Miscellaneous Topics

Buy a rifle, encrypt your data, and wait for the revolution

Smart Cards

Invented in the early 1970's

Technology became viable in early 1980's

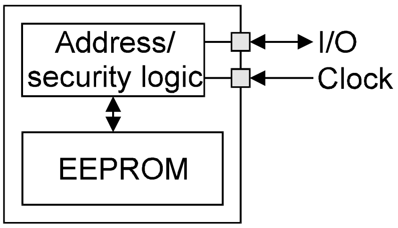
Major use is prepaid telephone cards (hundreds of millions)

- Use a one-way (down) counter to store card balance

Other uses

- Student ID/library cards
- Patient data
- Micropayments (bus fares, photocopying, snack food)

Memory Cards



Usually based on I^2C (serial memory) bus

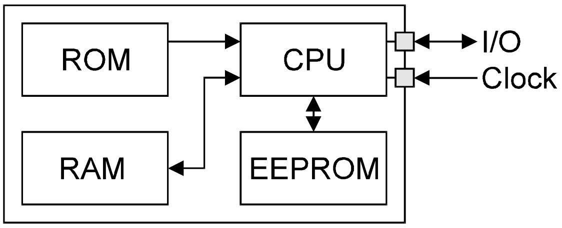
Typical capacity: 256 bytes

EEPROM capabilities

- Nonvolatile storage
- 10,000 write/erase cycles
- 10ms to write a cell or group of cells

Cost: $5

Microprocessor Cards



ROM/RAM contains card operating system and working storage

EEPROM used for data storage

Microprocessor Cards (ctd)

Typical specifications

- 8-bit CPU
- 16K ROM
- 256 bytes RAM
- 4K EEPROM

Size ratio of memory cells:

RAM = 4× EEPROM size
= 16× ROM size

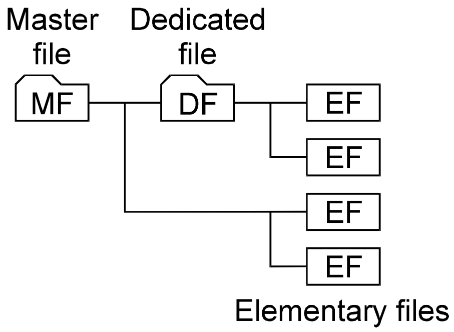
Cost: $5-50 (with crypto accelerator)

Smart Card Technology

Based on ISO 7816 standard, which defines

- Card size, contact layout, electrical characteristics
- I/O protocols
  - Byte-based
  - Block-based
- File structures

File Structures



Files addressed by 16-bit file ID (FID)

- FID is often broken into DF:EF parts (MF is always 0x3F00)

Files are generally fixed-length and fixed-format

File Types

Transparent

- Binary blob

Linear fixed

- $n \times$ fixed-length records

Linear variable

- n records of fixed (but different) lengths

Cyclic

- Linear fixed, oldest record gets overwritten

Execute

- Special case of transparent file

File Attributes

EEPROM has special requirements (slow write, limited number of write cycles) which are supported by card attributes

- WORM, only written once
- Multiple write, uses redundant cells to recover when some cells die
- Error detection/correction capabilities for high-value data
- Error recovery, ensures atomic file writes
  - Power can be removed at any point
  - Requires complex buffering and state handling

Card Commands

Typical commands are

- CREATE/SELECT/DELETE FILE
- READ/WRITE/UPDATE BINARY
  - Write can only change bits from 1 to 0, update is a genuine write
- ERASE BINARY
- READ/WRITE/UPDATE RECORD
- APPEND RECORD
- INCREASE/DECREASE
  - Changes cyclic file position

Card Commands (ctd)

Access control

- Based on PIN of chip holder verification (CHV)
- VERIFY CHV
- CHANGE CHV
- UNBLOCK CHV
- ENABLE/DISABLE CHV

Authentication

- Simple challenge/response authentication protocol
- INTERNAL AUTHENTICATE
  - Authenticate card to terminal
- EXTERNAL AUTHENTICATE
  - Authenticate terminal to card

Card Commands (ctd)

