Security – protocols, crypto etc

Computer Science Tripos part 2

Steven J. Murdoch
Slides originally by Ross Anderson

Security Protocols

- Security protocols are the intellectual core of security engineering
- They are where cryptography and system mechanisms meet
- They allow trust to be taken from where it exists to where it’s needed
- But they are much older then computers…

Real-world protocol

- Ordering wine in a restaurant
  - Sommelier presents wine list to host
  - Host chooses wine; sommelier fetches it
  - Host samples wine; then it's served to guests
- Security properties
  - Confidentiality – of price from guests
  - Integrity – can't substitute a cheaper wine
  - Non-repudiation – host can't falsely complain

Car unlocking protocols

- Principals are the engine controller E and the car key transponder T
- Static ($T \rightarrow E$: KT)
- Non-interactive

  $T \rightarrow E$: T, $\{T,N\}_{KT}$
- Interactive

  $E \rightarrow T$: N

  $T \rightarrow E$: $\{T,N\}_{KT}$
- N is a ‘nonce’ for ‘number used once’. It can be a serial number, random number or a timestamp

What goes wrong

- In cheap devices, N may be random or a counter – one-way comms and no clock
- It can be too short, and wrap around
- If it's random, how many do you remember? (the valet attack)
- Counters and timestamps can lose sync leading to DoS attacks
- There are also weak ciphers – Eli Biham's 2008 attack on the Keeloq cipher (2^{16} chosen challenges then 500 CPU days' analysis – some other vendors authenticate challenges)

Two-factor authentication

$S \rightarrow U$: N

$U \rightarrow P$: N, PIN

$P \rightarrow U$: $\{N, PIN\}_{KP}$

Identify Friend or Foe (IFF)

- Basic idea: fighter challenges bomber

  $F \rightarrow B: N$

  $B \rightarrow F: \{N\}_K$

- But what if the bomber reflects the challenge back at the fighter's wingman?

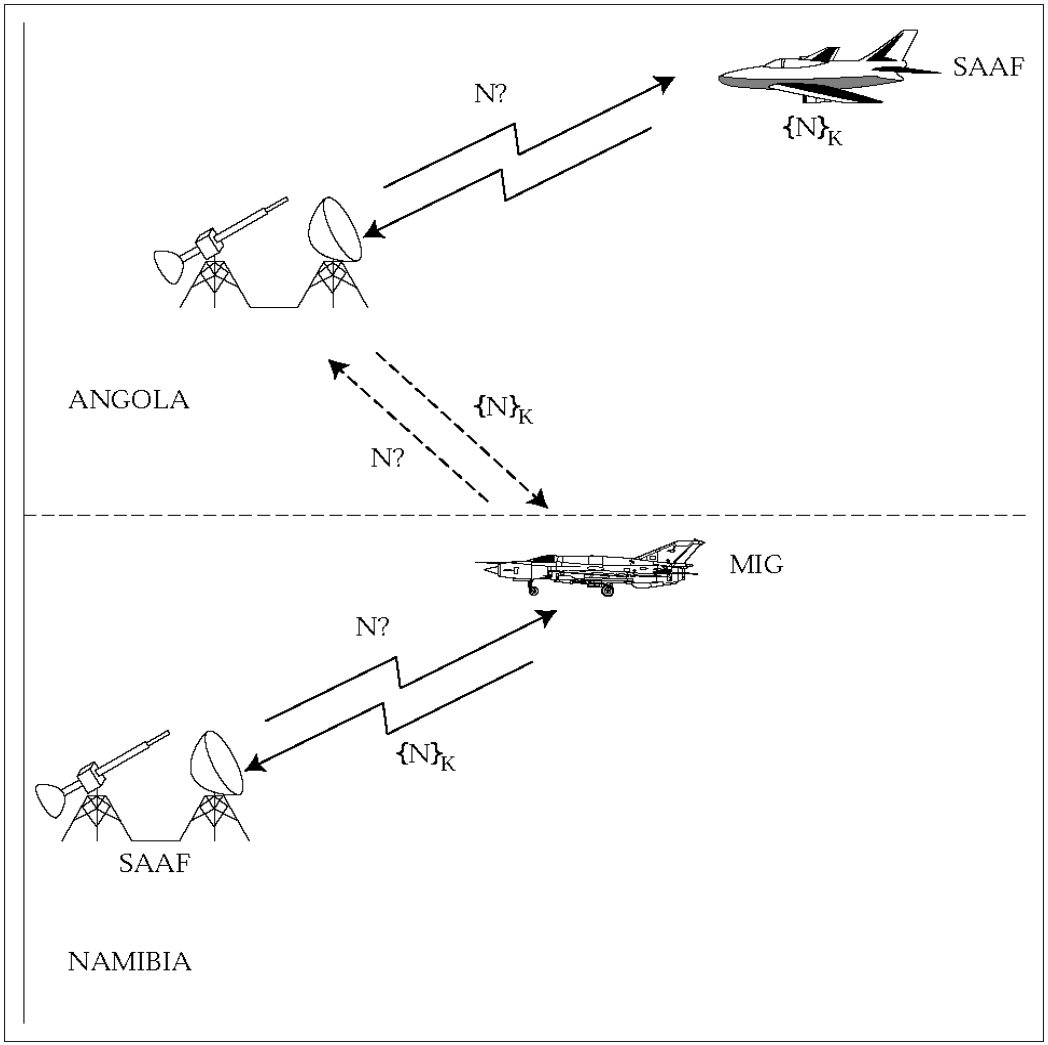
  $F \rightarrow B: N$

  $B \rightarrow F: N$

  $F \rightarrow B: \{N\}_K$

  $B \rightarrow F: \{N\}_K$

IFF (2)



IFF (3)

- The middleman attack is very general – Conway discussed how to beat a grandmaster at postal chess
- The fix for the man-in-the-middle attack is often application specific
- E.g. NATO mode 12 IFF: 32 bit encrypted challenge (to prevent enemy using IFF to locate beyond radar range) at rate of 250 per second

Offline PIN Problem, 1993

- IBM system for ATMs:

  $PIN = \{PAN\}_{KP}$

- Offline operation: write $\{PIN\}_{KA}$ to the card track and give KA to all ATMs
- What's wrong with this? (the crooks found out in 1993 and offline operations had to be suspended)

Chip Authentication Program (CAP)

- Introduced by UK banks to stop phishing
- Each customer has an EMV chipcard
- Easy mode:

  $U \rightarrow C$: PIN

  $C \rightarrow U$: $\{N, PIN\}_{KC}$

- Serious mode:

  $U \rightarrow C$: PIN, amt, last 8 digits of payee A/C…


Maestro
INSERT CARD
OR MENU
NatWest
ON/OFF
Identify
Respond
Sign
MENU
1
2
3
C
4
5
6
CL
7
8
9
OK
0

What goes wrong…

guardian.co.uk

Police think French pair tortured for pin details

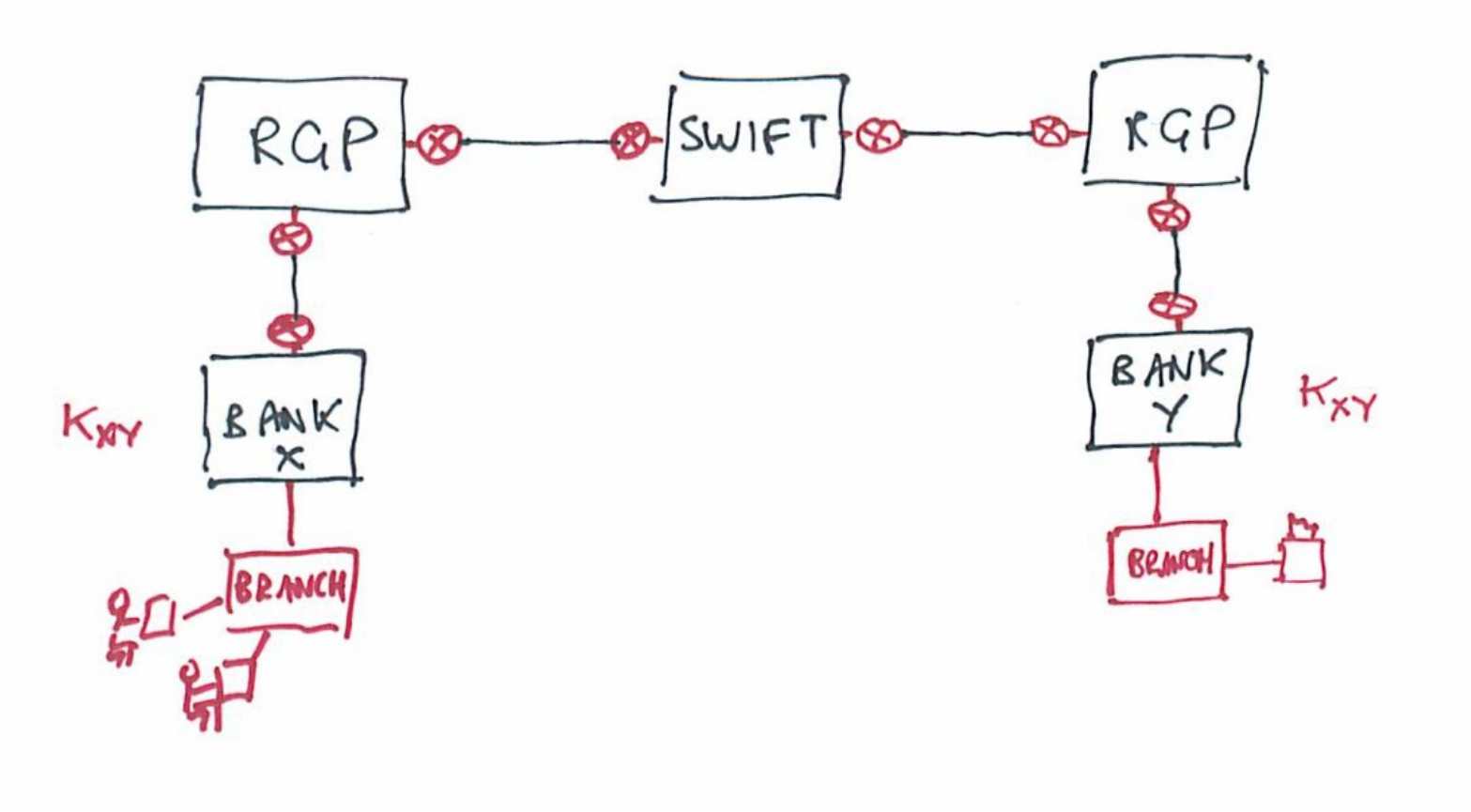
Matthew Taylor
The Guardian, Saturday July 5 2008

Laurent Bonomo and Gabriel Ferez, two French exchange students who were killed in London. Photographs: Met police/Getty

The two French students who were bound up and brutally murdered at a bedsit in south London may have been tortured for their bank and credit card pin numbers, police said yesterday.

Laurent Bonomo and Gabriel Ferez, both 23, were found at Bonomo's flat in New Cross, south London, on Sunday night. They had been stabbed more than 200 times, bound, gagged, and tortured over several hours.

SWIFT



Key Management Protocols

- HomePlug AV has maybe the simplest…
- *Secure mode*: type the device key KD from the device label into the network hub. Then

  $H \rightarrow D: \{KM\}_{KD}$

- *Simple-connect mode*: hub sends a device key in the clear to the device, and user confirms whether it's working
- Optimised for usability, low support cost

Key management protocols (2)

- Suppose Alice and Bob share a key with Sam, and want to communicate?
  - Alice calls Sam and asks for a key for Bob
  - Sam sends Alice a key encrypted in a blob only she can read, and the same key also encrypted in another blob only Bob can read
  - Alice calls Bob and sends him the second blob
- How can they check the protocol's fresh?

Key management protocols (2)

- Here's a possible protocol

  $A \rightarrow S$: A, B

  $S \rightarrow A$: $\{A, B, K_{AB}, T\}_{KAS}$, $\{A, B, K_{AB}, T\}_{KBS}$

  $A \rightarrow B$: $\{A, B, K_{AB}, T\}_{KBS}$

- She finally sends him whatever message she wanted to send, encrypted under K_{AB}

  $A \rightarrow B$: $\{M\}_{KAB}$

A Quick Test

- The following protocol was proposed by Woo and Lam for logon authentication

  $A \rightarrow B$: A

  $B \rightarrow A$: NB

  $A \rightarrow B$: $\{NB\}_{KAS}$

  $B \rightarrow S$: $\{A, \{NB\}_{KAS}\}_{KBS}$

  $S \rightarrow B$: $\{NB\}_{KBS}$

- Is it OK?

