UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Extending Security Protocol Analysis : New Challenges

Mike Bond, Jolyon Clulow
{Mike.Bond, Jolyon.Clulow}@cl.cam.ac.uk

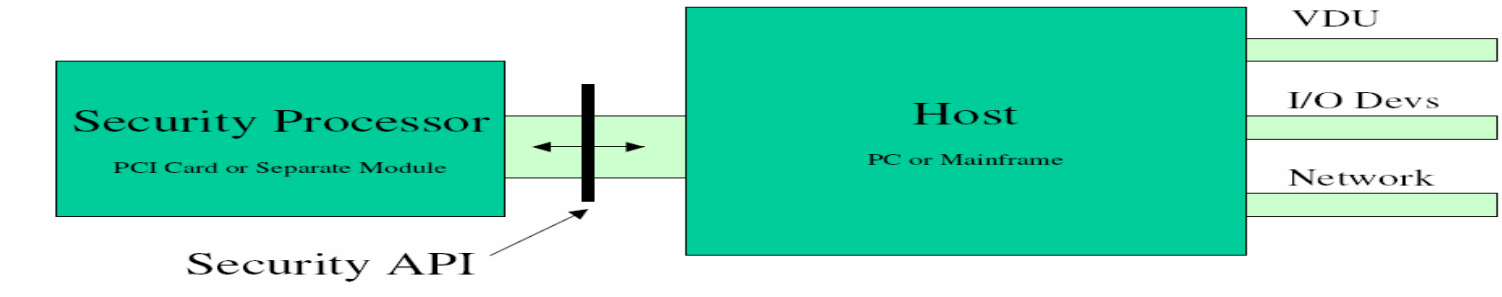
Workshop on
Automated Reasoning for Security Protocols Analysis (ARSPA 2004)
4th July 2004

Outline

- An introduction to security APIs
- Similarities between protocols and security APIs
- Why security APIs are of interest
- Perfect encryption
- Information leakage
- Protecting low entropy data
- Conclusion

What are Security APIs

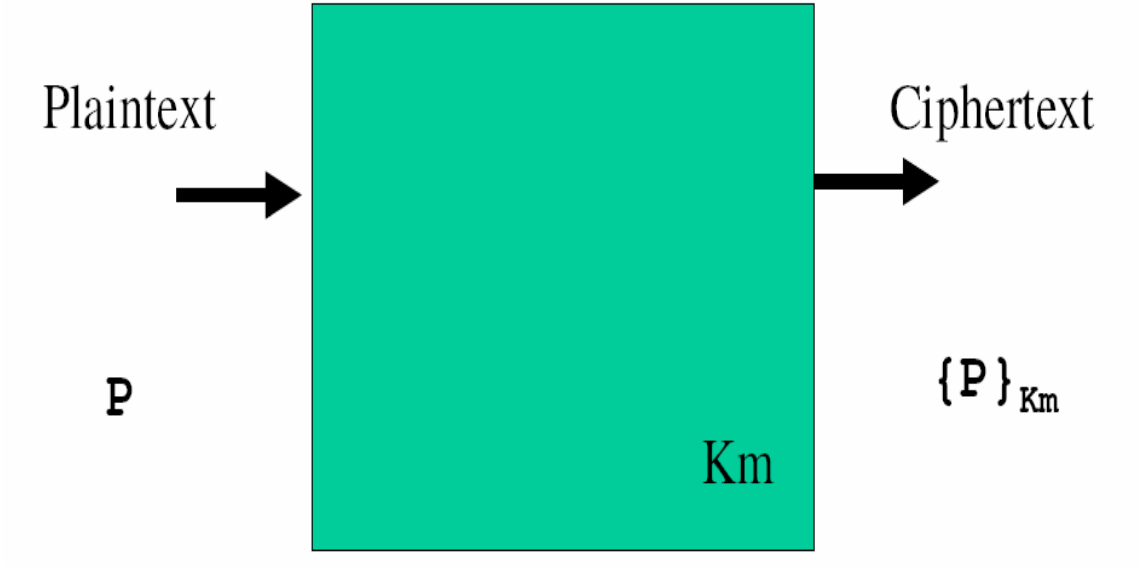
- An API that allows users to work with sensitive data and keys, provides cryptographic operations, and uses cryptographic techniques to enforce a policy on the usage of data.



