“Hacking cash machines”
Extracting 3DES Keys from an IBM 4758 running CCA software
(or, “Why I was wearing a tie on the telly”)

Richard Clayton

Presented at: Jesus College, 4th February 2002
Summary

- Keys and Ciphers
- The IBM 4758 Cryptoprocessor
- How PIN values work
- Mike Bond’s “API attacks”
- The low-cost hardware “DES cracker”
- How to extract 3DES keys from a IBM 4758
- Some thoughts on “full disclosure”
Keys and ciphers

• Kerckhoff’s doctrine (1883)
  – the security of a system should depend upon its key and not upon its design remaining obscure

• If there is no shortcut then the security of a system depends upon its key length
  – trying all possibilities @ 33 million keys/sec
    • $2^{40} = 9$ hours
    • $2^{56} = 68.4$ years
    • $2^{80} = 5$ billion years
A History of Tamper Resistance

**Problem**: another program on the same machine can access your sensitive data

- Put keys into separate microprocessor
- Put microprocessor into a tin box
- Photocells and tilt detection
- Epoxy “potting”
- Tamper detecting barriers
The IBM 4758

• Protective barrier with wires of chemically similar compound
• Detectors for temperature & X-Rays
• “Tempest” shielding for RF emission
• Low pass filters on power supply rails
• Multi-stage “ratchet” boot sequence

= STATE OF THE ART PROTECTION!
CCA and PIN values

- Common Cryptographic Architecture
  - runs on many IBM platforms
  - available for free to run on a 4758
- A PIN value (in the CCA world) is the account number encrypted with (112 bit) 3DES key and last few bytes made decimal
- Changing a PIN => changing an offset
Key Entry under CCA

• Each key is loaded in two parts, which are then XORed together
  – XOR means that knowing one part tells you NOTHING about the final key value

• Two security officers, “trusted” not to collude, are given one part of the key each.
  – they authenticate themselves and then separately load these into the 4758.

• This makes the key entirely secure...
Michael Bond’s “API attacks”

• New type of attack: use standard API in non-standard way to cause dumb things
  – Overloaded key types
  – Unauthorised type casting
  – 3DES binding attack
  – Related keys

Mike’s PhD topic targets formal methods that will detect (and avoid) these problems
The Meet-in-the-Middle Attack

**Idea:** Attack multiple keys in parallel

- Encrypt the same plaintext under each of the multiple keys to get a “test vector”
- Attack by trying all keys in sequence but check for a match against any test vector value (check is faster than encrypt)
- Typical case: A $2^{56}$ search for one key becomes a $2^{42}$ search for $2^{14}$ keys
Attacking the CCA : Part 1

• Create unknown DES key part
• XOR in “...001”, “...002”, “...003” etc
• Encrypt zero value under each key
• Repeat to get 16384 ($2^{14}$) results
• Some complexity because of parity issues, but essentially simple & takes 10 minutes.
• Use “brute-force” attack to get the DES key
$995 DES Cracker

1, 2, 3, 4, 5, 6, 7...

Value 1, 2, 3, 4, 5, 6, 7...

zero

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
etc etc etc

= ?
Low-cost DES Cracker

• $995 Excalibur kit (Altera 20K200 FPGA)
  – chip cost is ~$5 (in volume; $178 one-off)
• 33MHz pipeline (& 60MHz possible)
• this is $\sim 2^{25}$ keys/second
  – 56 bit DES = 68 years
• However... it looks for 16384 keys in parallel
  – with average luck, find first key in 25.4 hours
Design Overview

- Test Vectors
- PC Client
- UART
- 16-Bit NIOS Microprocessor
- I/O
- External RAM
- Instruction Decoder
- DES Pipeline
A DES Pipeline Stage
DES Pipeline

L₀ R₀
L₀ R₁
L₂ R₂
L₃ R₃

Linear Feedback Shift Register
Extension
Fitting the Design Onto the Chip

Max of 8320 LUTs … and using all except 17

- LFSR saves pipelining key values
- Careful attention to instruction decoder
- Minimal settings for NIOS processor
- Redesigned S-Boxes
Can always achieve:
6 LUTs / bit
=> 24 LUTs/S-Box
Some S-Boxes Have Structure

• SBOX4 : address : 543210 : 4 bit result =

  7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
  13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
  10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
  3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14.

• Rearrange addressing in order 532104

  7, 14, 0, 9, 1, 8, 11, 4, 10, 9, 12, 7, 15, 3, 5, 8,
  13, 11, 6, 0, 4, 2, 1, 14, 3, 0, 10, 13, 9, 5, 12, 2,
  13, 3, 6, 10, 2, 5, 12, 15, 6, 0, 11, 13, 1, 14, 2, 4,
  8, 5, 15, 3, 7, 12, 10, 9, 15, 6, 1, 8, 4, 11, 7, 14.

and then feed it into the logic minimiser...


R0 = a[0] # a[1] & X4 # !a[1] & !X3;
R1 = a[0] # a[1] & !X1 # a[1] & X2;

data0 = (a[1] & !X1 # a[1] & X2 # a[0]) & R0;
data1 = (a[1] & !X4 # a[1] & X3 # a[0]) & R1;

ie: SBOX4 uses just 16 LUTs, not 24
savings also on:

    SBOX2: 23
    SBOX3: 23
    SBOX7: 23
    SBOX8: 22

    total = 13 LUTs
    (* 16 stages = 208)
Why Use Hardware Anyway?

Hardware DES implementation is >>25 times faster than the best software implementations.

• eg: Software [seeking any 1 of 64K keys]
  – 6 modern PCs running in parallel
  – £4500
  – 84 hours (3.5 days)

• & Hardware [seeking any 1 of 16K keys]
  – Altera evaluation board (no soldering required)
  – $995
  – 22.5 hours (for same example, NB: 1/4 parallelism)
Attacking the CCA : Part 2

• Recall we had 16K related DES keys
• We can crack one of these in ~1 day
• Now create 16K related 3DES keys with “replicate” halves and “exporter” capability
  – 3DES = EncryptA; DecryptB; EncryptA
• Export the DES key under the 3DES keys
• Since replicate can also crack in ~1 day
Attacking the CCA : Part 3

• Create non-replicate 3DES key by combining two unequal halves with the replicate halves that we’ve now determined
• Export all the CCA keys under this key
• Download list of PIN offsets
• Use magnetic stripe writer to create cards
• Use any ATM to extract money from accounts
• Go to Bermuda!
IBM’s Response

• Nov 2000 (Mike’s first results)
  – nothing (typecasting seen as legitimate)

• May 2001 (Mike’s CHES paper)
  – nothing

• Nov 2001 (Newsnight program)
  – attack “infeasible in realistic system implementations”
  – followed by advice to disable Combine Key Parts

• Real Soon Now
  – new version of CCA available [+ bug fix]
“Full Disclosure”

• Should you tell vendor & keep quiet?
  – vendor has limited incentive to act

• Should you publish & be damned?
  – “black hats” may be unaware of problem

• Should exploits be published?
  – “script kiddies” & sysadmins both need them

• Current consensus is to tell vendor and publish after pre-set delay. Recent decisions to suppress exploit info are controversial.
Make Your Own!

http://www.cl.cam.ac.uk/~rncl/descrack/