Needham-Schroder

- 1978: uses nonces rather than timestamps

  $A \rightarrow S$: A, B, NA

  $S \rightarrow A$: $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{KBS}\}_{KAS}$

  $A \rightarrow B$: $\{K_{AB}, A\}_{KBS}$

  $B \rightarrow A$: $\{NB\}_{KAB}$

  $A \rightarrow B$: $\{NB - 1\}_{KAB}$

- The bug, and the controversy…

Otway-Rees

- Proposed fix for NS – also allows nested RPCs

  $A \rightarrow B$: M A, B, $\{N_A,M,A,B\}_{KAS}$

  $B \rightarrow S$: M A, B, $\{N_A,M,A,B\}_{KAS,}\{N_B,M,A,B\}_{KBS}$

  $S \rightarrow B$: M, $\{N_A, K_{AB}\}_{KAS},\{N_B, K_{AB}\}_{KBS}$

  $B \rightarrow A$: $\{N_A, K_{AB}\}_{KAS}$

- Passes formal verification…
- But can still break with poor implementation (e.g. if you use CBC encryption with block boundaries aligned with the protocol element boundaries)

Kerberos

- The ‘revised version’ of Needham-Schroder – nonces replaced by timestamps

  $A \rightarrow S$: A, B

  $S \rightarrow A$: $\{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}_{KBS}\}_{KAS}$

  $A \rightarrow B$: $\{T_S, L, K_{AB}, A\}_{KBS,} \{A, T_A\}_{KAB}$

  $B \rightarrow A$: $\{A, T_A\}_{KAB}$

- Now we have to worry about clock sync!
- Kerberos variants very widely used…

GSM

- Each handset SIM has an individual key Ki
- Home network sends visited network (RAND, SRES, Kc) where (SRES | Kc) = $\{RAND\}_{Ki}$
- Handset $\rightarrow$ Network: IMSI
- Network $\rightarrow$ Handset: RAND
- Handset $\rightarrow$ Network: SRES, $\{traffic\}_{Kc}$
- Attacks?

3g

- 3g (UMTS) protocol fixes the weak ciphers and vulnerability to rogue base stations
- $\{RAND\}_K = (RES|CK|IK|AK)$, giving keys for confidentiality, integrity and anonymity

  USIM $\rightarrow$ HE: IMSI

  HE $\rightarrow$ VLR: RAND,RES,CK,IK, SEQ$\oplus$AK, MAC

  VLR $\rightarrow$ USIM: RAND, SEQ$\oplus$AK, MAC

  USIM$\rightarrow$ VLR: RES

Formal methods

- Many protocol errors result from using the wrong key or not checking freshness
- Formal methods used to check all this!
- The core of the Burrows-Abadi-Needham logic:
  - M is true if A is an authority on M and A believes M
  - A believes M if A once said M and M is fresh
  - B believes A once said X if he sees X encrypted under a key B shares with A
- See book chapter 3 for a worked example

Another Quick Test

- In the ‘wide-mouthed frog’ protocol – Alice and Bob each share a key with Sam, and use him as a key-translation service

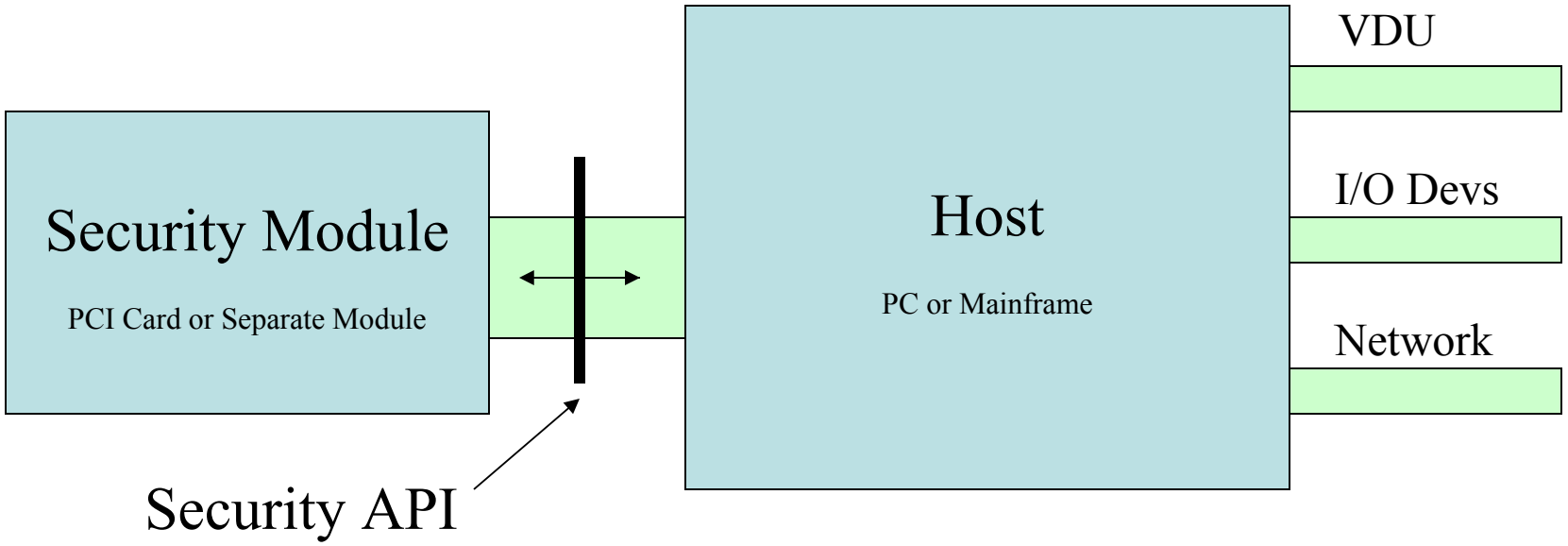
$$A \rightarrow S: \{T_A, B, K_{AB}\}K_{AS}$$

$$S \rightarrow B: \{T_S, A, K_{AB}\}K_{BS}$$

- Is this protocol sound, or not?

What is a Security API ?

- An API that allows users to work with sensitive data and keys, and uses cryptography to enforce a policy on the usage of data



Hardware Security Modules

- An instantiation of a security API
- Often physically tamper-resistant
  (epoxy potting, temperature & x-ray sensors)
- May have hardware crypto acceleration
  (not so important with speed of modern PC)
- May have special ‘trusted’ peripherals
  (key switches, smartcard readers, key pads)

(referred to as HSMs subsequently)

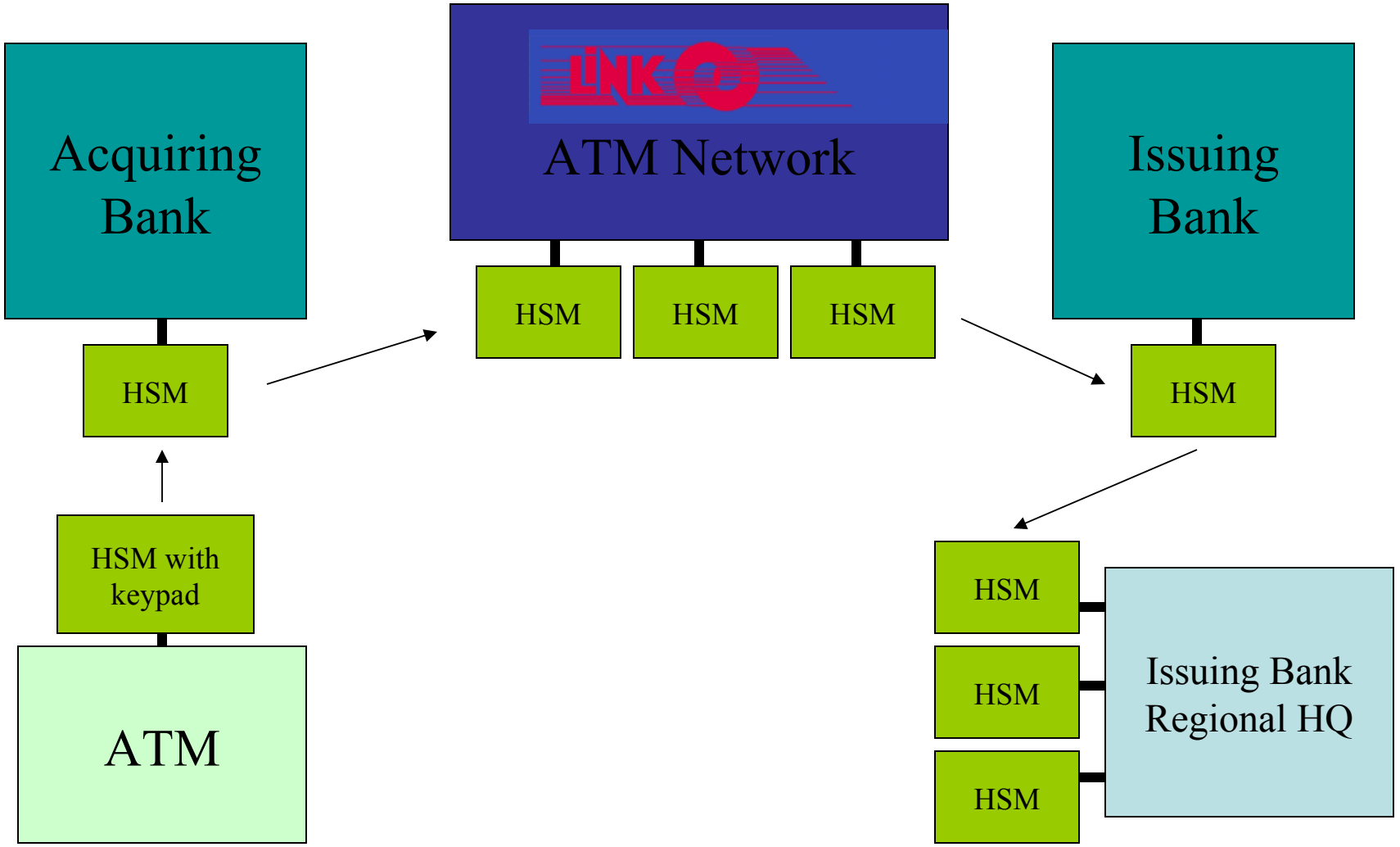
Hardware Security Modules

ATM Network Security

- ATM security was the “killer app” that brought cryptography into the commercial mainstream
- Concrete security policy for APIs:

  “Only the customer should know her PIN”
- Standard PIN processing transactions, but multiple implementations from different vendors using hardware to keep PINs / keys from bank staff
- IBM made CCA manual available online
  - Excellent detailed description of API
  - Good explanation of background to PIN processing APIs
  - Unfortunately: lots of uncatalogued weaknesses.

HSM Use in Banks

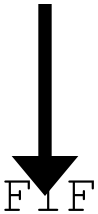


How are PINs Generated ?

Start with your bank account number (PAN)

5641 8203 3428 2218

Encrypt with PIN Derivation Key



22BD 4677 F1FF 34AC

Chop off the End

22 3

decimalise

(B->1)

(D->3)

How do I change my PIN?