Some Examples

- Cryptographic tokens (e.g. smart cards)
- Cryptographic accelerators
- Tamper protected devices (e.g. IBM 4758)
- Cryptographic Service Providers (e.g. MS CAPI)
- Standards (e.g. PKCS #11)

The Simplest API Call



$$U \rightarrow S : P$$

$$S \rightarrow U : \{P\}_{Km}$$

A Typical API Call

Data_Key_Import (CSNBDKM)

Platform/ Product	OS/2	AIX	NT	OS/400
IBM 4758-1	X	X	X	X

The Data_Key_Import verb imports an encrypted, source DES DATA key and creates or updates a target internal key token with the master-key enciphered source key. The verb can import the key into an internal key token in application storage or in key storage. This verb, which is authorized with a different control point than used with the Key_Export verb, allows you to limit the export operations to DATA keys as compared to the capabilities of the more general verb.

Specify the following:

- An external key token containing the source key to be imported. The external key token must indicate that a control vector is present; however, the control vector is usually valued at zero.

  Alternatively, you can provide the encrypted data key at offset 16 in an otherwise all X'00' key token. The verb will process this token format as a DATA key encrypted by the importer key and a null (all zero) control vector.
- An IMPORTER key-encrypting key under which the source key is deciphered.
- An internal or null key token. The internal key token can be located in application data storage or in key storage.

The verb builds the internal key token by the following:

- Creates a default control vector for a DATA key type in the internal key token, if the control vector in the external key token is zero. If the control vector is not zero, the verb copies the control vector into the internal key token from the external key token.
- Multiply-deciphers the key under the keys formed by the exclusive-OR of the key-encrypting key (identified in the *importer_key_identifier*) and the control vector in the external key token, then multiply-enciphers the key under keys formed by the exclusive-OR of the master key and the control vector in the internal key token. The verb places the key in the internal key token.
- Calculates a token-validation value and stores it in the internal key token.

This verb does not adjust the key parity of the source key.

Restrictions

None

Format

CSNBDKM			
return_code	Output	Integer	
reason_code	Output	Integer	
exit_data_length	In/Output	Integer	
exit_data	In/Output	String	*exit_data_length* bytes
source_key_token	Input	String	64 bytes
importer_key_identifier	Input	String	64 bytes
target_key_identifier	In/Output	String	64 bytes

Parameters

For the definitions of the *return_code*, *reason_code*, *exit_data_length*, and *exit_data* parameters, see "Parameters Common to All Verbs" on page 1-10.

source_key_token
The *source_key_token* parameter is a pointer to a 64-byte string variable containing the source key to be imported. The source key must be an external key.

importer_key_identifier
The *importer_key_identifier* parameter is a pointer to a 64-byte string variable containing the (IMPORTER) transport key used to decipher the source key.

target_key_identifier
The *target_key_identifier* parameter is a pointer to a 64-byte string variable containing a null key token, an internal key token, or the key label of an internal key token or null key token record in key storage. The key token receives the imported key.

Required Commands

The Data_Key_Import verb requires the Data Key Import command (offset X'0109') to be enabled in the hardware.

API Complexity

2 HOST COMMANDS

2.1 LIST OF HOST COMMANDS (ALPHABETICAL)

Similarities between Security APIs and Protocols

- Security APIs closely resemble protocols
- A cryptographic processor (imagine a PC in a safe) that is networked attached and is used as a service by one or more users, is conceptually similar to a trusted third party.
- A given protocol can be realised (or instantiated) by a security API.
- A given security API can be described by a set of protocols.
- A security API typically has finer granularity than a protocol since a single protocol message/operation may require multiple API calls.

Why Apply Formal Methods to Security APIs

- Similarity between security APIs and protocols
- Daunting size and complexity of security APIs make them difficult to analyse by hand
- Need for assurance of security for commercial security products
  - Many commercial products rely on a `trust us' attitude
- Custom extensibility of security APIs

Why are Security APIs of Interest?

- Rich source of vulnerabilities
- Little application of formal methods to the problems of security API research
- Verification and certification is a significant, real world problem with commercial implications for industry

(New) Challenges

- Our process
  - Reviewed the literature of attacks on security APIs
  - For each attack we asked the question “Can we detect this attack through the application of existing techniques?”
  - Describe the basic idea behind the attack by means of a simple example, preferably using protocol notation
- We present the results as a set of open problems and a wish list of functionality for future automated reasoning tools.

