UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Benefits and Pitfalls of Cryptographic Hardware

Mike Bond
Computer Security Group

2002 Information Security Forum **28th January**

Overview

- Using cryptographic hardware to protect your business
- Disasters in retail banking crypto hardware
- Developing the threat model
- Getting procedural controls right
- Gaining assurance and penetration testing
- Peer review and Summary

What is a Cryptoprocessor ?

- A tamper-resistant processor which uses cryptography to control processing of and access to sensitive data

- Attached to a host computer e.g. web server, mainframe which communicates requests via the Security API
- Can run software provided by manufacturer or client

Who Needs Crypto Hardware ?

- Those with high crypto throughput requirements

  Example: SSL acceleration for webservers

- Those who need to enforce access policies to sensitive information

  Example: Granting signing permission at a Certification Authority

- Those who need to protect mission critical sensitive data

  Example: Protecting PIN generation keys at banks

Using cryptographic hardware to protect your business

- Define the security-relevant code, and load it into the cryptoprocessor to isolate it from the rest of the system
- Keep the amount of security-relevant code to a minimum, to make it easier to get assurance of correctness

Two Approaches

- Write the security relevant code in-house
- Configure existing software provided by a manufacturer or third-party to suit your needs

But who tests the design?

Disasters in Retail Banking
Crypto Hardware

- Cryptoprocessors used for securing communications between banks, from banks to ATMs, and for storing customer PINs and PIN generation keys
- Major API designed by VISA; several manufacturers provide implementations e.g. Racal/Zaxus/Thales

✘ API specifications only available to banks and original designers

The Visa Security Module

Null Key Attack

- Top-level crypto keys exchanged between banks in several parts carried by separate couriers, which are recombined using the exclusive-OR function
- 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

Offset Calculation Attack

- 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

Type System Attack

- Encrypting communication keys for transfer to an ATMs used exactly the same process as calculating a customer PIN
- Customer PINs could be generated by re-labelling an account number as a communications key, and using the same encryption process

The IBM 4758 CCA

Meet-in-the-Middle Attack

- Brute force attack (guessing) to find a single DES key is extremely difficult
- But if there are many targets of equal value, the effort to discover one of the keys is much less

- Affects cryptoprocessors from at least six different manufacturers (every module examined so far)

CCA Failure Mode

- Complex systems fail in complex ways!

- Triple DES key binding design error reduces effort to crack to twice as hard as single DES
- Meet-in-the-middle attack cracks DES within 24 hours
- Poor design of procedural controls mean a single user could have all the relevant permissions

- In depth feasibility study of this attack at University of Cambridge received international publicity in Nov ‘01

Lessons Learned in Retail Banking

- Cryptoprocessors are only as secure as the software they run, or as the people who configure them
- Both standardised and in-house developed APIs are susceptible
- Even the massive in-house resources of a company such as IBM has not protected against serious faults

Developing the Threat Model

- How can the end user develop their crypto hardware application to use third-party products effectively, and be robust against attacks?

- Develop your threat model (understand your attackers)
- Understand the manufacturer's perception of your threat model (**not** the same as the features provided)
- Choose the product where the threat models match best

Your Threat Model

- What information/access is valuable?
- Main threat from insiders or outsiders?
- How much physical access would the attacker have?
- How much privilege might the attacker already have?
- How long would it take to discover a security breach?

Manufacturer's Threat Model

- How much tamper-resistance is provided?
- What actions can be put under dual control?
- Reliance on audit to spot attacks?
- What authentication tokens are available, and how are they normally mapped to personnel?
- Are those who initialise the module trusted?
- What information must travel via a trusted communications path?

Getting Procedural Controls Right

- Many failures occur when the end user makes false assumptions about the guarantees an API feature provides

- Example: IBM CCA key entry procedure provides dual control on the *confidentiality* of a key, but not on its *integrity*. Attacks involving integrity compromise must be protected against some other way

Gaining Assurance

- How can the manufacturer develop their crypto hardware to function correctly, and encourage safe usage?

- Publish the API (**not** standardise)
- Test API against specific threat models
- Detail not just intended usage, but all assumptions required for secure operation

Penetration Testing

- The ultimate test of security with a specific threat model
- But threat model is *too specific*. Will change as software updated, personnel move, and procedures modified.

- Only reveals a specific instance of a possible generic fault.
- Manufacturers faults get patched by end user.

Peer Review

- Lots of brainpower available in the open community for free. Only requirement is mutual benefit.
- The good guys/bad guys arms race is inevitable. Keeping APIs in-house is running the race blind.
- Crypto hardware is expensive and attacks generally require some degree of physical access. In this field, there is no such thing as a ‘script kiddie’.

Summary

- Physical attack is a serious threat, and crypto hardware can provide resistance to it
- Crypto hardware is susceptible to software flaws just like normal operating systems and PCs
- Crypto hardware is specially designed to enforce access control policies which resist attack by *individual* corrupt insiders unlike normal operating systems
- As much care must be taken understanding and configuring third party software for cryptoprocessors as in writing your own in-house
- The open community is a valuable tool, and can be used without adopting a ‘full disclosure’ mentality.

More Information

My Research Homepage

http://www.cl.cam.ac.uk/~mkb23/research.html

Attack on the IBM 4758 CCA

http://www.cl.cam.ac.uk/~rnc1/descrack