- Default is to 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 I change PIN from 4426 to 1979, offset stored is 7553 (digit-by-digit modulo 10)

Offset Calculation Attack (1989)

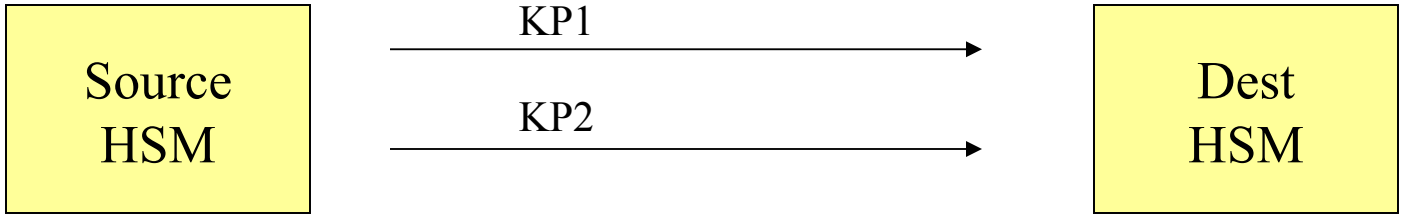
- Bank adds a new command to the API to calculate the offset between a new generated PIN and the customer's chosen PIN
- Possessing a bank account gives knowledge of one generated PIN. Any customer PIN could be revealed by calculating the offset between it and the known PIN

$U \rightarrow C$: Old PAN, Old offset, New PAN

$C \rightarrow U$: New offset

VSM Attack (2000)

- Top-level crypto keys exchanged between banks in several parts carried by separate couriers, which are recombined 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 : $\{KP1 \oplus KP2\}_{ZCMK}$

Idea: XOR To Null Key

- A single operator could feed in the same part twice, which cancels out to produce an ‘all zeroes’ test key. PINs could be extracted in the clear using this key

Combine components…

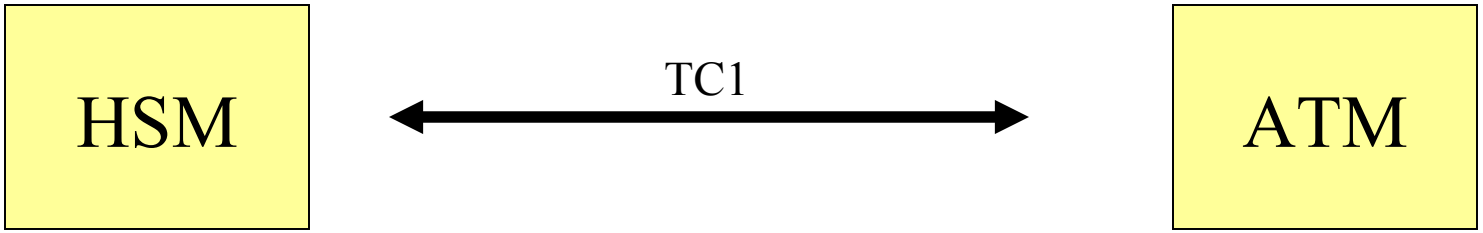
User $\rightarrow$ HSM : $\{KP1\}_{ZCMK}$, $\{KP1\}_{ZCMK}$

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

KP1 xor KP1 = 0

Type System Attack (2001)

- ATMs are simpler than HSMs and have only one master key. ATMs need to be sent Terminal Communications keys (session keys) for link cryptography.



Master Keys
TC – terminal communications
TMK – terminal master keys & PIN derivation keys
ZCMK – zone control master keys (between HSMs)
WK – working keys (session keys)
LP – local PIN storage key

Master Key
TMK-ATM - used for everything

but how?

$\{ TC1 \}_{TC}$ ----------> $\{ TC1 \}_{TMK-ATM}$

Type System Attack (2)

- PIN derivation keys (PDKs) share the same type as Terminal Master Keys (TMKs), and encrypting communication keys for transfer to an ATMs uses exactly the same process as calculating a customer PIN – encryption with single DES.

User->HSM : TC1

HSM->User : $\{ TC1 \}_{TC}$

User->HSM : $\{ TC1 \}_{TC}$, $\{ TMK\text{-}ATM \}_{TMK}$

HSM->User : $\{ TC1 \}_{TMK\text{-}ATM}$

The attack:

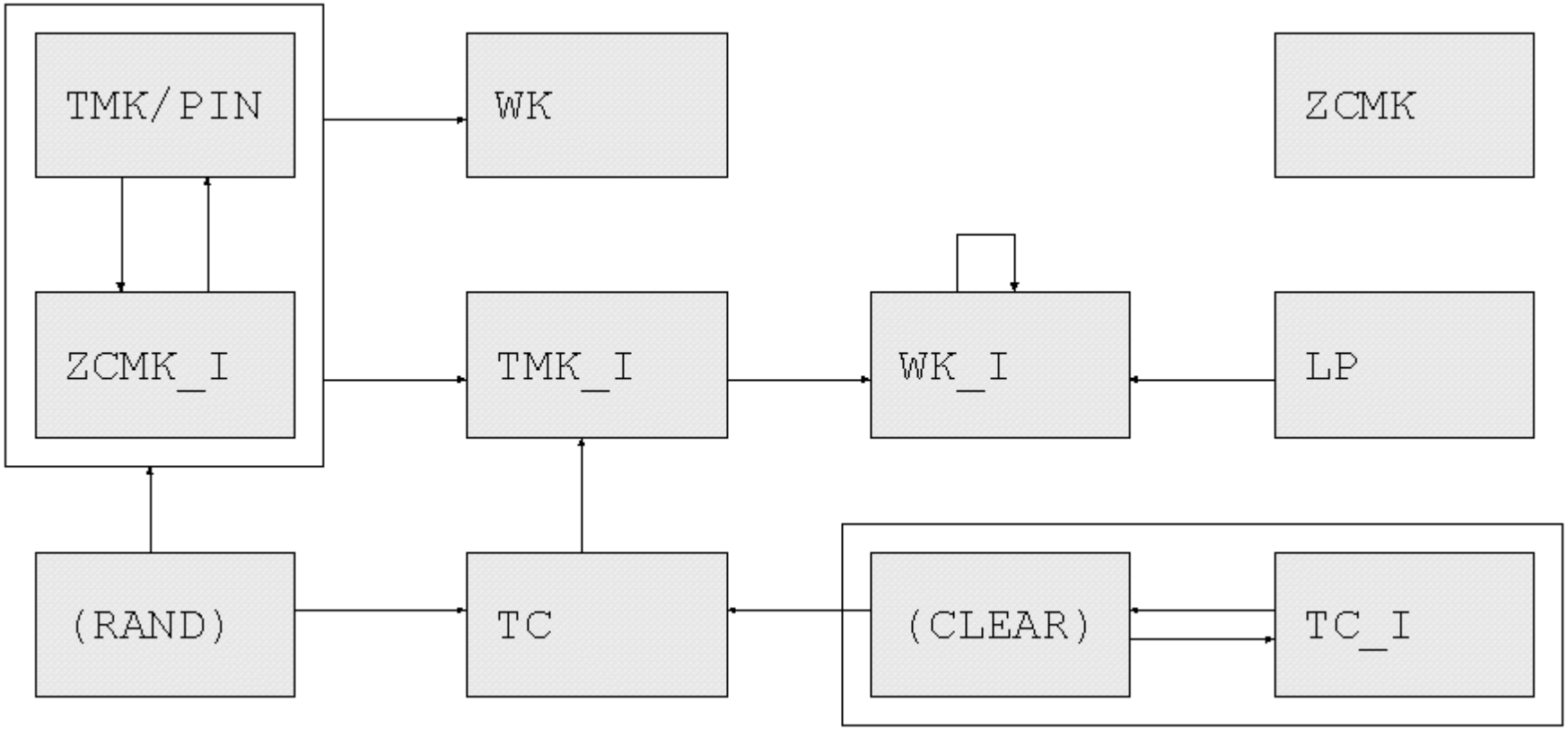
User->HSM : PAN

HSM->User : $\{ PAN \}_{TC}$

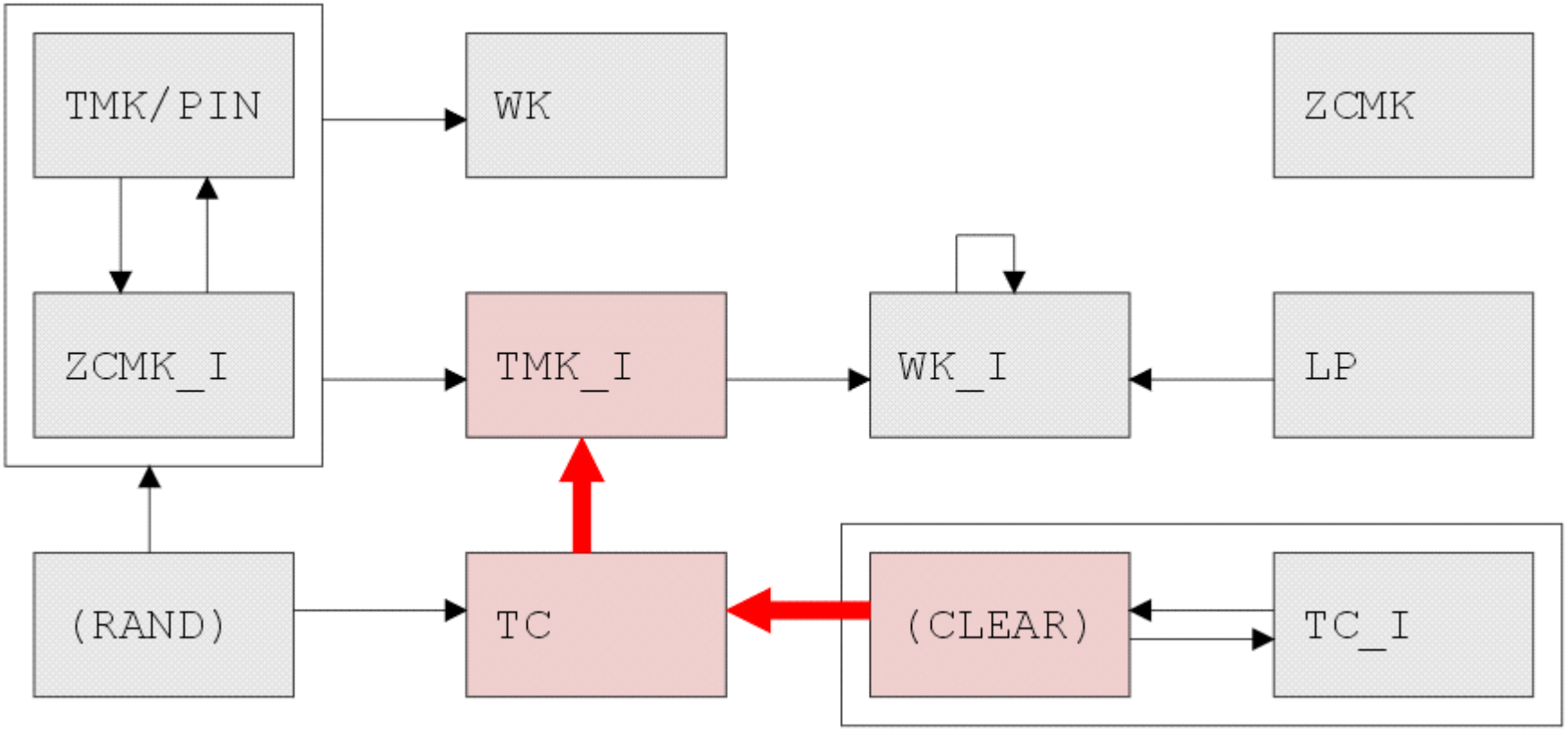
User->HSM : $\{ PAN \}_{TC}$, $\{ PDK1 \}_{TMK}$

