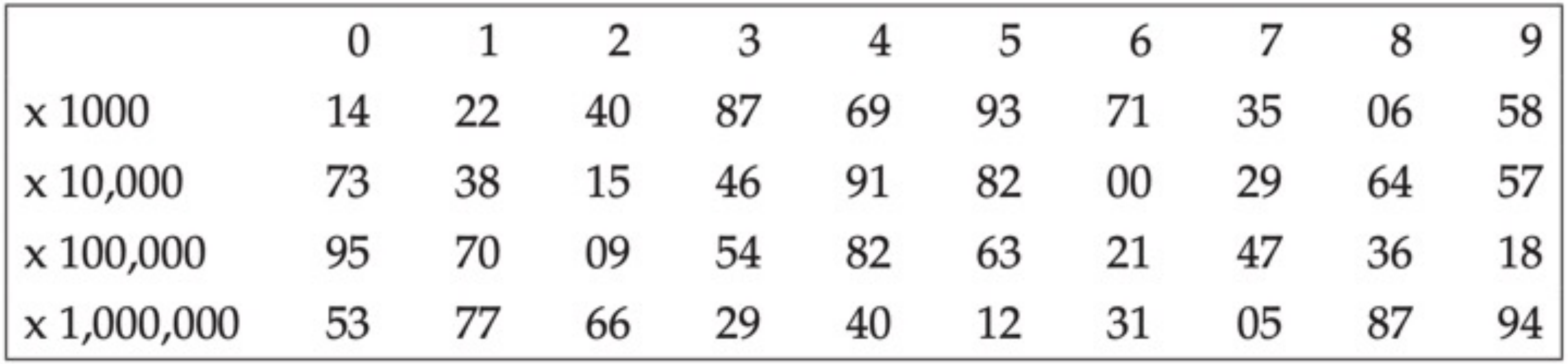
- For centuries we used things like cheques:
  - “Dear RBS, Please pay Alice £100 from my account with you no. 1234, signed Bob”
- Fine between mutually trusting parties
- For users: slow (3–10 days) and uncertain (forged signatures, out of funds...)
- For banks, expensive
- Many different hacks to push back on fraud in different countries...

How do banks ‘send’ money?

- Banks have accounts at correspondent banks in other countries
- If you want to send Aus$1000 to Aunt Agatha in Sydney, your bank sends:
- ‘To National Australia Bank. Please pay Aus$1000 to AD Jones, account XYZ, from our account with you number ABC’
- Now how do you do this electronically?

19th century: ‘Victorian Internet’

- From the 1840s, the telegraph allowed banks to transfer money quickly
- But how do you stop fraud?
- Answer: a ‘test key’ or manual crypto hash. E.g.:



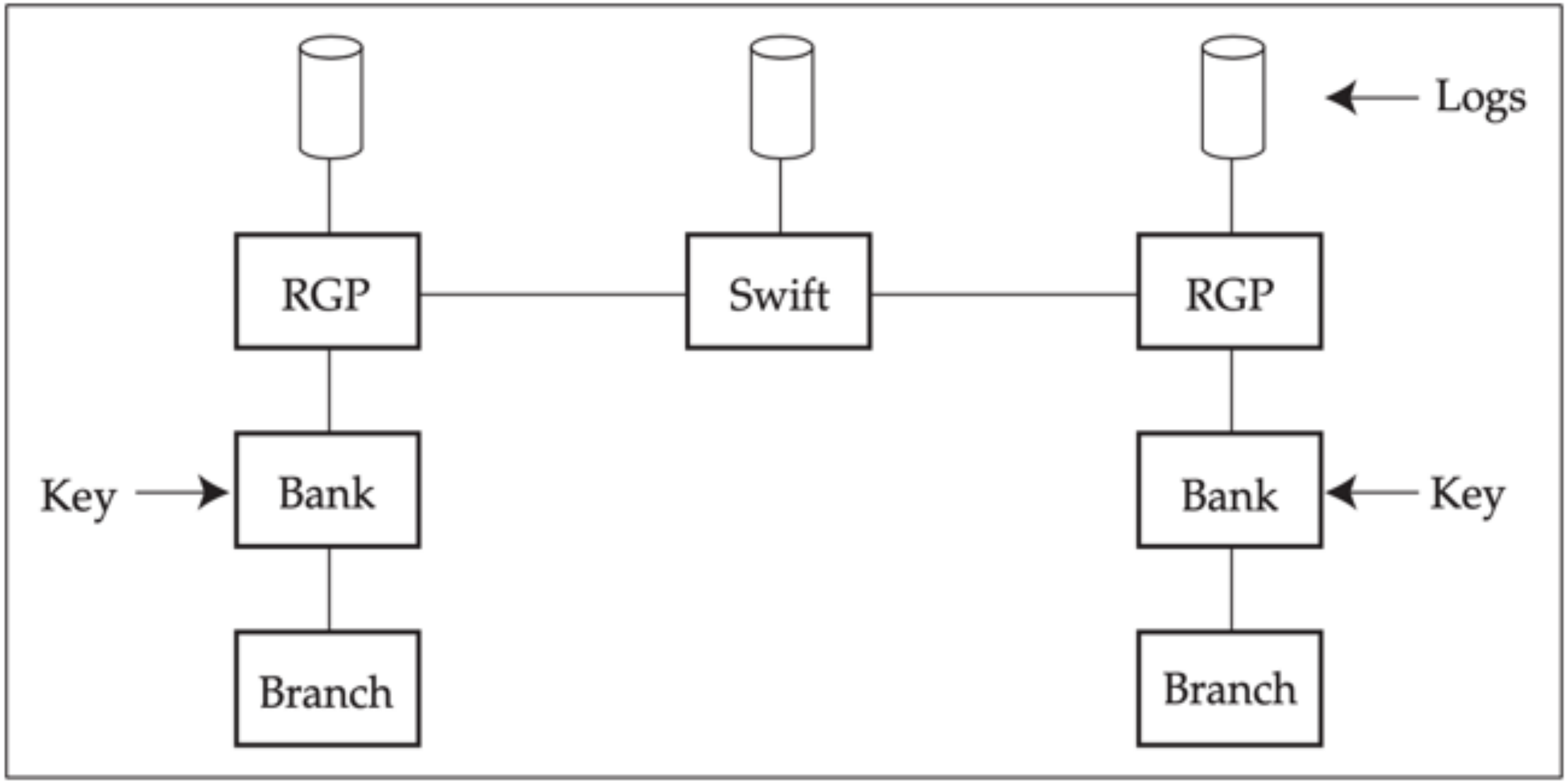
	0	1	2	3	4	5	6	7	8	9
x 1000	14	22	40	87	69	93	71	35	06	58
x 10,000	73	38	15	46	91	82	00	29	64	57
x 100,000	95	70	09	54	82	63	21	47	36	18
x 1,000,000	53	77	66	29	40	12	31	05	87	94

- Auth £376,000 as 71+29+54+53 = 208

What went wrong

- Entry controls: the Security Pacific case
- Failure to support dual control, except by having several people in the room when the test key books removed from the safe
- Weak crypto (only one attempted exploit I know of)
- Test keys persisted for many years after we had better tech...

