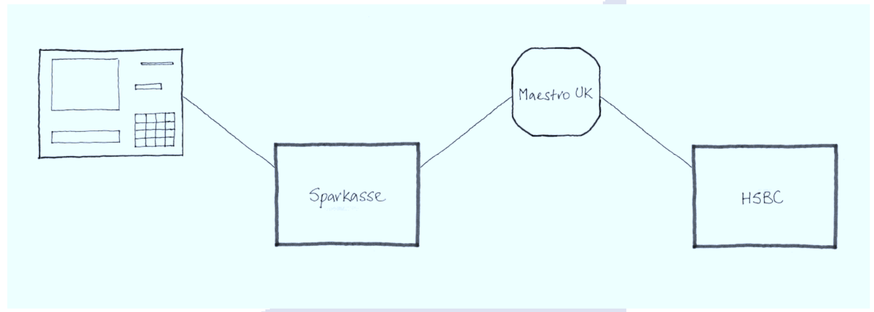
How Safe is Your PIN?

Graham Steel

http://dream.inf.ed.ac.uk/projects/aascs

The Problem

Suppose you bank in the UK. You travel to Germany, and put your bank card into a German ATM to withdraw some money. You type in your PIN, which has to be sent to the UK for your bank to authorise. The PIN is encrypted to prevent eavesdroppers from obtaining it. However, the ATM you use will not know the correct key for sending the PIN directly to your UK bank. Instead, the encrypted PIN will pass through several nodes in the ATM network before it gets to the issuing bank. At each node, the PIN will be decrypted and re-encrypted under the appropriate key to be sent to the next node.



ATM Network

It is considered an unacceptable security risk for the PIN to appear in 'plaintext' form in the memory of a normal PC in a bank. So, all PIN decryption, re-encryption and verification is carried out inside tamper-proof hardware security modules (HSMs) like the one shown below. Note the tamper-proof, shielded enclosure, inside which all the processing occurs. Inside the enclosure is a CPU, a dedicated cryptoprocessor for standard encryption algorithms, and a small amount of memory.

IBM 4758 HSM

These HSMs have strictly controlled APIs, designed to prevent corrupt insiders in a bank from being able to discover customers' PIN values. However, recent research has uncovered several flaws in such APIs which could allow PINs to be obtained. The aim of our project is to develop techniques that help API designers to specify their systems precisely and check them for flaws.

Key Management Attacks

Your original PIN is a cryptographic function of your personal account number (PAN). Typically, this means encrypting your PAN under a secret *PIN Derivation Key* (PDK), and returning the first 4 digits of the decimalised result.

$$\text{PIN} = \{\!|\ \text{PAN}\ |\!\}_{PDK}$$

One of the HSM's jobs is to store PDKs securely. This is usually done by encrypting them under the HSM's master key, which is stored in the HSM's tamper proof memory. The encrypted PDKs can then be stored on the bank computer's hard drive, and can only be used by sending it back into the HSM under the terms of the API.

Other keys must also be securely stored, for example data keys, used to encrypt communications in the ATM network. The API must allows data keys to be used for encryption, but must also prevent PDKs from being used this way, since this would allow an intruder to freely obtain PINs by encrypting PANs!. In 2001, Mike Bond discovered an flaw in the IBM 4758 CCA API that allowed an intruder to convert a PDK into a data key on import, permitting such an attack. In our project we have shown how to rediscover the attack using a formalisation in first-order logic [1]. We have analysed IBM's recommendations for fixing the flaw using a model checker, resulting in the discovery of another (new) attack, [2]. We have proved the CCA key-management problem to be decidable in this case [3], and using an automated decision procedure, we have proved certain versions of the key-management scheme to be secure, [4].

PIN Cracking Attacks

In order for a PIN to be encrypted by the standard encryption algorithms such as 3DES, it must be made into a 64-bit *PIN block*. To complicate matters somewhat, different banks expect the PIN to be formatted into a PIN block in different ways. For example, the ISO proposed the *ISO-0 format* in ISO standard 9564 ($\oplus$ represents bitwise XOR):

$$\begin{aligned} B_1 &= 0\ 4\ P_1\ P_2\ P_3\ P_4\ F\ F\ F\ F\ F\ F\ F\ F\ F\ F \\ B_2 &= 0\ 0\ 0\ 0\ A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12} \\ \text{PIN Block} &= B_1 \oplus B_2 \end{aligned}$$

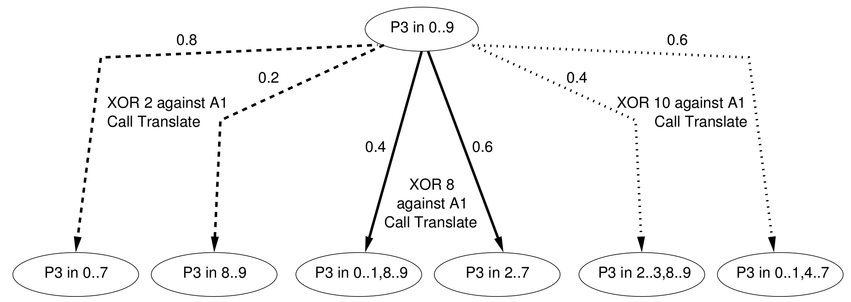
The 0 at the beginning of B_1 marks the block as being in format 0. The 4 indicates the length of the PIN, in this case 4 digits. The A_is in B2 are the 12 digits of the customer's personal account number (PAN). This way, two customers with the same PIN will not have the same encrypted block.

A Reformatting Attack

The API of the HSM may include a 'translate' command to allow blocks to reformatted ready to be sent to the next node. This introduces an attack, in which the attacker calls the translate function with a different value for the PAN. This modification could be to XOR in the value 8 against the first digit, producing $A'_1 = A_1 \oplus 8$. Now, if the PIN digit is in the range 2–7, the HSM will signal an error, since these values all XOR against 8 to give a value between A and F hexadecimal. However, if the PIN digit P_3 is 0, 1, 8 or 9, the error check will still pass, since these values XOR 8 all give a decimal value. The attack can be extended to discover the whole PIN in an average of 13.6 operations!

AnaBlock

There are several known families of PIN recovery attacks like the reformatting attack. Each customer for an HSM manufacturer will configure their HSMs differently, enabling different PIN block formats and different commands from the API. We have created a system, AnaBlock, which takes such a configuration and determines the most effective PIN block attack available to the intruder, [5].



A fragment of a tree produced by AnaBlock

About the Project

This work is part of the ongoing project 'Automated Analysis of Security Critical Systems', funded by EPSRC (grant number GR/S98139/01), with nCipher plc., a manufacturer of cryptographic hardware security modules, and CESG, the information assurance arm of GCHQ. For more details, see:

http://dream.dai.ed.ac.uk/projects/aascs

References

[1] G. Steel. Deduction with XOR Constraints in Security API Modelling. In Proceedings of the 20th Conference on Automated Deduction (CADE 20), July 2005, pages 322-336.

[2] G. Keighren, G. Steel, A. Bundy. Checking Security APIs with Protocol Analysis Tools. Submitted to Information and Computation.

[3] V. Cortier and G. Steel. On the Decidability of a Class of XOR-based Key-management APIs Presented at the FCS-ARSPA workshop at FLoC'06.

[4] V. Cortier, G. Keighren, G. Steel. Automatic Analysis of the Security of XOR-based Key Management Schemes. Submitted to TACAS 2007.

[5] G. Steel. Formal Analysis of PIN Block Attacks. Theoretical Computer Science, 367(1-2) p. 257-270, 2006.