HSM->User : $\{ PAN \}_{PDK1}$

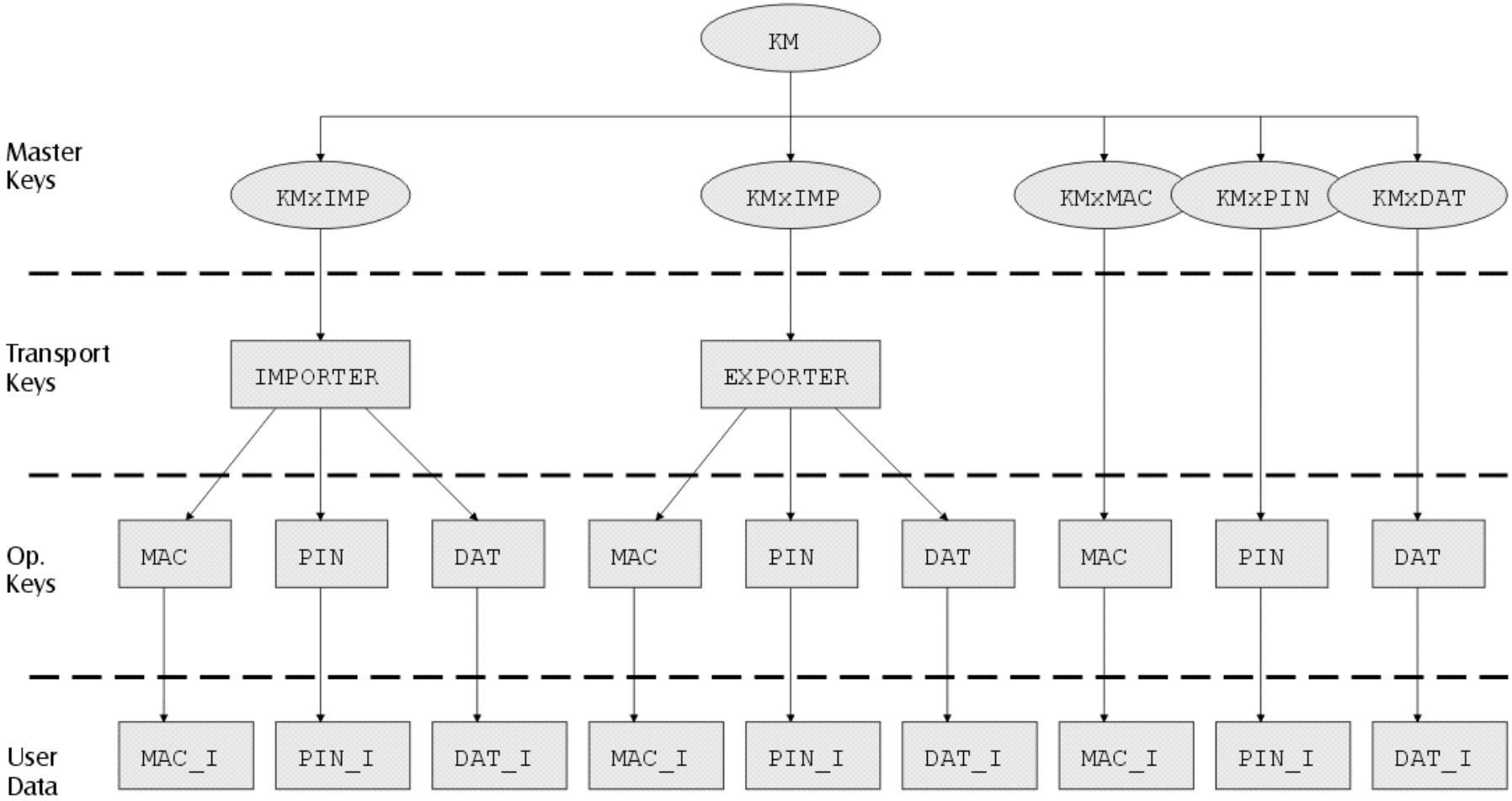
VSM Type Diagram



How Type-System Attack Was Found



IBM 4758 Key Hierarchy



Control Vectors

- IBM implementation, across many products since 1992, of the concept of ‘type’
- An encrypted key *token* looks like this :

$$E_{Km \oplus TYPE}(KEY), TYPE$$

Key Part Import

- Thee key-part holders, each have KPA, KPC, KPC
- Final key K is $KPA \oplus KPB \oplus KPC$
- All must collude to find K, but any one key-part holder can choose difference between desired K and actual value.

4758 Key Import Attack

$KEK1 = KORIG$

$KEK2 = KORIG \oplus (old\_CV \oplus new\_CV)$

Normally ...

$D_{KEK1 \oplus old\_CV}(E_{KEK1 \oplus old\_CV}(KEY)) = KEY$

Attack ...

$D_{KEK2 \oplus new\_CV}(E_{KEK1 \oplus old\_CV}(KEY)) = KEY$

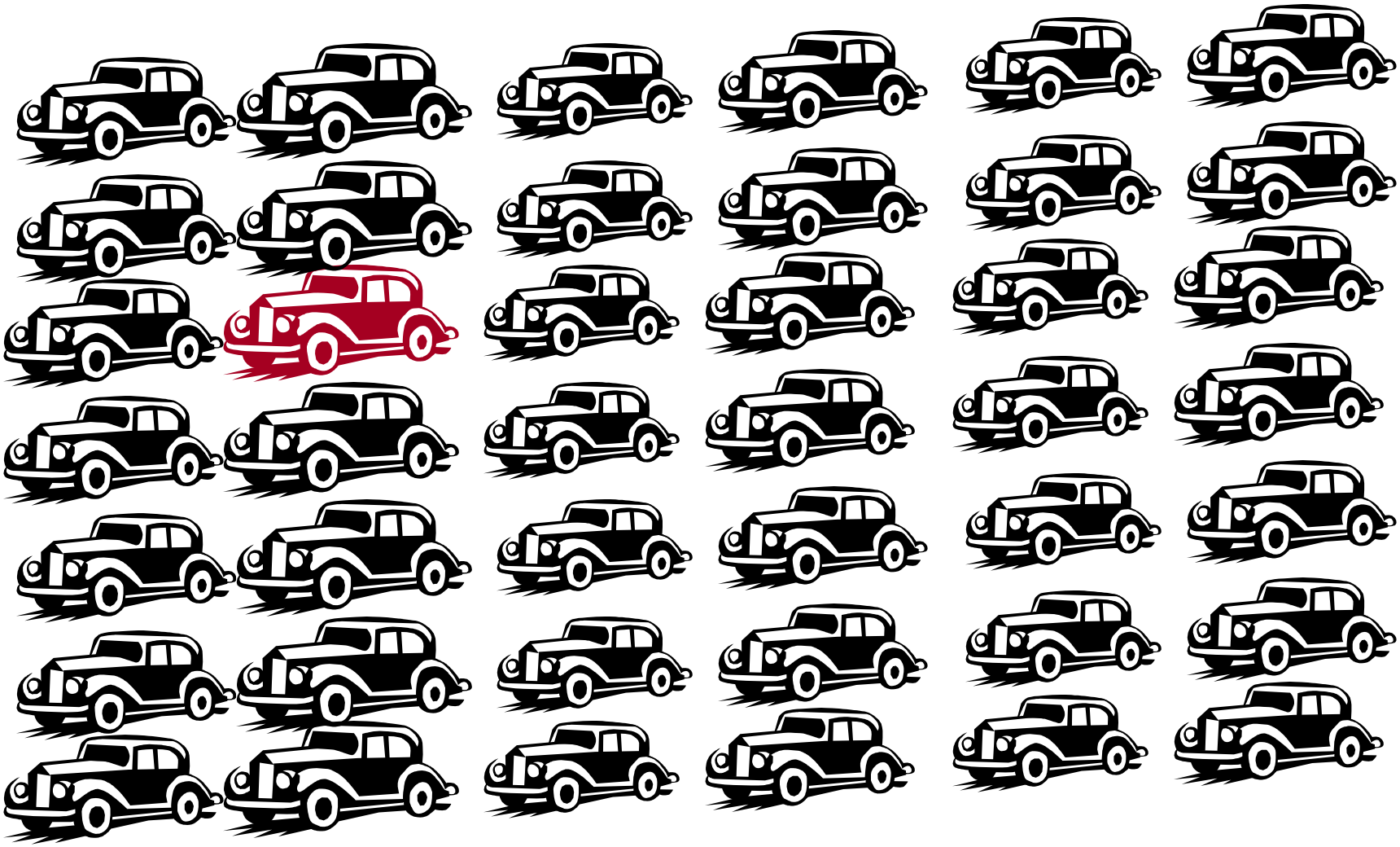
IBM had known about this attack, documented it obscurely, and then forgotten about it!

Collision-Search Attacks

- A thief walks into a car park and tries to steal a car...

- How many keys must he try?

Car Park 1929



Car Park 2009

Collision-Search Attacks (2)

- Capture-recapture statistics; also ‘meet in the middle’
- Attack multiple keys in parallel, given a ‘test vector’ (same plaintext encrypted under each key)
- Typical case: A 2^{56} search for one key becomes a 2^{40} search for any one of 2^{16} keys
- Any one key of a given type is usually enough - typical HSMs translate between keys of one type
- Poor implementations of 3DES (EK1, DK2, EK1) allow 3DES key halves to be attacked individually