1970s: SWIFT



- MACs for integrity, logs for non-repudiation

SWIFT controls

- Key management: send keys between senior bank managers in person or by post
- Transaction control: clerk A enters transactions, accountant B checks and can change them, manager C releases them
- Owned by 11,000 banks worldwide
- SWIFT now ‘sends’ $125tr a year but it’s the banks’ own systems that balance funds

Attacks on SWIFT

- Send a guarantee rather than cash!
- If an attacker can figure out how to send money, the hard part is the laundry step
- North Koreans stole $81m from the Bank of Bangladesh, 2016: laundry through a casino
- Attacks on confidentiality by LE/intel agencies, especially post-9/11, and exclusion of Iran

Credit cards

- Slow start in 1950s USA
- Initially: mechanical voucher printing
- 1960s Visa, Mastercard global franchises
- 1980s: a profitable mass market product
- Added online authorization in the UK, then PINs for cash withdrawals from ATMs
- Late 1980s: added CVVs to stop forgery

1970s: Early ATMs

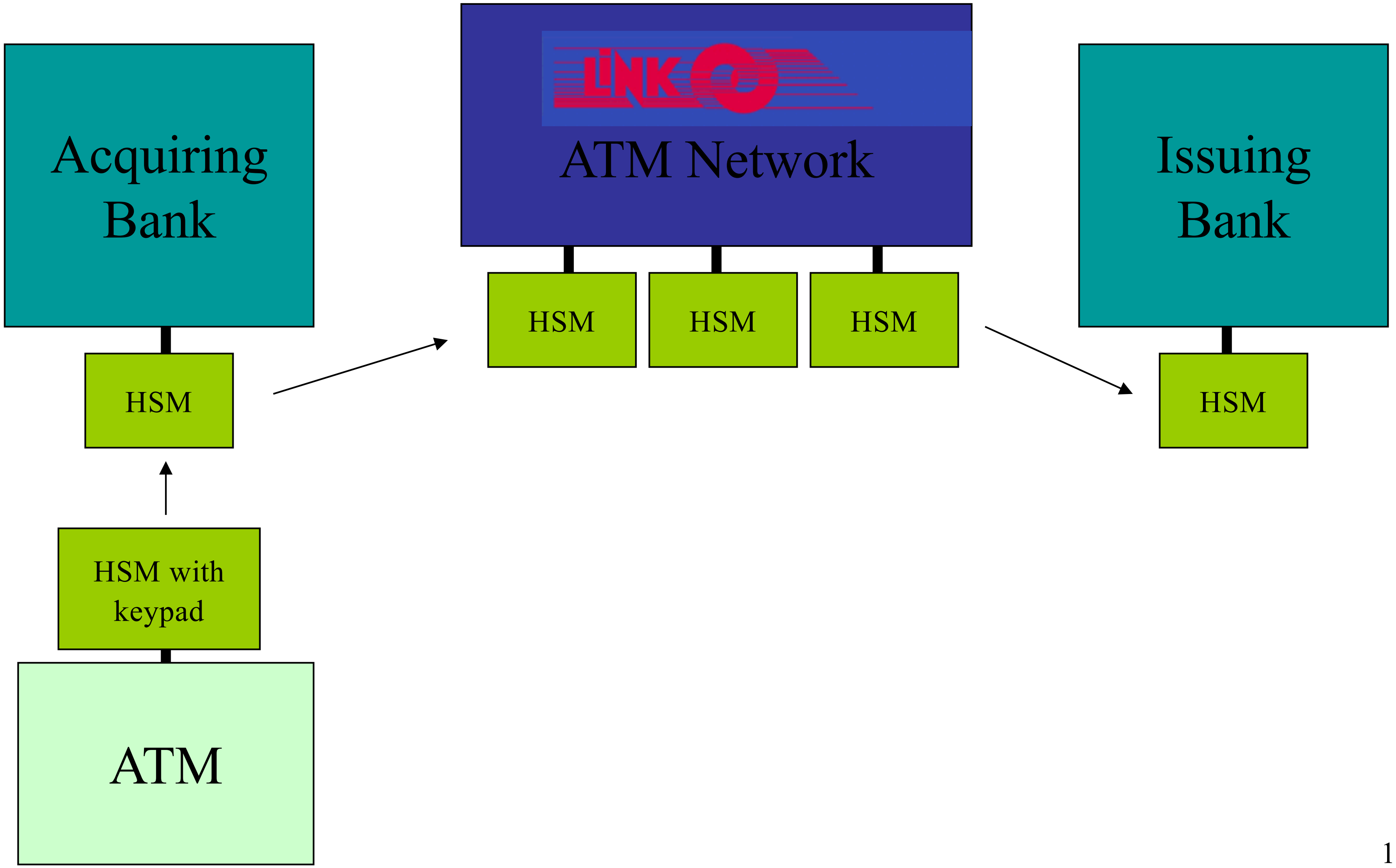
- In my student days: £10 punched cards were returned with your bank statement
- Fraud once people figured out the PIN code
- Lloyds asked IBM for better crypto
- IBM supplied Lucifer, a 128-bit block cipher
- NIST called for a Data Encryption Standard
- NSA limited key length to 56 bits...