Perfect Encryption

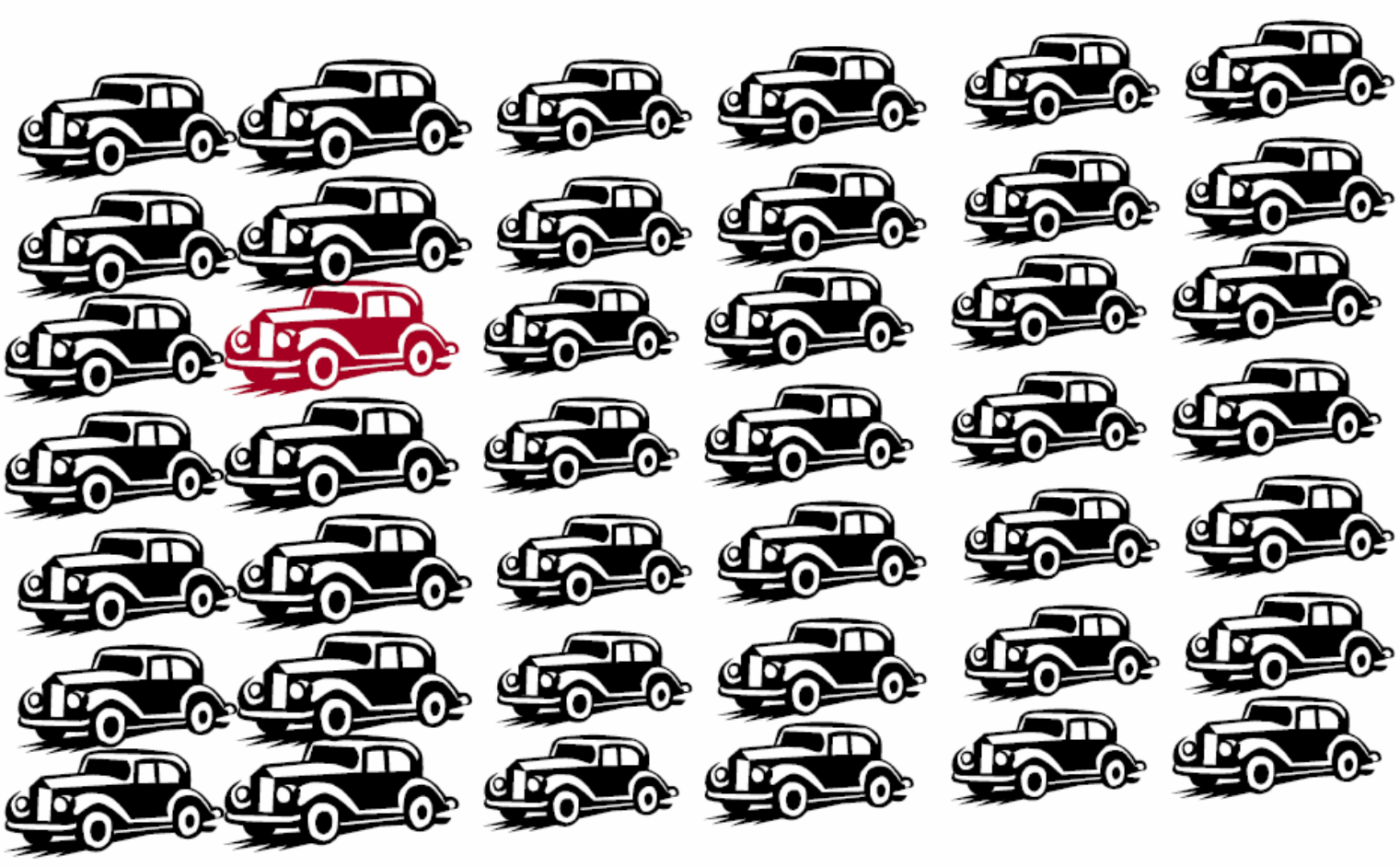
- Is $\{X\}_K$ secure?
  - Not necessarily a valid assumption for low cost, low power and embedded system (e.g. lightweight ciphers in car key-fobs where every bit transmitted is expensive in power consumption)
  - Exporting keys under weaker keys/algorithms (e.g. PKCS #11)
  - Key binding issues

Parallel Key Search

- A thief walks into a car park.

- How many keys must he try?