Collision Search Attack on HSMs

- Generate 2^{16} keys
- Encrypt test vectors

U->C : { KEY1 }$_{KM}$

C->U : { 0000000000000000 }$_{KEY1}$

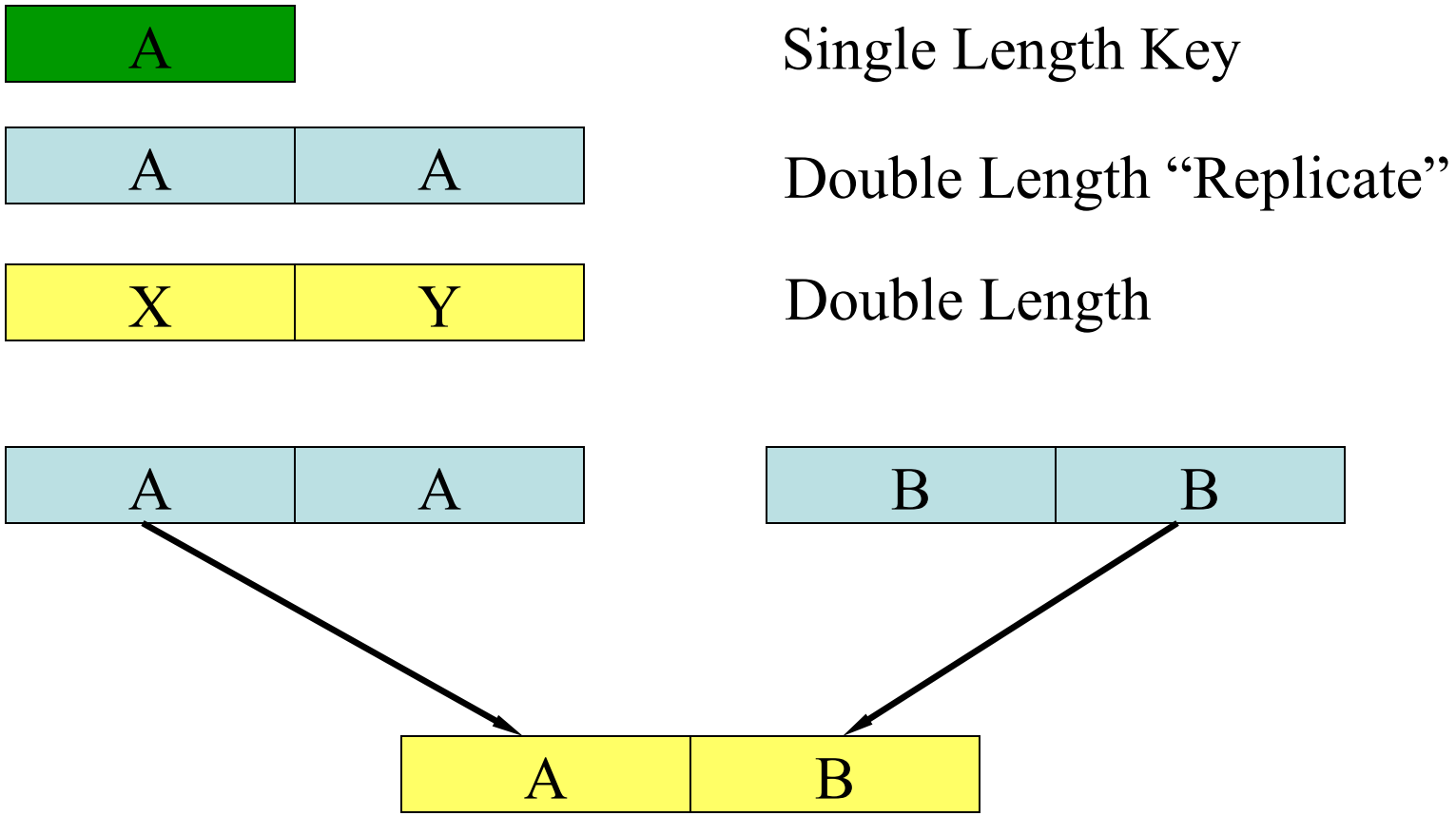
- Do 2^{40} search

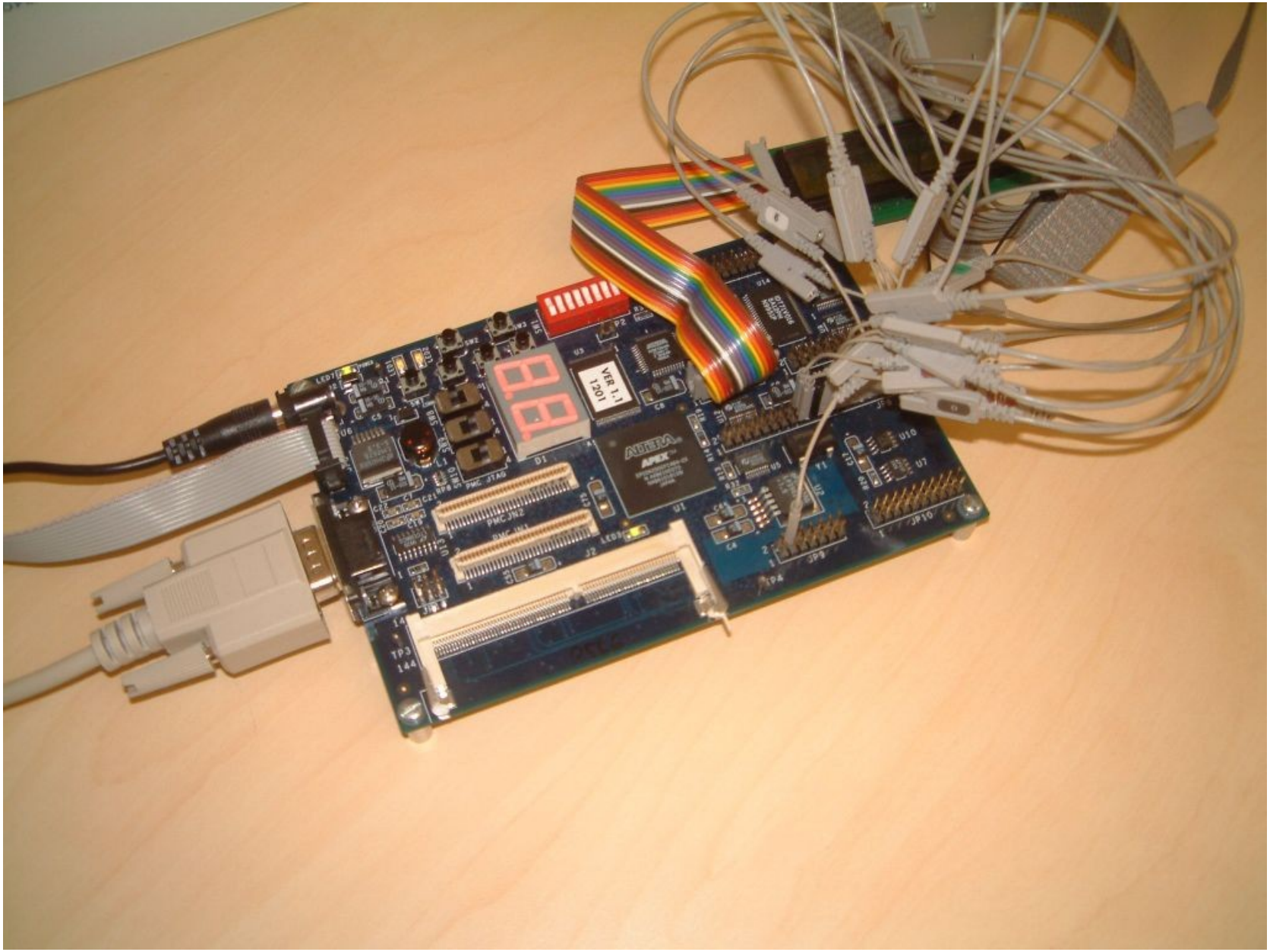
Cryptoprocessor's Effort → ← Search Machine's Effort

56 bit key space

Collision Search on 3DES

$$E_K(D_K(E_K(\ KEY\ )) = E_K(KEY)$$



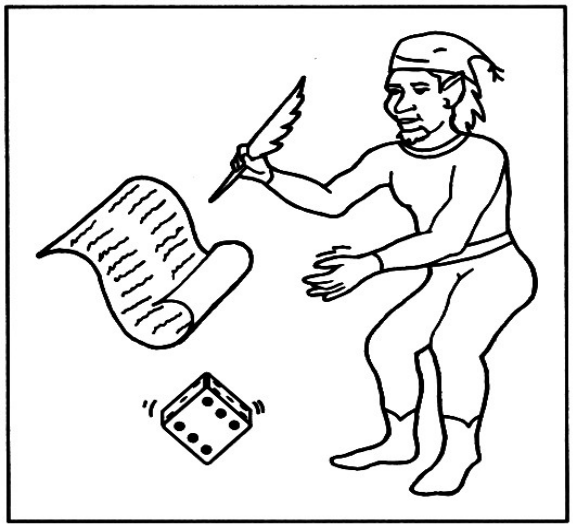

VER 1.1
1201
ALTERA
APEX
PMC_JN2
PMC_JN1
PMC_JTAG

A Framework for Crypto

- Cryptography (making), cryptanalysis (breaking), cryptology (both)
- Traditional cryptanalysis – what goes wrong with the design of the algorithms
- Then – what goes wrong with their implementation (power analysis, timing attacks)
- Then – what goes wrong with their use (we've already seen several examples)
- How might we draw the boundaries?

A Framework for Crypto (2)

- The ‘random oracle model” gives us an idealisation of ciphers and hash functions
- For each input, give the output you gave last time – and a random output if the input’s new



A Framework for Crypto (3)

- There are three basic ‘random oracle’ primitives
  - Stream ciphers have a fixed-length input (the key) and an unrestricted length output
  - Hash functions have an unrestricted length input and a fixed length output (the hash)
  - Block ciphers have fixed input and output. They are also invertible
- Block ciphers have an implicit key in this model; keyed hash functions may have too
- Random versus pseudorandom
- Let’s look at some historical examples

Stream Ciphers

- Julius Caesar: $c_i = p_i$ + ‘d’ (mod 24)

  `veni vidi vici`

  `ZHQM ZMGM ZMFM`

- Abbasid caliphate – monoalphabetic substition

  `abcdefghijklmno …`

  `SECURITYABDFGHI …`

- Solution: letter frequencies. Most common letters in English are e, t, a, I, o, n, s, h, r, d, l, u

Stream Ciphers (2)

- 16th century – the Vigenère

```
plaintext          tobeornottobethatistheques …
key          runrunrunrunrunrunrunrunru …
ciphertext KIOVIEEIGKIOVNURNVJNUVKHVM …
```

- Solution: patterns repeat at multiples of keylength (Kasiski, 1883) – here, ‘KIOV’
- Modern solution (1915): index of coincidence, the probability two letters are equal, $I_c = \sum p_i^2$
- This is 0.038 = 1/26 for random letters, 0.065 for English and depends on keylength for Vigenère

Stream Ciphers (3)

- The one-time pad was developed by Frank Miller (1882) then reinvented for use in WW1, then used in WW2 (and since)
- It's a Vigenère with an infinitely long key
- Provided the key is random and not reused or leaked, it's provably secure
- A spy caught having sent message X can claim he sent message Y instead, so long as he destroyed his key material!
- See Leo Marks, "Between Silk and Cyanide"

Stream Ciphers (4)

Plain:	`heilhitler`
Key:	`wclnbtdefj`
Cipher:	`DGTYIBWPJA`

Cipher:	`DGTYIBWPJA`
Key:	`wggsbtdefj`
Plain:	`hanghitler`

- The spy if caught can say he sent something completely different!
- But the flip side is that anyone who can manipulate the channel can turn any known message into any arbitrary one

Stream Ciphers (5)

- The Hagelin M-209 is one of many stream cipher machines developed in the 1920s and 30s
- Used by US forces in WW2

An Early Block Cipher – Playfair

P	A	L	M	E
R	S	T	O	N
B	C	D	F	G
H	I	K	Q	U
V	W	X	Y	Z

- Charles Wheatstone's big idea: encipher two letters at a time!
- Use diagonals, or next letters in a row or column
- Used by JFK in the PT boat incident in WW2

Plain:	lo rd gr an vi lx le sl et te rz
Cipher:	MT TB BN ES WH TL MP TA LN NL NV

Test Key Systems

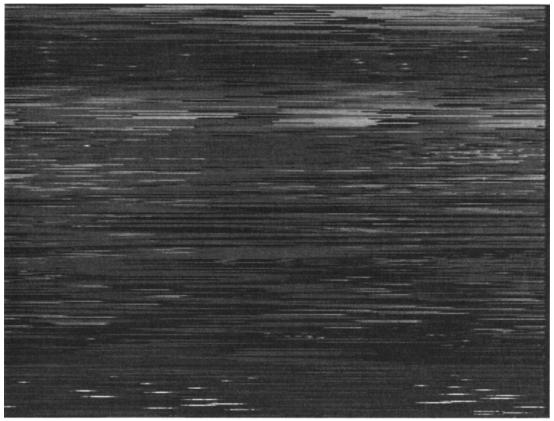
- Stream ciphers can’t protect payment messages – the plaintext is predictable, and telegraph clerks can be bribed
- So in the 19th century, banks invented ‘test key’ systems – message authentication codes using secret tables
- Authenticator for £276,000 = 09+29+71 = 109

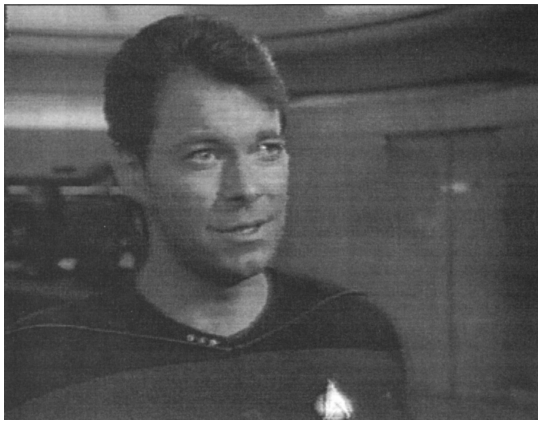
	0	1	2	3	4	5	6	7	8	9
x 1000	14	22	40	87	69	93	71	35	06	58
x 10,000	73	38	15	46	91	82	00	29	64	57
x 100,000	95	70	09	54	82	63	21	47	36	18
x 1,000,000	53	77	66	29	40	12	31	05	87	94

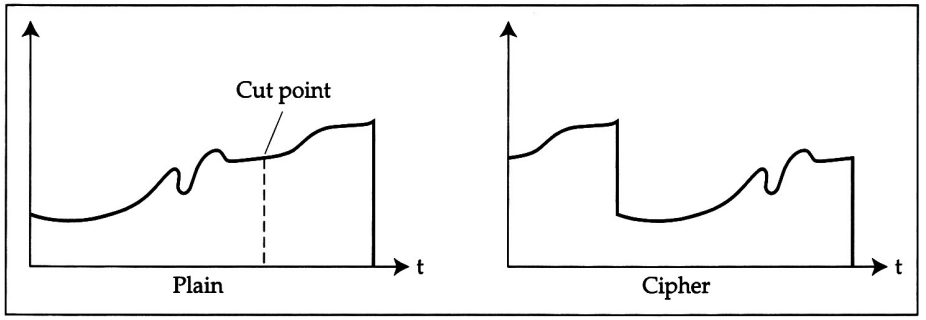
Modern Cipher Systems

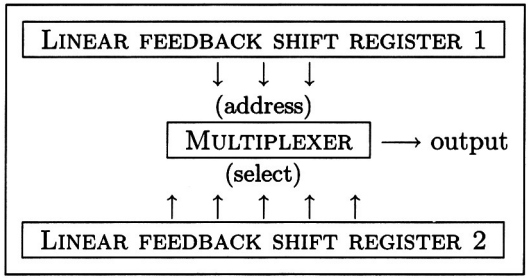
- Many systems from the last century use stream ciphers for speed / low gate count
- Bank systems use a 1970s block cipher, the data encryption standard or DES; recently moving to triple-DES for longer keys
- New systems mostly use the Advanced Encryption Standard (AES), regardless of whether a block cipher or stream cipher is needed
- For hashing, people use SHA, but this is getting insecure; a new hash function is underway and in the meantime people use SHA-256