Encryption: Various functions, typically

- ENCRYPT/DECRYPT
- SIGN DATA/VERIFY SIGNATURE

Electronic purse instructions

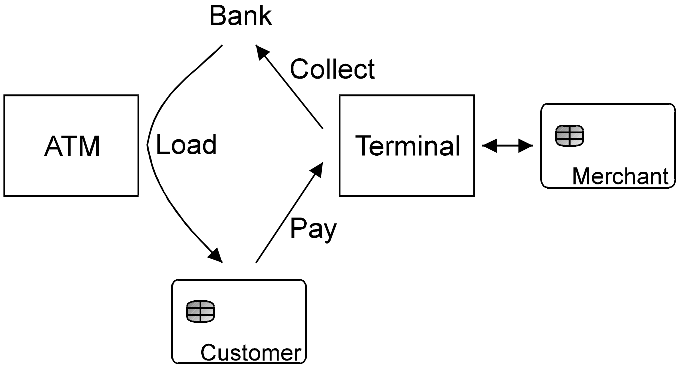
- INITIALISE/CREDIT/DEBIT

Application-specific instructions

- RUN GSM ALGORITHM

prEN 1546

Inter-sector electronic purse (IEP) standard, 1995



Both customer and merchant use smart-card based electronic purses to handle payment transactions

prEN 1546 (ctd)

Defines the overall framework in some detail, but leaves algorithms, payment types and parameters, and other details to implementors

- Specifies the file layout and data elements for the IEP
- Defines commands INITIALISE IEP, CREDIT IEP, DEBIT IEP, CONVERT IEP CURRENCY, and UPDATE IEP PARAMETER
- Specifies exact payment routines in a BASIC-like pseudolanguage
- All messages are "signed" (typically with a 4-byte DES MAC)
- Handles everything but purse-to-purse transactions

Includes many variants including a cut-down version for phonecards and extra acknowledgements for transactions

Credit IEP Transaction

IEP		Bank
INITIALISE bank for load (amount and currency)	→	Verify currency and balance
Verify details	←	INITIALISE IEP for load
Sign(DEBIT account)	→	Verify details Debit account
Verify details Update card state	←	Sign(CREDIT IEP)
Sign(Load acknowledgement)	→	Verify acknowledgement

Credit Merchant Transaction

IEP		Merchant
	←	INITIALISE IEP for purchase
Sign(INITIALISE merchant for purchase)	→	Verify details
Verify details Update card state	←	Sign(DEBIT IEP)
Sign(CREDIT Merchant)	→	Verify details Record transaction for transmission to bank
	←	Sign(Purchase acknowledgement)

Working with Cards

ISO 7816 provides only a standardised command set, implementation details are left to vendors

- Everyone does it differently

Standardised API's are slow to appear

PKCS #11 (crypto token interface) is the most common API

- Functionality is constantly changing to handle different card/vendor features
- Vendors typically only implement the portions which correspond to their products
- For any nontrivial application, custom handling is required for each card type

Working with Cards (ctd)

JavaCard

- Standard smart card with an interpreter for a Java-like language in ROM
- Card runs Java with most features (multiple data types, memory management, most class libraries, and all security (via the bytecode verifier)) stripped out
  - Can run up to 200 times slower than card native code

Provides the ability to mention both "Java" and "smart cards" in the same sales literature

Attacks on Smart Cards

Use doctored terminal/card reader

- Reuse and/or replay authentication to card
- Display \$*x* transaction but debit \$*y*
- Debit account multiple times

Protocol attacks

- Card security protocols are often simple and not terribly secure

Fool CPU into reading from external instead of internal ROM

Manipulating supply voltages can affect security mechanisms

- Picbuster
- Clock/power glitches can affect execution of instructions

Attacks on Smart Cards (ctd)

Erasing an EEPROM cell requires a high voltage (12 vs 5V) charge

- Don't provide the power to erase cells
- Most cards now generate the voltage internally
  - Destroy the (usually large) on-chip voltage generator to ensure the memory is never erased

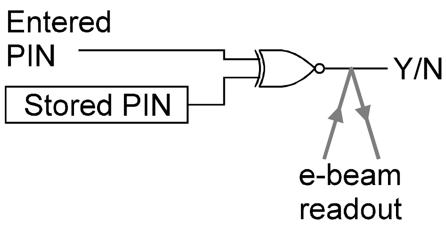
Physical Attacks

Erase onboard EPROM with UV spot beam

Remove chip from encapsulation with nitric acid

- Use microprobing to access internal circuit sections
- Use electron-beam tester to read signals from the operational circuit

Example: PIN recovery with an e-beam tester



Physical Attacks (ctd)

Modify the circuit using a focused ion beam (FIB) workstation

- Disable/bypass security circuitry (Mondex)
- Disconnect all but EEPROM and CPU read circuitry

Attacking the Random Number Generator

Generating good random data (for encryption keys) on a card is exceedingly difficult

- Self-contained, sealed environment contains very little unpredictable state

Possible attacks

- Cycle the RNG until the EEPROM locks up
- Drop the operating voltage to upset analogue-circuit RNG's
- French government attack: Force manufacturers to disable key generation
  - This was probably a blessing in disguise, since externally generated keys may be much safer to use