Parallel Key Search (2)

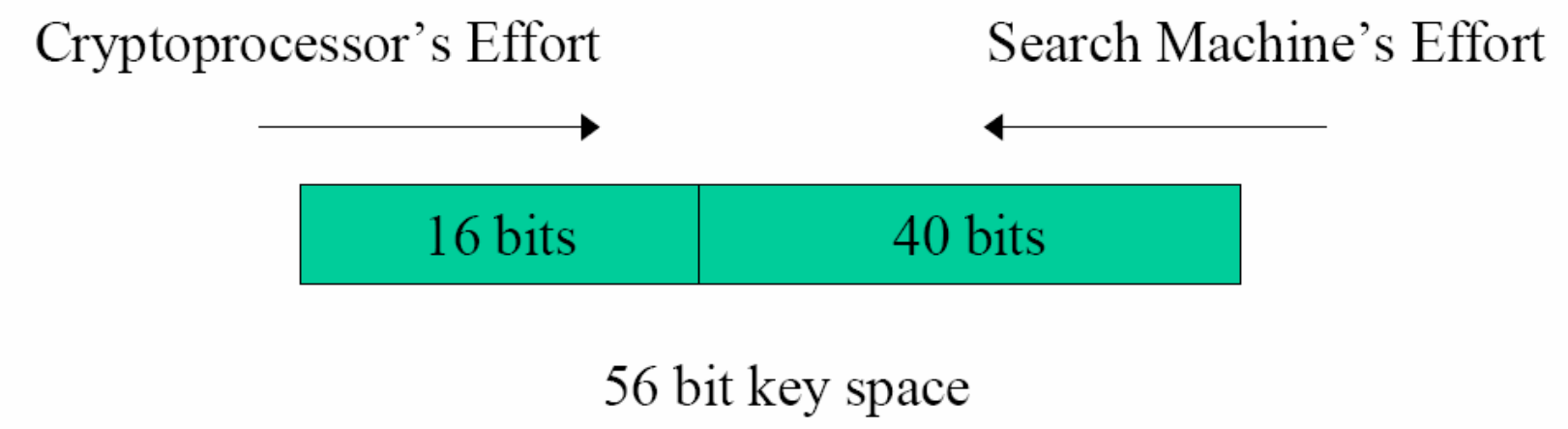


Parallel Key Search

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

  `U -> C: X,` $\{KEY\_i\}_{KM}$

  `C -> U:` $\{X\}_{KEY\_i}$
- Do 2^{40} search



Parallel Key Search using Key Offsets

$$A \rightarrow S : X, \{K\}_{KM}, i$$

$$S \rightarrow A : \{X\}_{K \oplus i}$$

Other Examples?

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

$$S \rightarrow A : \{R_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$

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

$$B \rightarrow A : \{R_B\}_{K_{AB}}$$

$$A \rightarrow B : \{R_B - 1\}_{K_{AB}}$$

Parallel Key Search using the Needham-Schroeder Protocol

$$E \rightarrow S : i, B, X$$

$$S \rightarrow A : \{X, B, K_{iB}, \{K_{iB}, A\}_{K_{BS}}\}_{K_{iS}}$$

- Generate *i* encryptions of *X* under different keys.