1980s: ATM networks

- ATM security, and especially networking, brought cryptography into commerce
- Concrete security policy:

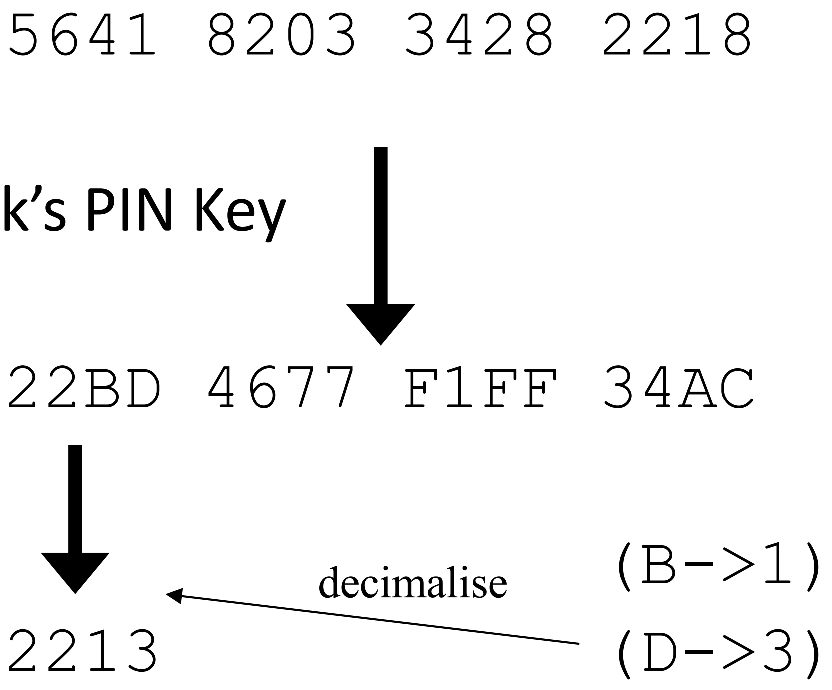
  “Only the customer should know their PIN”
- Standard PIN transactions, using hardware security modules that keep PINs / keys from individual bank staff
- A network switch translates PINs and also keeps settlement accounts

Deep dive – HSM use in banks



How are PINs generated ?

Start with your primary account number (PAN)



5641 8203 3428 2218

Encrypt with your bank's PIN Key

22BD 4677 F1FF 34AC

Chop off the

End

2213

decimalise

(B->1)

(D->3)

(CVVs similar, but use different keys and card version no. too)

How do I change my PIN?

- IBM system: store an offset between the original derived PIN and your chosen PIN
- Example bank record…

– PAN	5641 8233 6453 2229
– Name	Mr M K Bond
– Balance	£1234.56
– PIN Offset	0000

- If Mr Bond changes his PIN from 2213 to 1979, the offset is 9766 (digits mod 10)

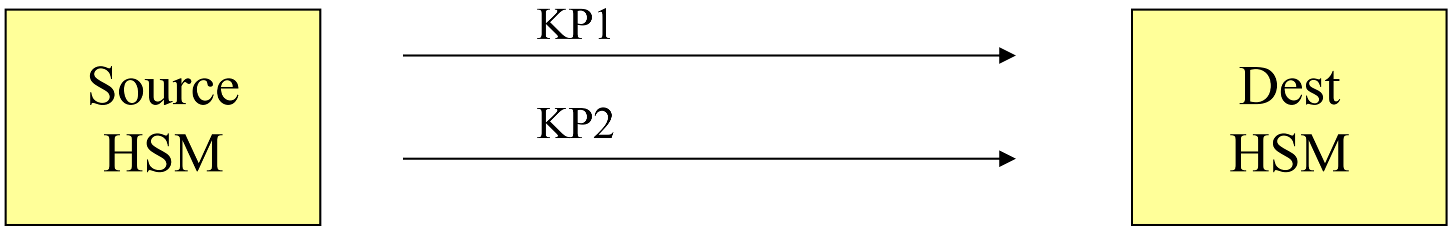
Offset calculation attack (1989)

- Bank needed to issue new account numbers when replacing core banking system
- Got HSM vendors to add a new command to the API to calculate the offset between a new generated PIN and the customer’s chosen PIN
- Oops! Any customer PIN could be revealed by calculating its offset from a known PIN

```
U →C : Old PAN, Old offset, New PAN
C →U : New offset
```

VSM attack (2000)

- Master keys exchanged between banks in two or three parts carried by separate couriers or posted on different days
- These were combined using the exclusive-OR function



Repeat twice…

User→ HSM : Generate Key Component

HSM → Printer: KP1

HSM → User : $\{KP1\}_{ZCMK}$

Combine components…

User→ HSM : $\{KP1\}_{ZCMK}$, $\{KP2\}_{ZCMK}$

HSM → User : $\{KP1 \oplus KP2\}_{ZCMK}$

Repeat twice…

User→ HSM : KP1

HSM → User : $\{KP1\}_{ZCMK}$

Combine components…

User→ HSM : $\{KP1\}_{ZCMK}$, $\{KP2\}_{ZCMK}$

HSM → User : $\{KP1 \oplus KP2\}_{ZCMK}$

Oops! XOR To null key

- XOR was a bad idea! A single operator could feed in the same part twice, getting an 'all zeroes' master key. PINs (and even PIN master keys) could then be extracted in the clear using this key