Stream Cipher Example – Pay-TV









The old Sky-TV system

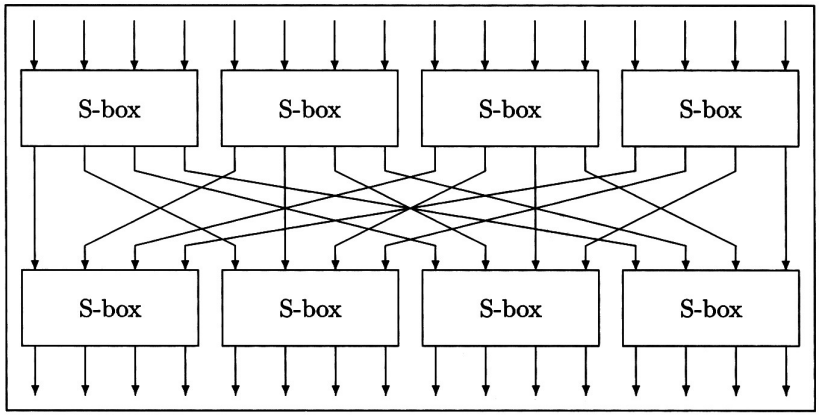
Stream Cipher Example – GSM

- WEP (and SSL/TLS) use RC4, a table shuffler a bit like rotor machines

```
i:= i+1          (mod 256)
j:= j+s[i]       (mod 256)
swap(s[i],s[j])
t:= s[i]+s[j]    (mod 256)
k:= s[t]
```

- RC4 encryption is fairly strong because of the large state space – but in WEP the algo used to set up the initial state of the table s[i] is weak (24-bit IVs are too short)
- Result: break WEP key given tens of thousands of packets

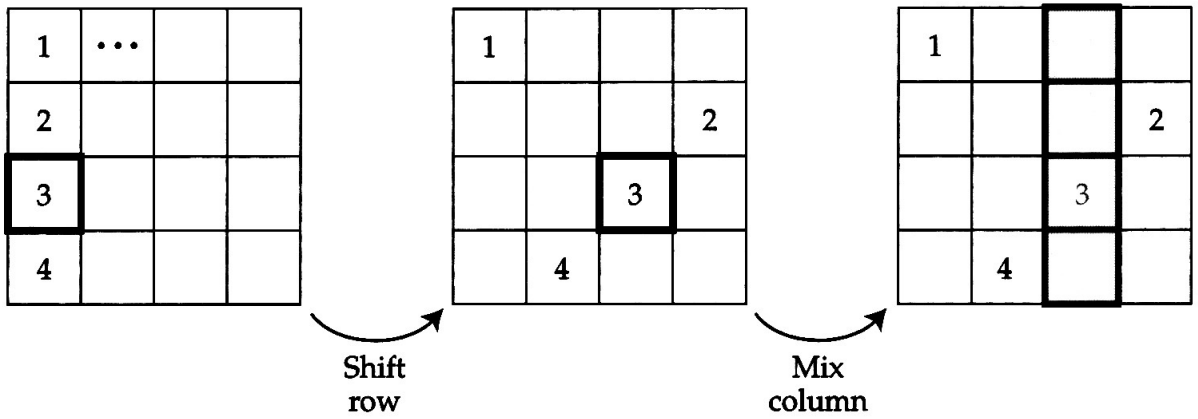
Block Cipher – Basic Idea



- Shannon (1948) – iterate substitution, permutation
- Each output bit depends on input, key in complex way
- E.g. our AES candidate algorithm Serpent – 32 4-bit S-boxes wide, 32 rounds; 128-bit block, 256-bit key
- Security – ensure block and key size large enough; that linear approximations don’t work (linear cryptanalysis), nor bit-twiddling either (differential cryptanalysis)

The Advanced Encryption Standard

- AES has a 128-bit block, arranged as 16 bytes
- Each round: shuffle bytes as below, xor key bytes, then bytewise S-box $S(x) = M(1/x) + b$ in GF(2^8)
- 10 rounds for 128-bit keys; 12 for 192, 14 for 256
- Only ‘certificational’ attacks are known (e.g. 2^{119} effort attack against 256-bit keys)



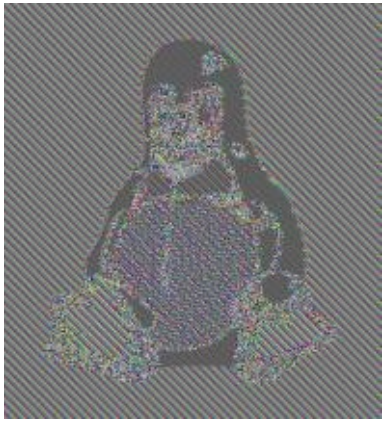
The Data Encryption Standard

- DES was standardised in 1977; it's widely used in banking, and assorted embedded stuff
- Internals: a bit more complex than AES (see book)
- Shortcut attacks exist but are not important:
  - differential cryptanalysis (2^{47} chosen texts)
  - linear cryptanalysis (2^{41} known texts)
- 64-bit block size, hinders upgrade to AES
- 56-bit keys – keysearch is the real vulnerability!

Keysearch

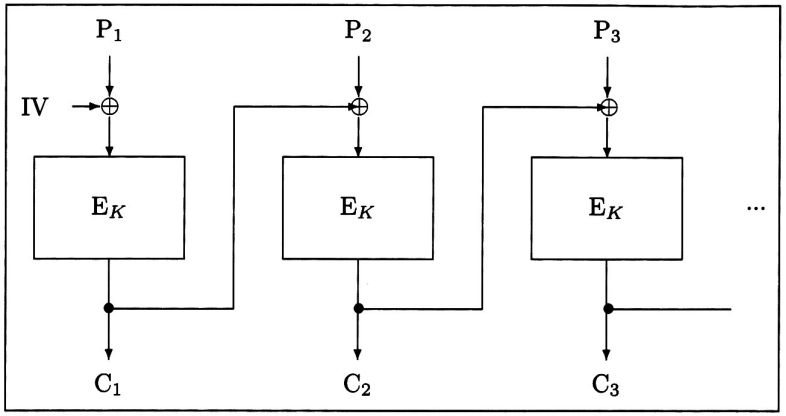
- DES controversy in 1977 – 1M chips, 1Mkey/s, 2^{15} sec: would the beast cost $10m or $200m?
- Distributed volunteers (1997) – 5000 PCs
- Deep Crack (1998) – $250K (1000 FPGAs), 56 h
- 2005 – single DES withdrawn as standard
- Copacabana (2006) – $10K of FPGAs, 9 h
- Even 64-bit ciphers such as A5/3 (Kasumi) used in 3g are now vulnerable to military kit
- Banks moving to 3DES (EDE for compatibility)

Modes of Operation



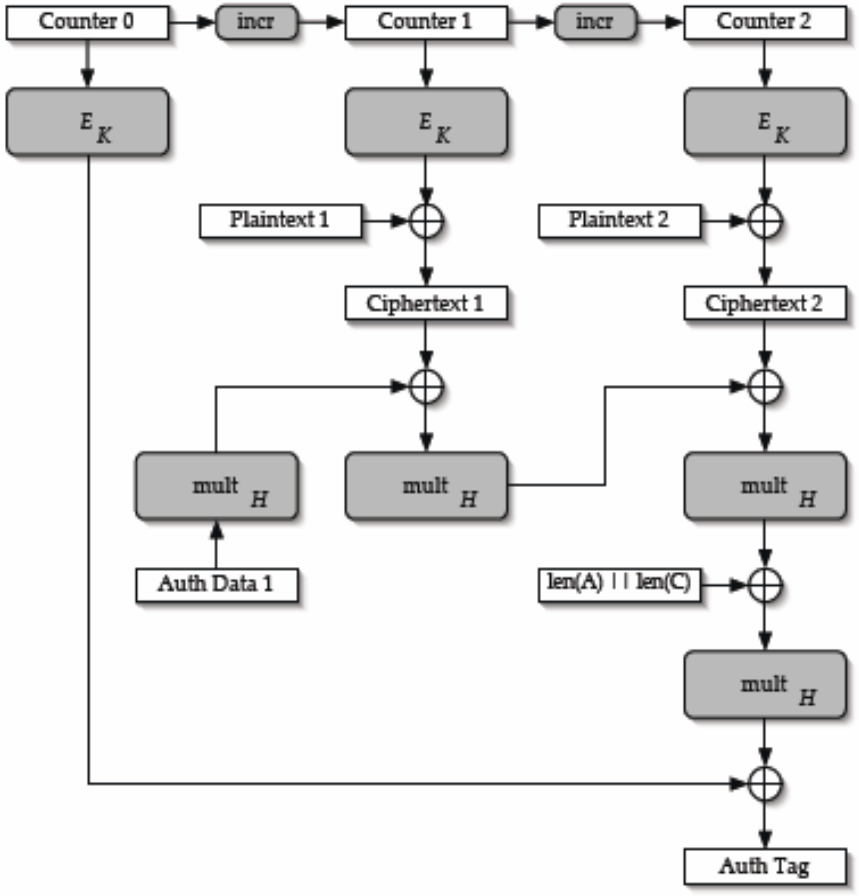
- ECB – electronic codebook – mode just encrypts a block at a time
- Patterns can still be fairly obvious
- In 1b, you saw other modes that can be used to hide them – and do other things too

Modes of Operation (2)



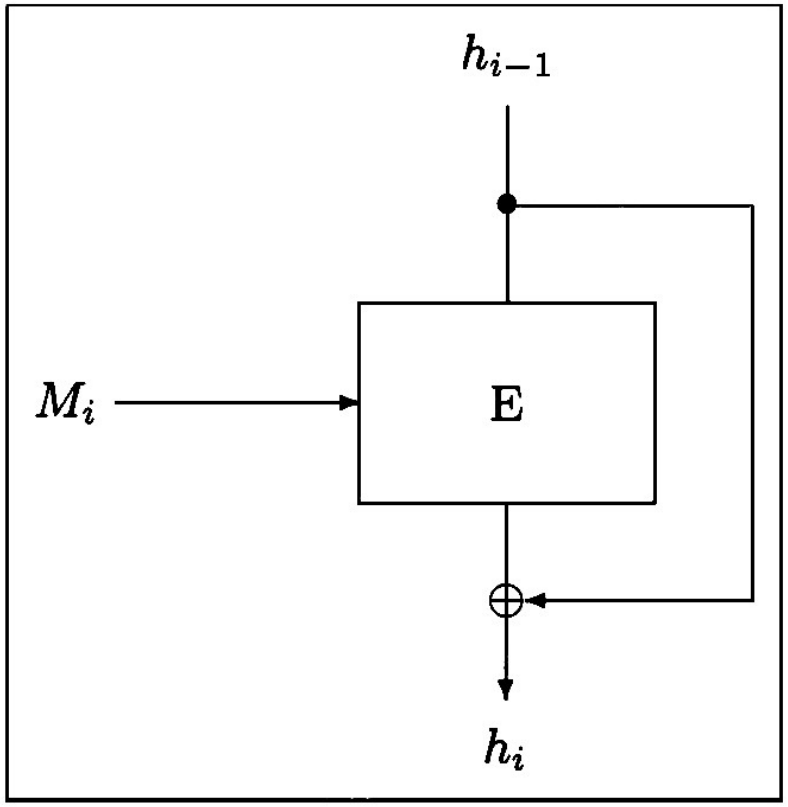
- Cipher block chaining (CBC) was the traditional mode for bulk encryption
- It can also be used to compute a message authentication code (MAC)
- But it can be insecure to use the same key for MAC and CBC (why?), so this is a 2-pass process

Modes of Operation (3)



- Counter mode (encrypt a counter to get keystream)
- New (2007) standard: Galois Counter Mode (GCM)
- Encrypt an authenticator tag too
- Unlike CBC / CBC MAC, one encryption per block – and parallelisable!
- Used in SSH, IPSEC, …

Modes of Operation (4)



- Feedforward mode turns a block cipher into a hash function
- Input goes into the key port
- The block size had better be more than 64 bits though!
- (Why?)