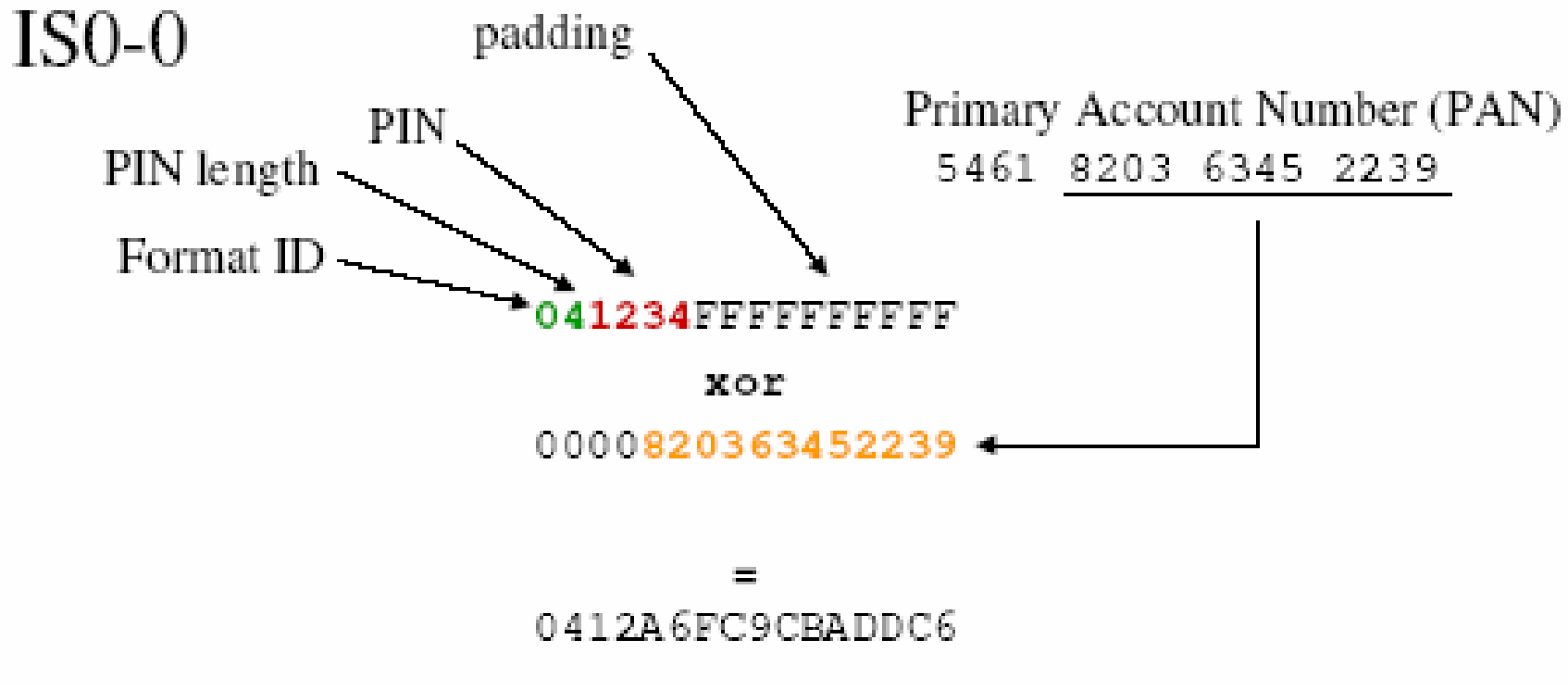
Wish List for Perfect Encryption

- Reason efficiently about 3DES keys.
- Formal tools capable of analysing protocols/APIs identifying when it is possible to obtain the necessary data required for such attacks.
- Or the ability to calculate a numerical bound that limits the parameters of the system thereby ensuring security.

Information Leakage

- Similar to Side Channel attacks against physical devices or implementations (e.g. timing attacks, power analysis, etc).
- Protocols themselves may leak a small amount of information per protocol run
- Ultimately may lead to the recovery of a secret or bring a secret within range of a brute force attack
- Non trivial algorithm may be required to convert the information revealed into knowledge of the secret

PIN Block Formats



PIN Integrity Check Protocol

$$A \rightarrow S : X, \{P \oplus A\}_K$$

$$S \rightarrow A : true \text{ iff } (P \oplus A) \oplus X < 10$$

Identifying the PIN

X	P+A					
		0,1	2,3	4,5	6,7	8,9
	0,1	Pass	Pass	Pass	Pass	Pass
	2,3	Pass	Pass	Pass	Pass	FAIL
	4,5	Pass	Pass	Pass	Pass	FAIL
	6,7	Pass	Pass	Pass	Pass	FAIL
	8,9	Pass	FAIL	FAIL	FAIL	Pass
	A,B	FAIL	Pass	FAIL	FAIL	Pass
	C,D	FAIL	FAIL	Pass	FAIL	Pass
	E,F	FAIL	FAIL	FAIL	Pass	Pass

Wish List for Information Leakage

- Identifying potential leakages of information and understanding how this information might be used
- Constructing an algorithm that assimilates the leaked information and reconstructs the underlying secret - an unrealistic goal?
- Identifying the rate at which information is lost and establishing a bound on security.