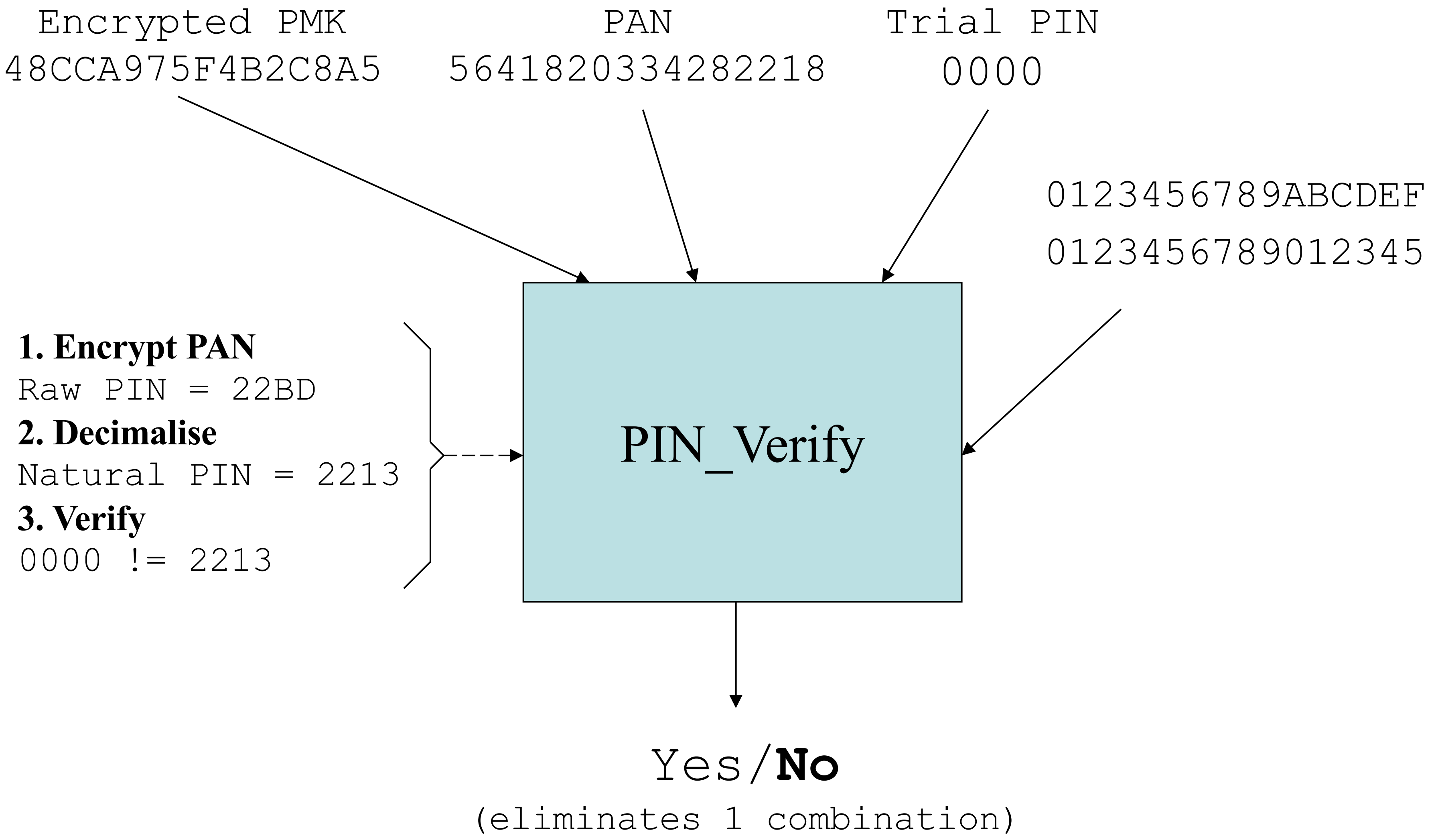
Combine components…

User → HSM : $\{KP1\}_{ZCMK}$, $\{KP1\}_{ZCMK}$

HSM → User : $\{KP1 \oplus KP1\}_{ZCMK}$

KP1 xor KP1 = 0

Decimalisation table attack (1)



Encrypted PMK
48CCA975F4B2C8A5

PAN
5641820334282218

Trial PIN
0000

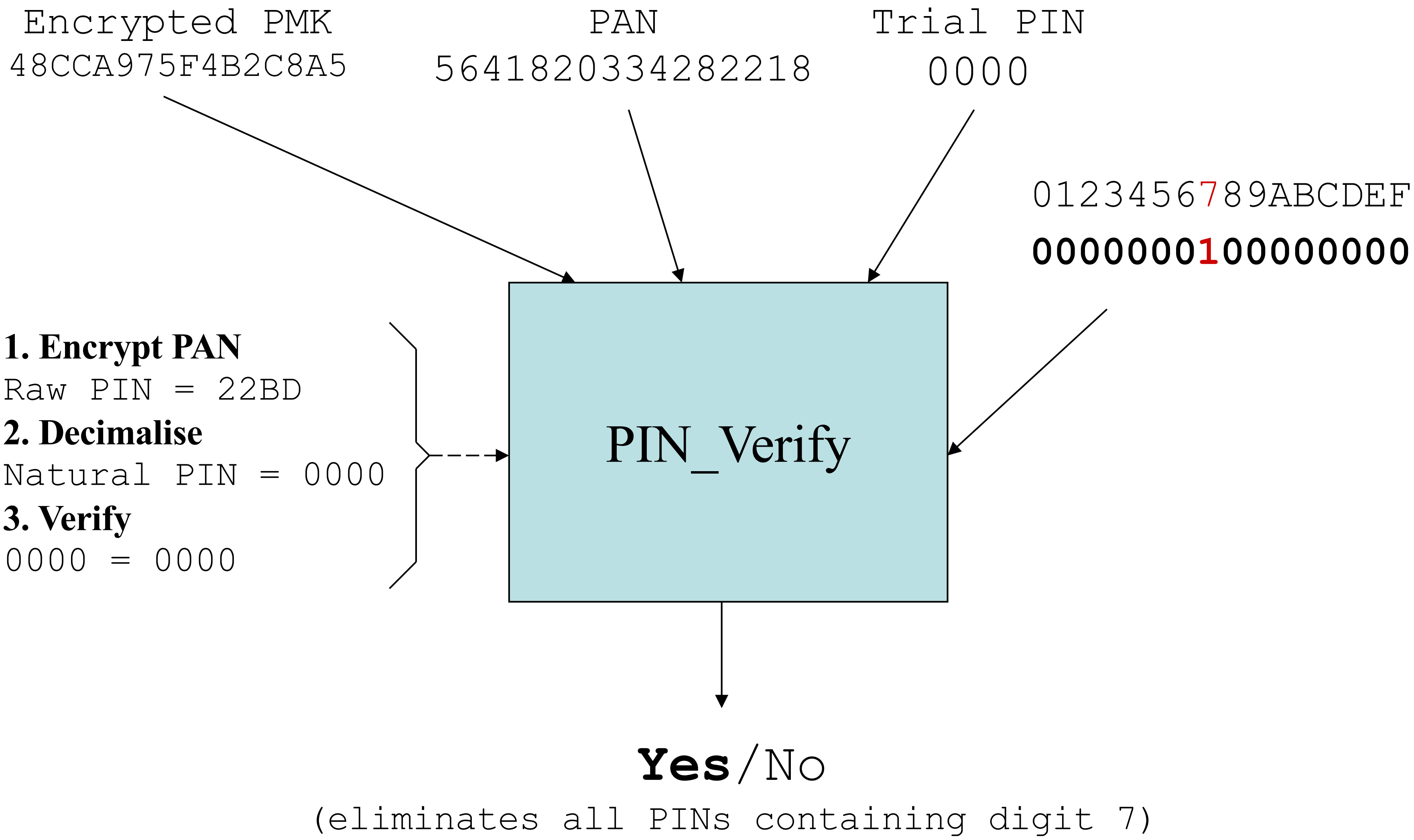
0123456789ABCDEF
0123456789012345

1. Encrypt PAN
Raw PIN = 22BD
2. Decimalise
Natural PIN = 2213
3. Verify
0000 != 2213

PIN_Verify

Yes/**No**
(eliminates 1 combination)