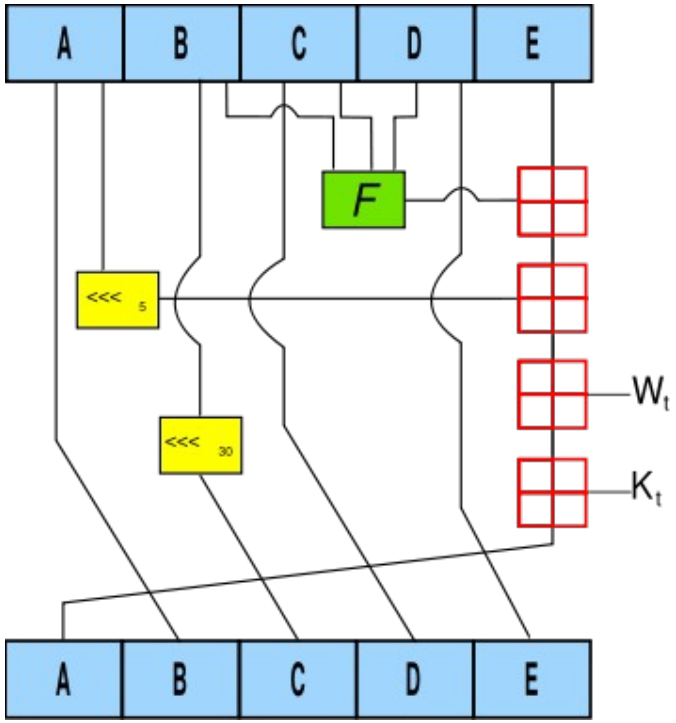
Hash Functions

- A cryptographic hash function distills a message M down to a hash h(M)
- Desirable properties include:
  1. Preimage resistance – given X, you can't find M such that h(M) = X
  2. Collision resistance – you can't find M1, M2 such that h(M1) = h(M2)
- Applications include hashing a message before digital signature, and computing a MAC

Hash Functions (2)

- Common hash functions use feedforward mode of a special block cipher – big block, bigger ‘key’
- MD5 (Ron Rivest, 1991): still widely used, has 128-bit block. So finding a collision would take about 2^{64} effort if it were cryptographically sound
- Flaws found by Dobbertin and others; collision existence by 2004; fake SSL certificates by 2005 (two public keys with same MD5 hash); now collision attack takes only a minute
- Next design was SHA

Hash Functions (3)



- NSA produced the secure hash algorithm (SHA or SHA1), a strengthened version of MD5, in 1993
- 160-bit hash – the underlying block cipher has 512-bit key, 160-bit block, 80 rounds
- One round shown on left

Hash Functions (4)

- At Crypto 2005, a 2^{69} collision attack on SHA was published by Xiaoyun Wang et al
- As an interim measure, people are moving to SHA256 (256-bit hash, modified round function) or for the paranoid SHA512
- There's a competition underway, organised by NIST, to find 'SHA3'

Hash Functions

- If we want to compute a MAC without using a cipher (e.g. to avoid export controls) we can use HMAC (hash-based message authentication code):

  HMAC(k,M) = h(k_1, h(k_2, M))

  where k_1 = k xor 0x5c5c5c…5c5c, and k_2 = 0x363636…3636 (why?)
- Another app is tick payments – make a chain h_1 = h(X), h_2 = h(h_1), … ; sign h_k; reveal h_{k-1}, h_{k-2}, … to pay for stuff
- A third is timestamping; hash all the critical messages in your organisation in a tree and publish the result once a day

Advanced Crypto Engineering

- Once we move beyond ‘vanilla’ encryption into creative used of asymmetric crypto, all sorts of tricks become possible
- It’s also very easy to shoot your foot off!
- Framework:
  - What’s tricky about the maths
  - What’s tricky about the implementation
  - What’s tricky about the protocols etc
- To roll your own crypto, you need specialist help

Public Key Crypto Revision

- Digital signatures: computed using a private signing key on hashed data
- Can be verified with corresponding public verification key
- Can't work out signing key from verification key
- Typical algorithms: DSA, elliptic curve DSA
- We'll write $sig_A\{X\}$ for the hashed data X signed using A's private signing key

Public Key Crypto Revision (2)

- Public key encryption lets you encrypt data using a user's public encryption key
- She can decrypt it using her private decryption key
- Typical algorithms Diffie-Hellman, RSA
- We'll write $\{X\}_A$
- Big problem: knowing whose key it is!

PKC Revision – Diffie-Hellman

- Diffie-Hellman: underlying metaphor is that Anthony sends a box with a message to Brutus
- But the messenger's loyal to Caesar, so Anthony puts a padlock on it
- Brutus adds his own padlock and sends it back to Anthony
- Anthony removes his padlock and sends it to Brutus who can now unlock it
- Is this secure?

PKC Revision – Diffie-Hellman (2)

- Electronic implementation:

  $A \rightarrow B: \quad M^{rA}$

  $B \rightarrow A: \quad M^{rArB}$

  $A \rightarrow B: \quad M^{rB}$

- But encoding messages as group elements can be tiresome so instead Diffie-Hellman goes:

  $A \rightarrow B: \quad g^{rA}$

  $B \rightarrow A: \quad g^{rB}$

  $A \rightarrow B: \quad \{M\}g^{rArB}$

PKC Revision – El Gamal

- Encryption – DH can use long-term keys, say private key xA and public key yA = g^{xA}
- The Bob looks up yA and makes the long-term shared key $yA^{xA} = g^{xAxB} = yB^{xA}$
- In El Gamal, combine with a transient private key k
- Bob encrypts M as $M.yA^{k}$, g^{k}
- Alice decrypts by forming yA^{k} as $(g^{k})^{xA}$

PKC Revision – El Gamal (2)

- Signature trick: given private key xA and public key $yA = g^{xA}$, and transient private key k and transient public key $r = g^{k}$, form the private equation

  $rxA + sk = m$

- The digital signature on m is (r,s)
- Signature verification is

  $g^{(rxA + sk)} = g^{m}$

- i.e. $yA^{r}.r^{s} = g^{m}$

PKC Revision – DSS

- The Digital Signature Standard is ElGamal with a few technical weaknesses fixed
- p: a prime of 1024 bits; q: a prime dividing p-1; g: an element of order q in the integers mod p
- Signature on m is (r,s) such that

  $r = (g^k \bmod p) \bmod q$

  $h((M) = xAr + ks$
- Verification: exercise
- Only known vuln: choose q = h(M1) - h(M2)

Public Key Crypto Revision (3)

- One way of linking public keys to principals is for the sysadmin to physically install them on machines (common with SSH, IPSEC)
- Another is to set up keys, then exchange a short string out of band to check you're speaking to the right principal (STU-II, Bluetooth simple pairing)
- Another is certificates. Sam signs Alice's public key (and/or signature verification key)

  $CA = sig_S\{T_S,L,A,K_A,V_A\}$

- But this is still far from idiot-proof…

The Denning-Sacco Protocol

- In 1982, Denning and Sacco pointed out the revocation problem with Needham-Schroder and argued that public key should be used instead

  $A \rightarrow S$: A, B

  $S \rightarrow A$: CA, CB

  $A \rightarrow B$: CA, CB, $\{sig_A\{T_A, K_{AB}\}\}_{KB}$

- What's wrong?

The Denning-Sacco Protocol (2)

- Twelve years later, Abadi and Needham noticed that Bob can now masquerade as Alice to anyone in the world!

  $A \rightarrow S$: A, B

  $S \rightarrow A$: CA, CB

  $A \rightarrow B$: CA, CB, $\{sig_A\{T_A, K_{AB}\}\}_{KB}$

  $B \rightarrow S$: B, C

  $S \rightarrow B$: CB, CC

  $B \rightarrow C$: CA, CC, $\{sig_A\{T_A, K_{AB}\}\}_{KC}$

Encrypting email

- Standard way (PGP) is to affix a signature to a message, then encrypt it with a message key, and encrypt the message with the recipient's public key

  $A \rightarrow B$: $\{KM\}_B$, $\{M, sig_A\{h(M)\}\}_{KM}$

- X.400 created a detached signature

  $A \rightarrow B$: $\{KM\}_B$, $\{M\}_{KM}$, $sig_A\{h(M)\}$

- And with XML you can mix and match… e.g. by signing encrypted data. Is this good?

Public-key Needham-Schroeder

- Proposed in 1978:

  $A \rightarrow B: \{NA, A\}_{KB}$

  $B \rightarrow A: \{NA, NB\}_{KA}$

  $A \rightarrow B: \{NB\}_{KB}$

- The idea is that they then use $NA \oplus NB$ as a shared key
- Is this OK?

Public-key Needham-Schroeder (2)

- Attack found eighteen years later, in 1996:

  $A \rightarrow C$: $\{NA, A\}_{KC}$

  $C \rightarrow B$: $\{NA, A\}_{KB}$

  $B \rightarrow C$: $\{NA, NB\}_{KA}$

  $C \rightarrow A$: $\{NA, NB\}_{KA}$

  $A \rightarrow C$: $\{NB\}_{KC}$

  $C \rightarrow B$: $\{NB\}_{KB}$

- Fix: explicitness. Put all names in all messages

Public Key Protocol Problems

- It’s also very easy to set up keys with the wrong people – man-in-the-middle attacks get more pervasive. Assumptions are slippery to pin down
- Technical stuff too – if the math is exposed, an attacker may use it against you!
- So data being encrypted (or signed) must be suitably packaged
- Many other traps, some extremely obscure…

PKC Revision – RSA

- Recall from 1a discrete maths: private key is two large primes p, q
- Public key is n = pq plus public exponent e
- Encryption: $c = m^e \pmod n$
- Decryption: $m = c^d \pmod n$
- This works iff $de = 1 \pmod{(p-1)(q-1)}$
- Proof: $m^{ed} = m^{(1+k(p-1)(q-1))} = m.1 \pmod n$ by Euler's theorem
- Similarly signature $s = m^d \pmod n$

Extra Vulnerabilities of RSA

- Decryption = signature, so ‘sign this to prove who you are’ is really dangerous
- Multiplicative attacks: if m3 = m1.m2 then s3 = s1.s2 – so it’s even more important to hash messages before signature
- Also before encrypting: break multiplicative pattern by ‘Optimal asymmetric encryption padding’. Process key k and random r to X, Y as

  $X = m \oplus h(r)$

  $Y = r \oplus h(X)$

Fancy Cryptosystems (1)

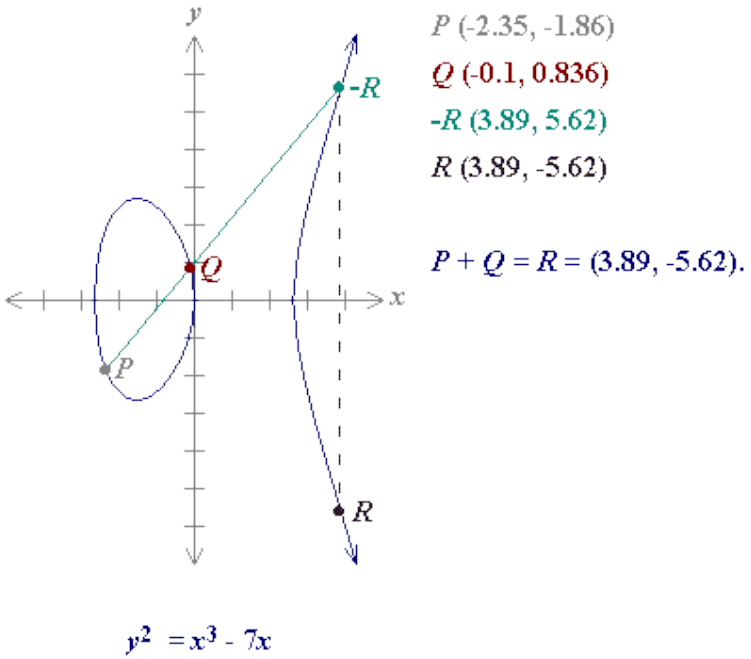
- Shared control: if all three directors of a company must sign a cheque, set d = d1 + d2 + d3
- Threshold cryptosystems: if any k out of l directors can sign, choose polynomial P(x) such that P(0) = d and deg(P) = k-1. Give each a point xi, P(xi)
- Lagrange interpolation: $P(z) = \sum xi \prod (z-xi)/(xj-xi)$
- So signature $h(M)^{P(0)} = h(M)^{\sum xi \prod (z-xi)/(xj-xi)}$

$= \prod h(M)^{xi \prod (...)}$

Fancy Cryptosystems (2)

- Identity-based cryptosystems: can you have the public key equal to your name?
- Signature (Fiat-Shamir): let the CA know the factors p, q of n. Let si = h(name,i), and the CA gives you $\sigma i = \sqrt{si} \pmod n$
- Sign M as r^2, $s = r\prod_{hi(M)=1} \sigma i \pmod n$ where hi(M) is 1 if the ith bit of M is one, else 0
- Verify: check that $r^2\prod_{hi(M)=1} si = s^2 \pmod n$
- (Why is the random salt r used here, not just the raw combinatorial product?)