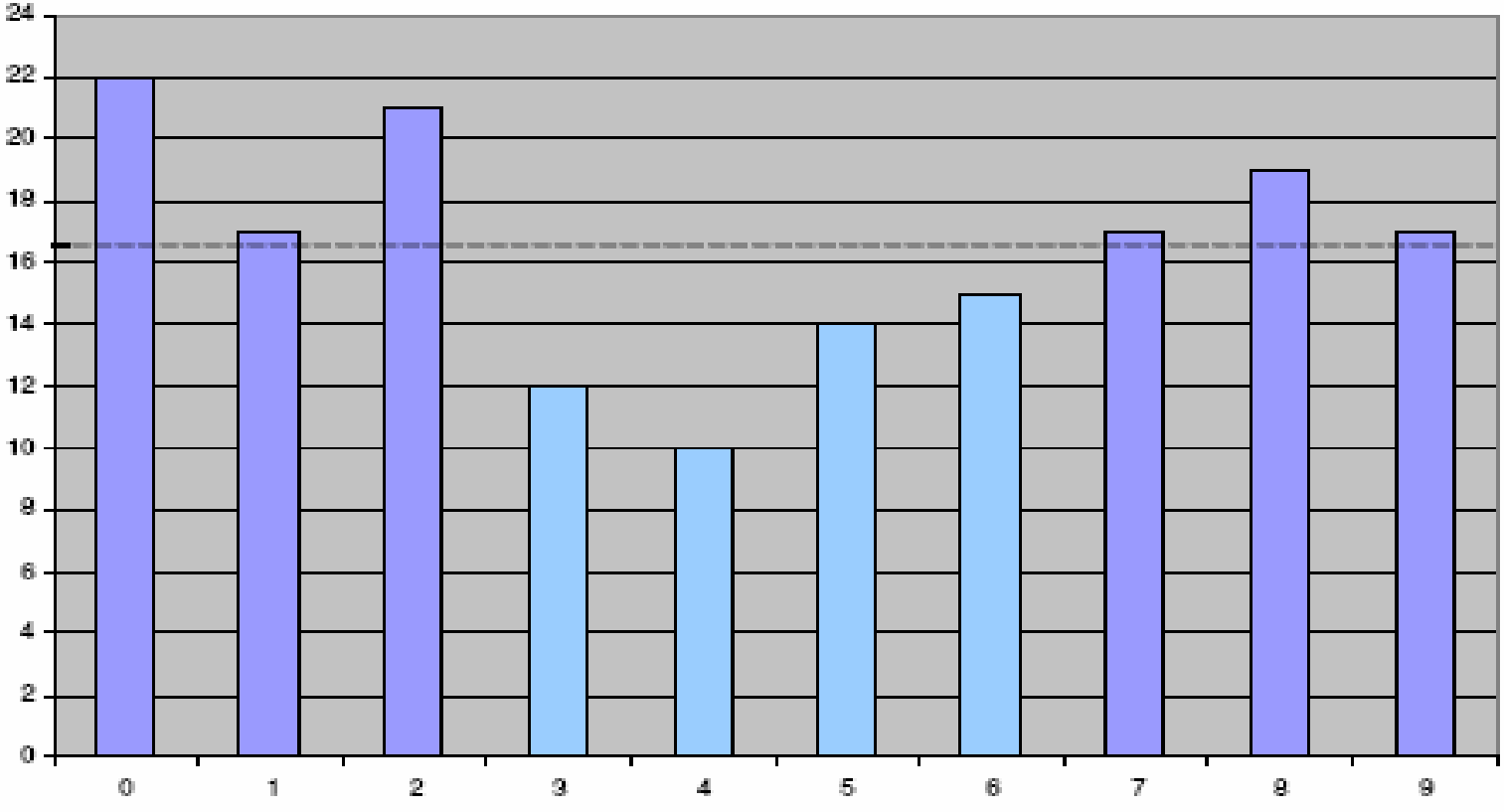
Protecting Low-Entropy Data

- Weak secrets and guessable passwords
  - Authenticating principals with weak passwords
  - Boot strapping strong session keys from weak secrets
  - Interrogating encrypted, randomised databases (e.g. medical databases)
- Lowe describes work using FDR to guess weak secrets used as keys in offline attacks
- What about online attacks against weak secrets as data? What about manipulations of or operations on weak data?

Statistical Distribution Attacks against PINs

- Personal Identification Numbers (PINs) are weak secrets
- Encrypted as data $\{PIN\}_{KEY}$
- Generated with a non-uniform, measurable distribution

Example Distribution: HSBC



Sample size: 45 people (just large enough to prove non-uniform hypothesis with 1% conf)

Statistical Attacks (2)

- Some manipulation is possible ({PIN + PAN}$_{KEY}$ where PAN is the supplied account number)
- How does the distribution change?
- What does this tell you about the possible PIN values?

Wish List for Low-Entropy Data

- Generic framework for reasoning about information flow through security protocols
- Cope with leakage that may be both necessary and acceptable
- Provide assurance that the total rate of leakage cannot exceed some limit.

Conclusions

- Some attacks, ideas and issues …
- Research into automated reasoning can benefit from looking at security APIs.

More Info

Home page

- www.cl.cam.ac.uk/users/mkb23/
- www.cl.cam.ac.uk/users/jc407/

Some initial results using automated tools to attack financial systems

- “Using a Theorem Prover to Rob a Bank” …coming soon.

Come talk to us.