Decimalisation table attack (2)



Encrypted PMK
48CCA975F4B2C8A5

PAN
5641820334282218

Trial PIN
0000

0123456789ABCDEF
0000000100000000

1. Encrypt PAN
Raw PIN = 22BD
2. Decimalise
Natural PIN = 0000
3. Verify
0000 = 0000

PIN_Verify

Yes/No
(eliminates all PINs containing digit 7)

Decimalisation table attack (3)

- Many different attacks can be carried out by playing with the decimalisation table and offset
- Many people who understood HSMs thought up variants once the basic idea was known!
- Generally, it’s a ‘differential attack’ on a private computation. Can you tweak some (untrusted) inputs of a computation so that other (private) inputs are leaked?
- Take care when computing with encrypted data!

What actually went wrong

- Magnetic strip cards were easy to alter or copy
- Gangs learned to do shoulder surfing (fix: CVVs)
- Various implementation bugs, insider attacks, network diversion, malware, social engineering PINs for stolen cards (see my book for more)
- Fraud was industrialised using ATM skimmers
- Fix:
  - intrusion detection systems, e.g. FICO
  - move to smartcards with EMV from 2004

Europay/MasterCard/VISA (EMV)

Chip and PIN

EMVCo

VISA

EXP: 30/11/09

ICC

SALE

£50.00

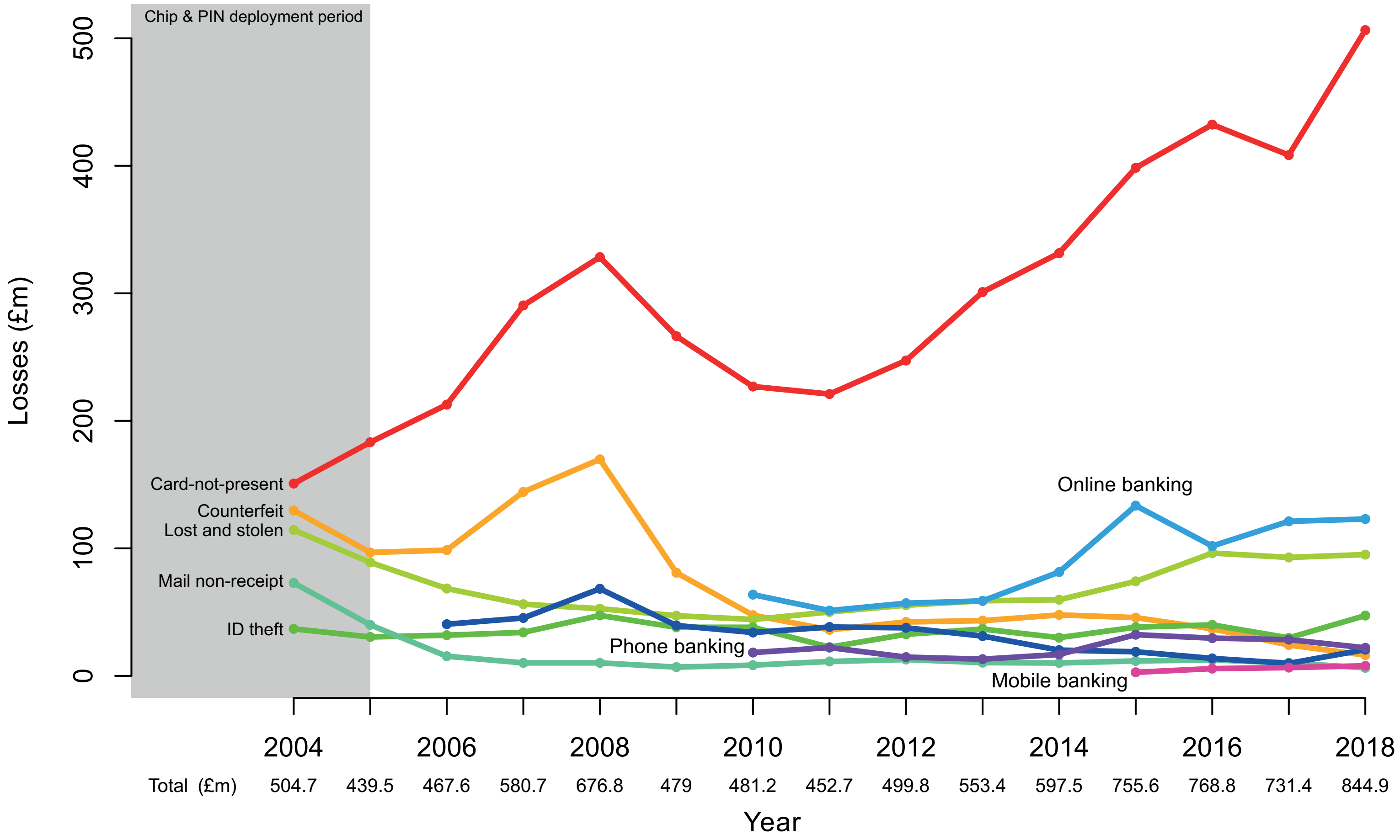
VERIFIED BY PIN

- EMV (‘Chip and PIN’) deployed from 2004, for debit & credit cards
- ‘Liability shift’ – disputes charged to cardholder if pin used, else to merchant
- Changed many things, not always in the ways banks expected!

Security economics of PINs

- With cheques, a forged signature was null and void
- So the forgery risk fell on the relying party
- With PINs, banks pushed hard for the risk to move to the user
- ‘Your card and PIN were used so you were negligent or complicit’
- What could possibly go wrong?