Timing/Power Analysis

Crypto operations in cards

- Take variable amounts of time depending on key and data bits
- Use variable amounts of power depending on key and data bits
  - Transistors are voltage-controlled switches which consume power and produce electromagnetic radiation
  - Power analysis can provide a picture of DES or RSA en/decrypt operations
  - Recovers 512-bit RSA key at ~3 bits/min on a PPro 200

Differential power analysis is even more powerful

- Many card challenge/response protocols are DES-based → apply many challenge/response operations and observe power signature

Voice Encryption

Built from three components

Hardware-based

- DSP with GSM or CELP speech compression
- DSP modem

Software-based

- GSM or CELP in software
- External modem or TCP/IP network connection

Mostly built from off-the-shelf parts (GSM DSP, modem DSP, software building blocks)

Typical Voice Encryption System

Speech compression

- GSM compression (high-bandwidth)
- CELP compression (low-bandwidth)

Security

- DH key exchange
- DES (larger manufacturers)
- 3DES, IDEA, Blowfish (smaller manufacturers, software)
- Password/PIN authentication

Typical Voice Encryption System (ctd)

Communications

- Built-in modem (hardware)
- Internet communications (software)

Speak Freely, `http://www.fourmilab.ch/netfone/windows/speak_freely.html`

- Typical software implementation
- Uses standard software components
- Portable across several operating systems

Problems

Latency issues (dropped packets)

Authentication/MITM attacks

No standardisation

GSM

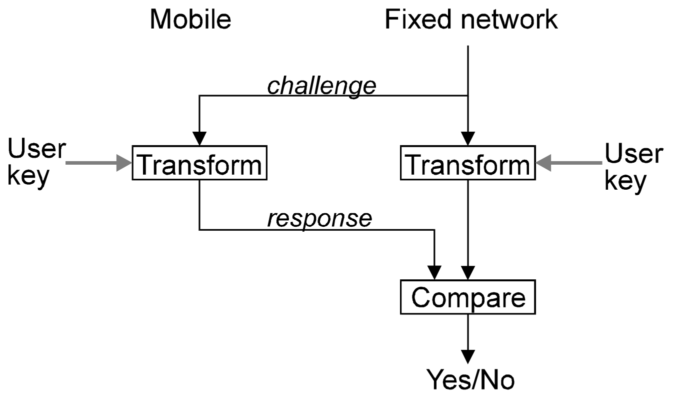
GSM subscriber identity module (SIM) contains

- International Mobile Subscriber Identity (IMSI)
- Subscriber identification key K_i

Used for authentication and encryption via simple challenge/response protocol

- A3 and A8 algorithms provide authentication (usually combined as COMP128)
- A5 provides encryption

GSM (ctd)

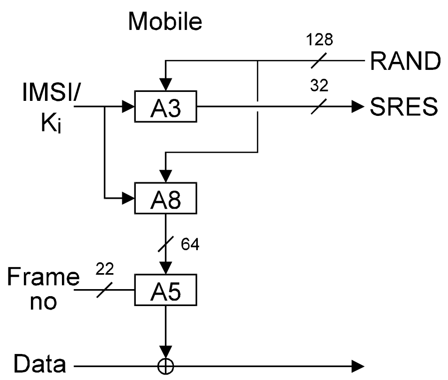


Authentication is simple challenge/response using A3 and IMSI/K_i

GSM Security

A3 used to generate response

A8 used to generate A5 key



GSM Security (ctd)

1. Base station transmits 128-bit challenge RAND
2. Mobile unit returns 32-bit signed response SRES via A3
3. RAND and K_i are combined via A8 to give a 64-bit A5 key
4. 114-bit frames are encrypted using the key and frame number as input to A5

GSM Security (ctd)

GSM security was broken in April 1998

- COMP128 is weak, allows IMSI and K_i to be extracted
  - Direct access to SIM (cellphone cloning)
  - Over-the-air queries to phone
- A5 was deliberately weakened by zeroing 10 key bits
- Claimed GSM fraud detection system doesn't seem to exist
- Affects 80 million GSM phones

GSM Security (ctd)

Key weakening was confirmed by logs from GSM base stations

```
BSSMAP GSM 08.08 Rev 3.9.2  (BSSM) HaNDover REQuest (HOREQ)
-------0 Discrimination bit D BSSMAP
0000000- Filler
00101011 Message Length    43
00010000 Message Type      0x10
Channel Type
00001011 IE Name           Channel type
00000011 IE Length         3
00000001 Speech/Data Indicator    Speech
00001000 Channel Rate/Type Full rate TCH channel Bm
00000001 Speech encoding algorithm      GSM speech algorithm
Encryption Information
00001010 IE Name           Encryption information
00001001 IE Length         9
00000010 Algorithm ID      GSM user data encryption V.1
******** Encryption  Key   C9 7F 45 7E 29 8E 08 00
Classmark Information Type 2
```