Fancy Cryptosystems (3)



- Elliptic curve cryptosystems use a group of points on an elliptic curve $y^2 = x^3 + ax + b$ rather than a group mod p
- Group law: if P, Q, R on a line then P+Q+R = 0 (the point at ∞)
- DH, DSA etc go over

Fancy Cryptosystems (4)

- Elliptic curve crypto makes it even harder to choose good parameters (curve, generator)
- Also: a lot of implementation techniques are covered by patents held by Certicom
- OTOH: you can use smaller parameter sizes, e.g. 128-bit keys for equivalent of 64-bit symmetric keys, 256-bit for equivalent of 128
- Encryption, signature run much faster
- Being specified for next-generation Zigbee
- Also: can do tricks like identity-based encryption

Fancy Cryptosystems (5)

- Identity-based encryption: some pairs of elliptic curves have ‘bilinear pairing’ $G_1 \times G_1 \rightarrow G_2$ such that $e(aP,bQ) = e(P,Q)^{ab}$
- System secret s; public point P on G_1; public key $W = sP$; user public key $g_{ID} = e(h(ID),W)$; private key $d_{ID} = sID$
- Encrypt M: $C = (rW, M \oplus h(g_{ID}^{\ r}) = (U,V)$
- Decrypt U,V: $M = V \oplus h(e(d_{ID},U))$

$= V \oplus h(e(sID,rW))$

$= V \oplus h(e(ID,W)^r)$

$= V \oplus h(g_{ID})^r$

Fancy Cryptosystems (6)

- Forward secure encryption – equipment capture should not compromise old traffic
  - First option: DH to create transient key, then authenticate this
  - Second option (US Defense Messaging System): create one-time ElGamal keys signed using your DSA key and serve them up
  - Third option: use an identity-bases scheme to create a ‘key of the day’ for each future day and destroy the corresponding private keys as they expire
- Can trade algorithms / interactivity / infrastructure

Fancy Cryptosystems (7)

- Blind signatures: suppose Alice wants Bob the banker to sign a banknote without knowing its serial number. With RSA she sends him

  $$M' = M.R^e \pmod{n}$$

- He sends her $S' = M'^d \pmod{n}$
- She divides by R to recover $M^d \pmod{n}$
- Such ‘digital cash’ in general illegal, but similar ideas used in digital elections, and in crypto toolkits to combat side-channel attacks

General Problems with PKC

- Keys need to be long – we can factor / do discrete log to about 700 bits. For DSA/RSA, 1024 is marginal, 2048 considered safe for now
- Elliptic curve variants can use shorter keys – but are encumbered with patents
- Computations are slow – several ms on Pentium, almost forever on 8051 etc
- Power analysis is a big deal: difference between squaring and doubling is visible. Timing attacks too
- For many applications PKC just isn’t worth it

TLS

- Formerly SSL, became TLS after many bugs fixed:

  $C \rightarrow S$: C, C#, N_C ‘client hello’

  $S \rightarrow C$: S, S#, N_S CS ‘server hello’

  $C \rightarrow S$: $\{k_0\}_{KS}$ ‘k_0 = pre-master secret’

  $C \rightarrow S$: {finished, MAC_{K1}(everything to date)}

  $S \rightarrow C$: {finished, MAC_{K2}(everything to date)}

  K1, K2 hashed from ‘master secret’ K1 = $h(k_0, N_C, N_S)$

- Formally verified to ‘work’ but still often used inappropriately (more later…)

TLS (2)

- Why doesn’t TLS stop phishing?
  - Noticing an ‘absent’ padlock is hard
  - Understanding URLs is hard
  - Websites train users in bad practice
  - …
- In short, TLS as used in e-commerce dumps compliance costs on users, who can’t cope
- There are solid uses for it though

Chosen protocol attack

- Suppose that we had a protocol for users to sign hashes of payment messages (such a protocol was proposed in 1990s):

  $C \rightarrow M$: order

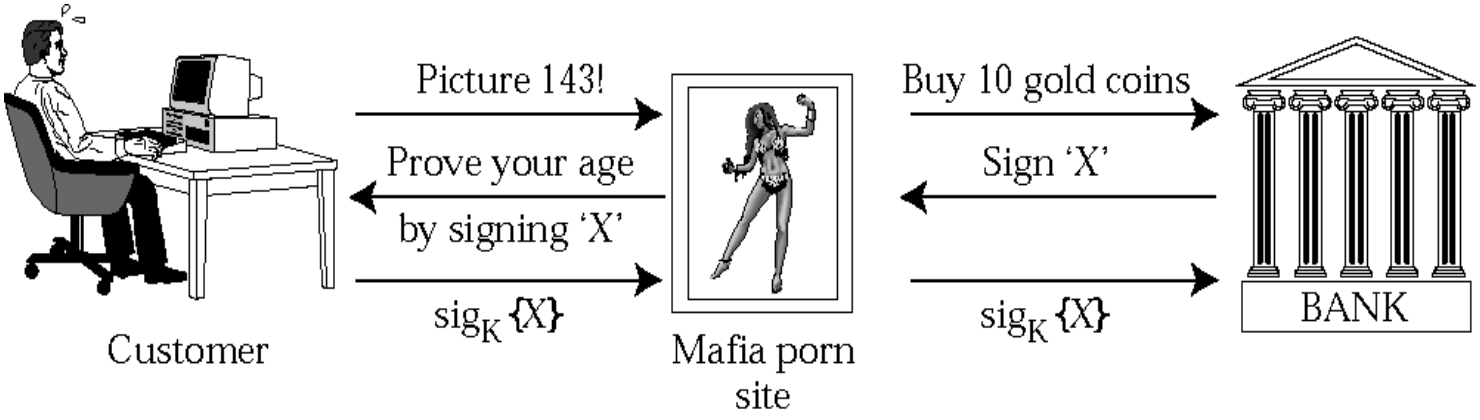
  $M \rightarrow C$: X [= hash(order, amount, date, …)]

  $C \rightarrow M$: $sig_K\{X\}$

- How might this be attacked?

Chosen protocol attack (2)

The Mafia demands you sign a random challenge to prove your age for porn sites!



Foot-Shooting Prevention Agreement

I, _____ , promise that once
Your Name
I see how simple AES really is, I will not implement it in production code even though it would be really fun.

This agreement shall be in effect until the undersigned creates a meaningful interpretive dance that compares and contrasts cache-based, timing, and other side channel attacks and their countermeasures.

X _____ _____
Signature Date

Building a Crypto Library is Hard!

- Sound defaults: AES GCM for encryption, SHA256 for hashing, PKC with long enough keys
- Defend against power analysis, fault analysis, timing analysis , and other side-channel attacks. This is nontrivial!
- Take great care with the API design
- Don’t reuse keys – ‘leverage considered harmful’!
- My strong advice: do not build a crypto library! If you must, you need specialist (PhD-level) help
- But whose can you trust?

How Certification Fails

- PEDs ‘evaluated under the Common Criteria’ were trivial to tap
- GCHQ wouldn’t defend the brand
- APACS said (Feb 08) it wasn’t a problem
- It sure is now…

Cryptographic Engineering 19c

- Auguste Kerckhoffs' six principles, 1883
  - The system should be hard to break in practice
  - It should not be compromised when the opponent learns the method – security must reside in the choice of key
  - The key should be easy to remember & change
  - Ciphertext should be transmissible by telegraph
  - A single person should be able to operate it
  - The system should not impose mental strain
- Many breaches since, such as Tannenberg (1914)

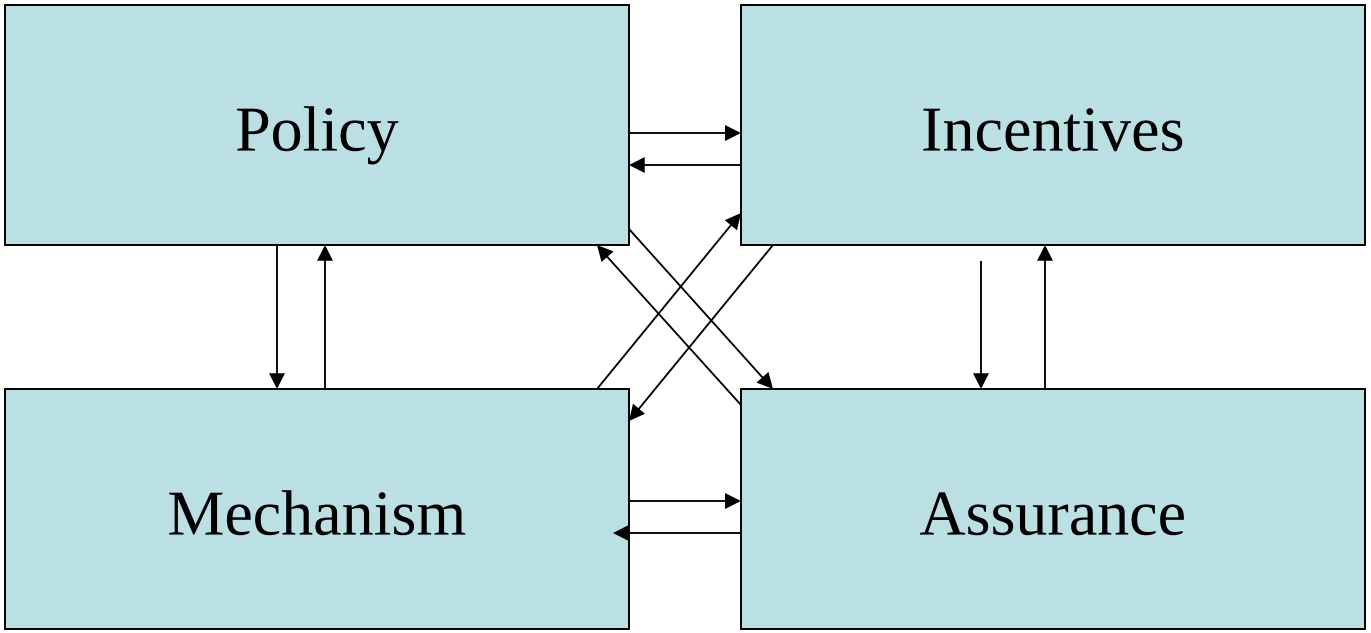
What else goes wrong

- See ‘Why cryptosystems fail’, my website (1993):
  - Random errors
  - Shoulder surfing
  - Insiders
  - Protocol stuff, like encryption replacement
- Second big wave now (see current papers):
  - ATM skimmers
  - Tampered PIN entry devices
  - Yes cards and other protocol stuff
  - Watch this space!

Security Engineering

- No different in essence from any other branch of system engineering
  - Understand the problem (threat model)
  - Choose/design a security policy
  - Build, test and if need be iterate
- Failure modes:
  - Solve wrong problem / adopt wrong policy
  - Poor technical work
  - Inability to deal with evolving systems
  - Inability to deal with conflict over goals

A Framework



Security Economics Example – Facebook

- Clear conflict of interest
  - Facebook wants to sell user data
  - Users want feeling of intimacy, small group, social control
- Complex access controls – 60+ settings on 7 pages
- Privacy almost never salient (deliberately!)
- Over 90% of users never change defaults
- This lets Facebook blame the customer when things go wrong

Conflict theory

- Does the defence of a country or a system depend on the least effort, on the best effort, or on the sum of efforts?
- The last is optimal; the first is really awful
- Software is a mix: it depends on the worst effort of the least careful programmer, the best effort of the security architect, and the sum of efforts of the testers
- Moral: hire fewer better programmers, more testers, top architects

How Much to Spend?

- How much should the average company spend on information security?
- Governments, vendors say: much much more than at present
- But they've been saying this for 20 years!
- Measurements of security return-on-investment suggest about 20% p.a. overall
- So the total expenditure may be about right. Are there any better metrics?

Skewed Incentives

- Why do large companies spend too much on security and small companies too little?
- Research shows an adverse selection effect
- Corporate security managers tend to be risk-averse people, often from accounting / finance
- More risk-loving people may become sales or engineering staff, or small-firm entrepreneurs
- There's also due-diligence, government regulation, and insurance to think of

Skewed Incentives (2)

- If you are DirNSA and have a nice new hack on XP and Vista, do you tell Bill?
- Tell – protect 300m Americans
- Don't tell – be able to hack 400m Europeans, 1000m Chinese,…
- If the Chinese hack US systems, they keep quiet. If you hack their systems, you can brag about it to the President
- So offence can be favoured over defence

More …

- See www.ross-anderson.com for a survey article, ENISA report, and pages on security economics and security psychology
- WEIS – Workshop on Economics and Information Security
- Workshop on Security and Human Behaviour
- ‘Security Engineering – A Guide to Building Dependable Distributed Systems’