Fraud in the UK since EMV



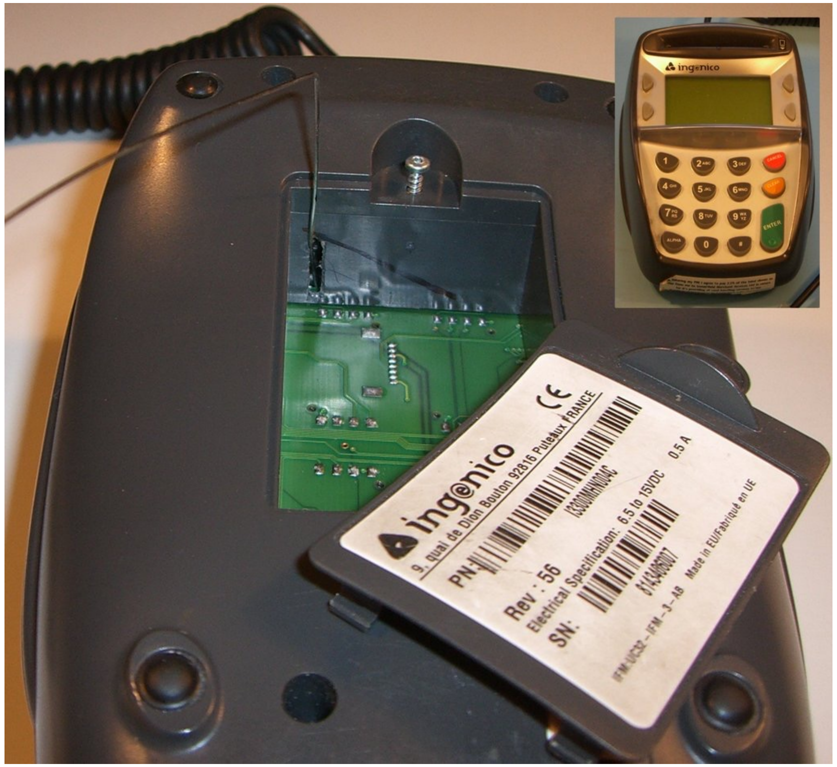
EMV shifted the landscape…

- It caused the fraud to find new channels
- Card-not-present fraud shot up at once
- Counterfeit took off once the crooks realised:
  - It’s easier to steal card and pin details once pins are used everywhere
  - You could still use mag-strip fallback for several years (especially overseas)
  - Tamper-resistance didn’t work properly

Security economics of PEDs

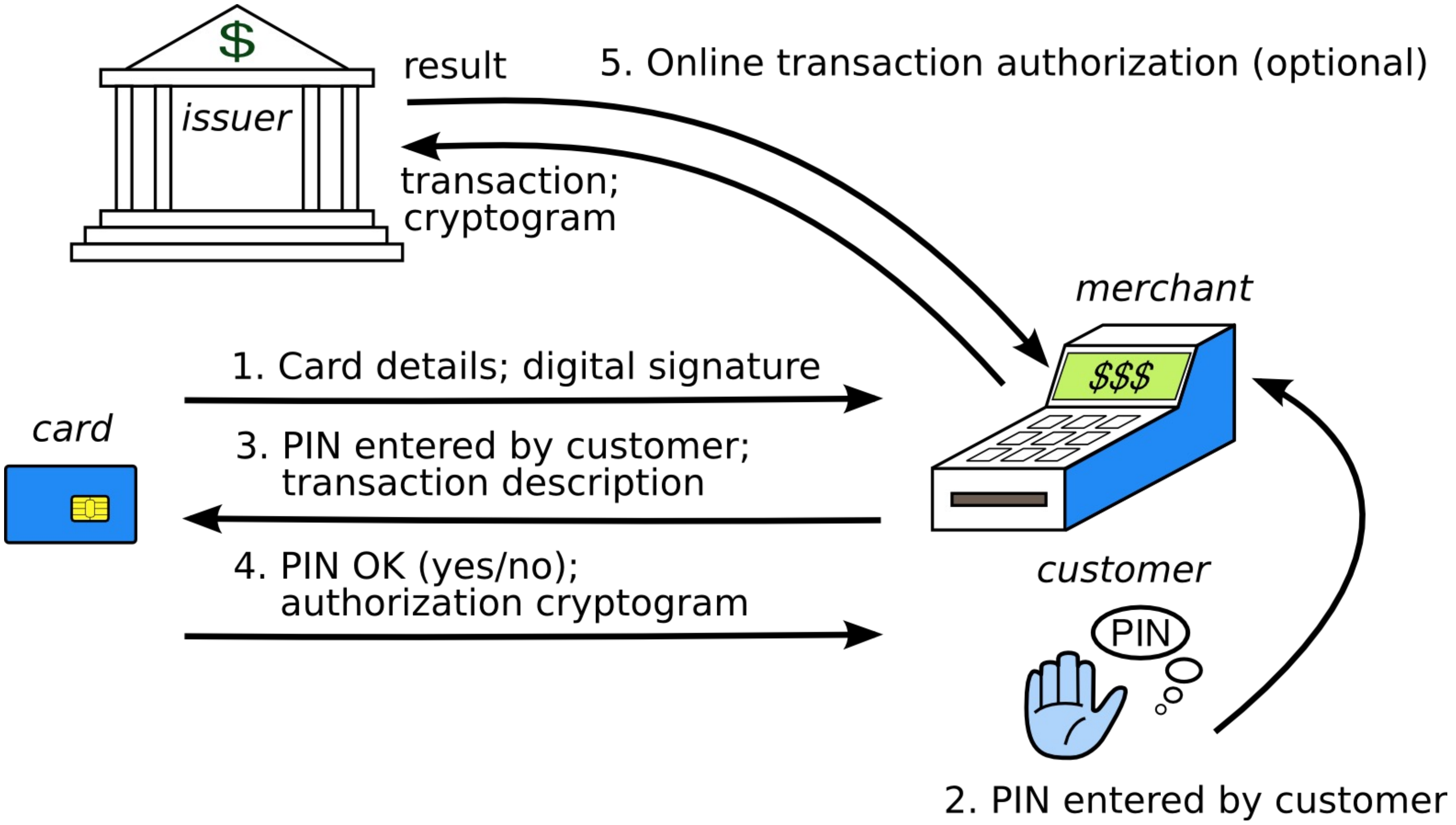
- If the merchant services division of a bank can pay an extra £10 per terminal to make 1m PIN entry devices secure, and thereby save £100m in fraud, will they?
- Not if the bank has only 8% of the cards in issue!
- And especially not if the card division’s manager is a rival of the merchant services division’s manager

How Certification Fails

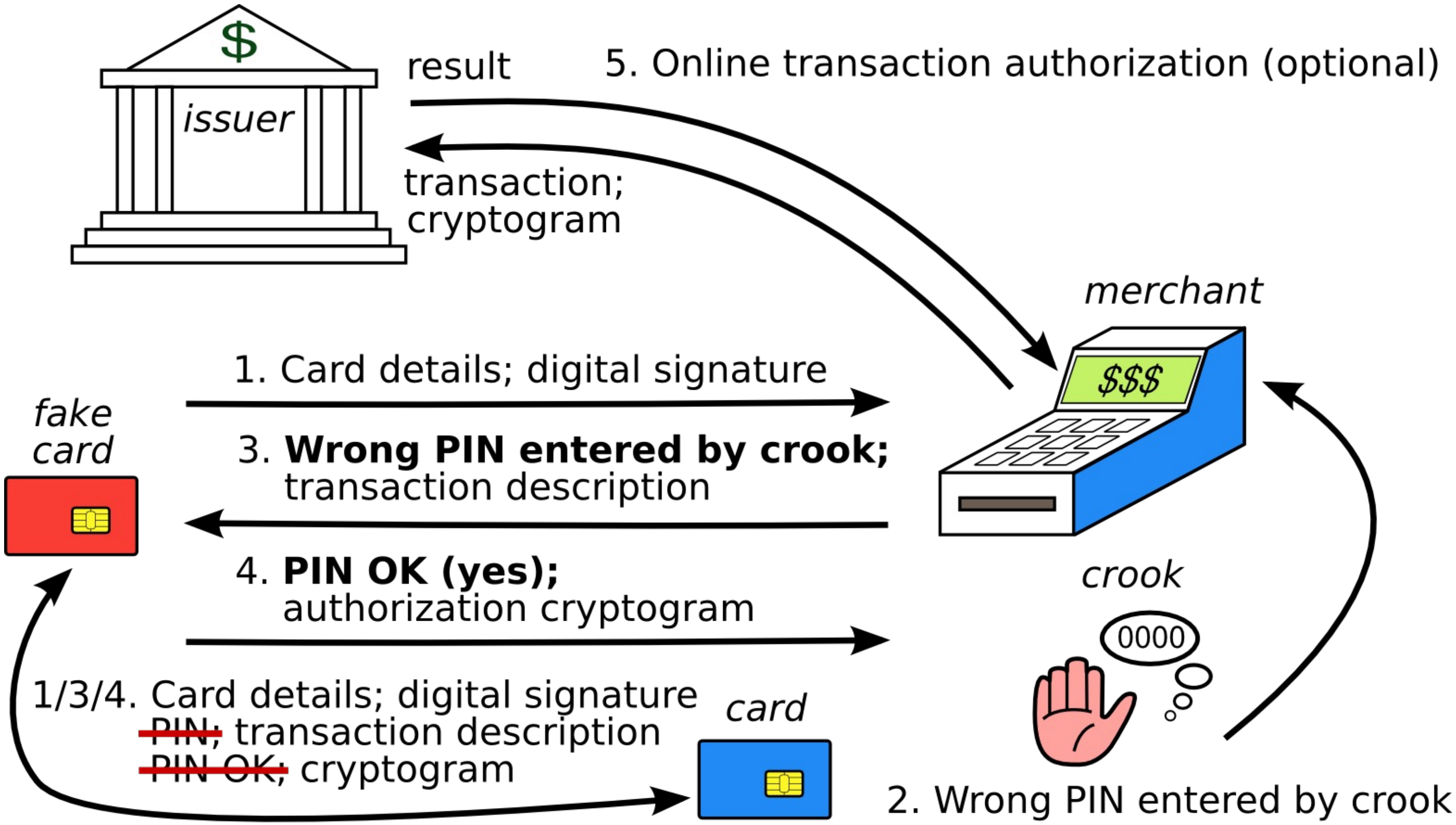

ingenico
9, quai de Dion Bouton 92816 Puteaux FRANCE
PN:
Rev : 56
Electrical Specification: 6.5 to 15VDC 0.5 A
SN:
Made in EU/Fabriqué en UE
ingenico
ENTER

- PIN entry devices ‘evaluated under the Common Criteria’ were trivial to tap
- Banks claimed it wasn’t a problem
- CC didn’t want to know
- Suddenly Shell had to replace all its PEDs …

A normal EMV transaction


issuer
$
result
5. Online transaction authorization (optional)
transaction;
cryptogram
merchant
$$$
1. Card details; digital signature
card
3. PIN entered by customer;
transaction description
4. PIN OK (yes/no);
authorization cryptogram
customer
PIN
2. PIN entered by customer

The ‘No-PIN’ attack (2010)



issuer
$
result
5. Online transaction authorization (optional)
transaction;
cryptogram
merchant
$$$
1. Card details; digital signature
fake
card
3. **Wrong PIN entered by crook;**
transaction description
4. **PIN OK (yes);**
authorization cryptogram
crook
0000
1/3/4. Card details; digital signature
~~PIN;~~ transaction description
~~PIN OK;~~ cryptogram
card
2. Wrong PIN entered by crook

Fixing the ‘No PIN’ attack

- In theory: might block at terminal, acquirer, issuer
- In practice: may have to be the issuer (as with terminal tampering, acquirer incentives are poor)
- Barclays introduced a fix July 2010; removed Dec 2010 (too many false positives?); banks asked for student thesis to be taken down from web instead
- Real problem: EMV spec now far too complex
- With 100+ vendors, 20,000 banks, millions of merchants … everyone passes the buck

API attacks keep coming back!

- A new HSM transaction was defined by VISA for EMV support
- Send key from HSM1 to HSM2 as {text | key} – where text is variable-length
- Attack – encrypt {text |00}, {text |01}, etc to get first byte of key, and so on
- This vulnerability turned up in all HSMs!
- As fast as the HSM industry removed API vulnerabilities, the banks put them back

ATMs and Random Numbers

- Let’s look at the authentication request cryptogram (ARQC) the card computes
- The terminal sends a random number N to the card along with the date d and the amount X; the ARQC authenticates (N, d, X)
- So what happens if I can predict N for d?
- Answer: if I have access to your card I can precompute an ARQC for amount X, date d

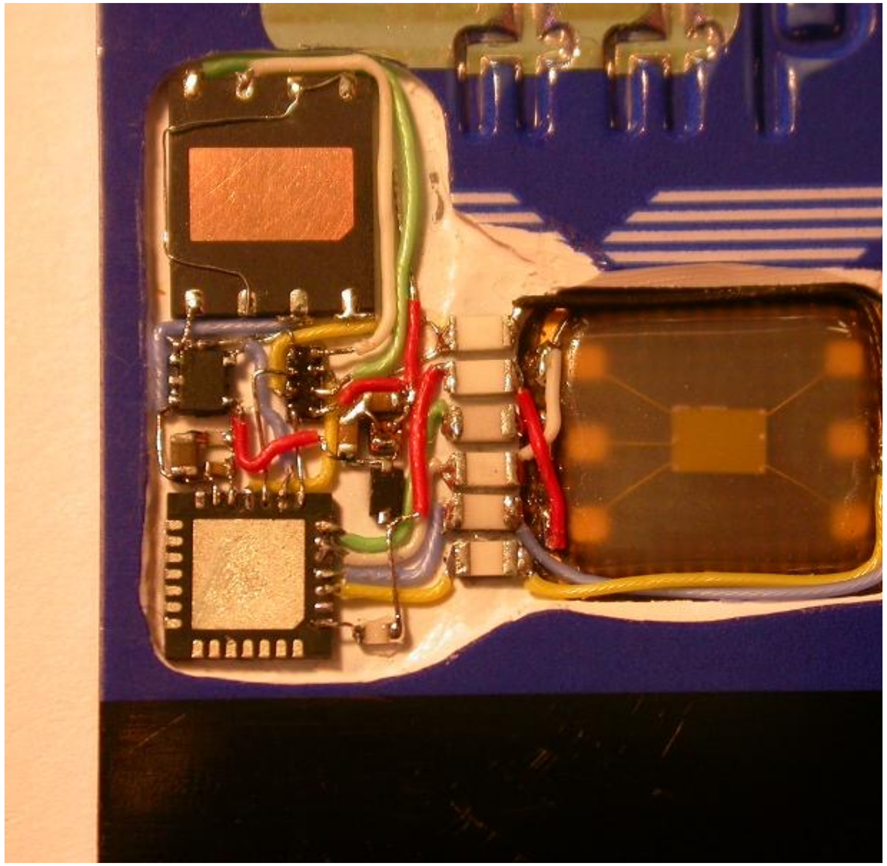
ATMs and Random Numbers (2)

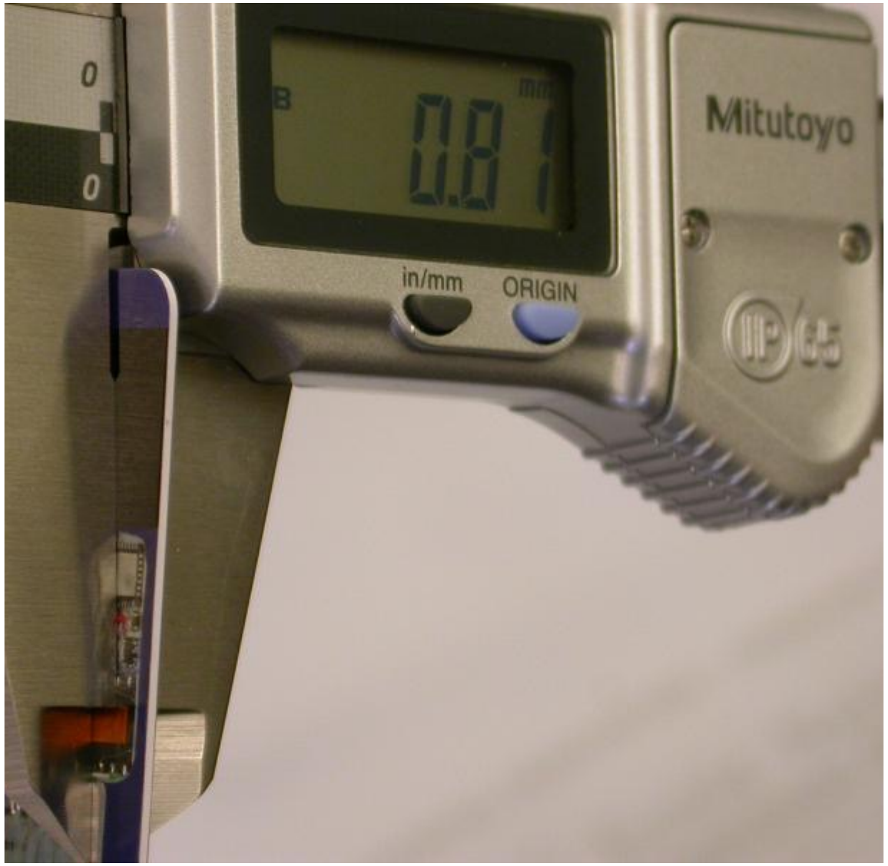
- Log of disputed transactions in Majorca:

```
2011-06-28          10:37:24          F1246E04
2011-06-28          10:37:59          F1241354
2011-06-28          10:38:34          F1244328
2011-06-28          10:39:08          F1247348
```

- N is a 17 bit constant followed by a 15 bit counter cycling every 3 minutes
- We test, & find half of ATMs use counters!

ATMs and Random Numbers (3)





The preplay attack (2014)

- British sailor goes into bar in Barcelona and pays €33 for two drinks
- Wakes up with a sore head and finds that €33,000 taken in ten payments of €3,300
- Lloyds says ‘Your card and PIN were used…’
- 10 transactions an hour apart, from the same terminal, but through three banks
- EMV lacks a trustworthy user interface!

The preplay attack (2)

- Bournemouth council had a dispute with a lap-dancing club
- Multiple customers had complained of excessive card charges; banks, police did nothing
- ‘The Bournemouth Echo’ wrote about the preplay attack, and more victims contacted the councillor
- The club’s license was restricted for six months
- Police still not interested...