GSM Security (ctd)

Many countries were sold a weakened A5 called A5/2

- Workfactor to break A5 is ~2^{40}
- Workfactor to break A5/2 is ~2^{16}
- Much easier attack is to bypass GSM entirely and attack the base station or land lines/microwave links

Most other cellphone security systems have been broken too

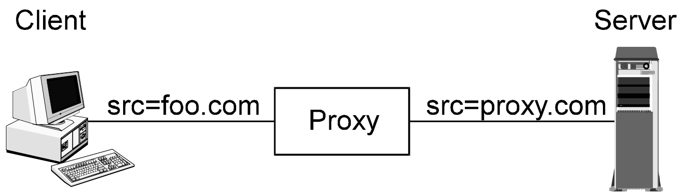
- Secret design process with no public scrutiny or external review
- Government interference to ensure poor security

Traffic Analysis

Monitors presence of communications and source/destination

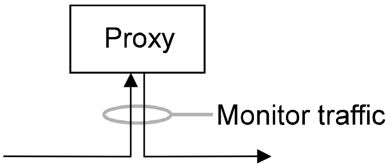
- Most common is analysis of web server logs
- Search engines reveal information on popularity of pages
- The mere presence of communications can reveal information

Simple Anonymiser Proxy



HTTP version at `http://www.anonymizer.com`

Fairly easy to defeat:

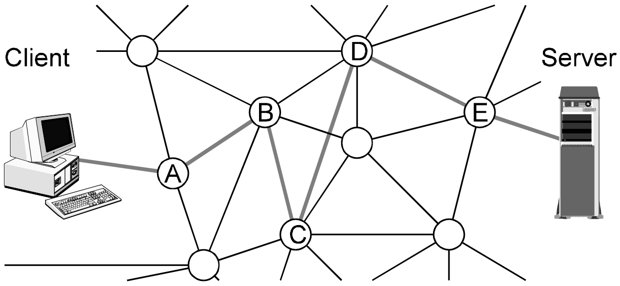


Mixes

Encrypted messages sent over user-selected route through a network

- Packet = A(B(C(D(E(data)))))
- Each server peels off a layer and forwards the data

Servers can only see one hop



Sender and receiver can’t be (easily) linked

Attacks on Mixes

Incoming messages result in outgoing messages

- Reorder messages
- Delay messages

Message sizes change in a predictable manner

Replay message (spam attack)

- Many identical messages will emerge at some point

Onion Routing

Message routing using mixes, `http://www.itd.nrl.navy.mil/ITD/5540/projects/onion-routing`

Routers have permanent socket connections

Data is sent over short-term connections tunnelled over permanent connections

- 5-layer onions
- 48-byte datagrams
- CREATE/DESTROY for connection control
- DATA/PADDING to move datagrams
- Limited form of datagram reordering
- Onions are padded to compensate for removed layers

Mixmaster

Uses message ID's to stop replay attacks

Message sizes never change

- 'Used' headers are moved to the end, remaining headers are moved up one
- Payload is padded to a fixed size
- Large payloads are broken up into multiple messages
- All parts of the message are encrypted

Encryption is 1024 bit RSA with triple DES

Message has 20 headers of 512 bytes and a 10K body

Crowds

Mixes have two main problems

- Routers are a vulnerable attack point
- Requires static routing

Router vulnerability solved via jondo (anonymous persona)

Messages are forwarded to a random jondo

- Can't tell whether a message originates at a given jondo
- Message and reply follow the same path

Steganography

From the Greek for “hidden writing”, secures data by hiding rather than encryption

- Encryption is usually used as a first step before steganography

Encrypted data looks like white noise

Steganography hides this noise in other data

- By replacing existing noise
- By using it as a model to generate innocuous-looking data

Hiding Information in Noise

All data from analogue sources contains noise

- Background noise
- Sampling/quantisation error
- Equipment/switching noise

Extract the natural noise and replace it with synthetic noise

- Replace least significant bit(s)
- Spread-spectrum coding
- Various other modulation techniques

Examples of channels

- Digital images (PhotoCD, GIF, BMP, PNG)
- Sound (WAV files)
- ISDN voice data

Generating Synthetic Data

Usually only has to fool automated scanners

- Needs to be good enough to get past their detection threshold

Two variants

- Use a statistical model of the target language to generate plausible-looking data
  - "Wants to apply more or right is better than this mechanism. Our only way is surrounded by radio station. When leaving. This mechanism is later years".
  - Works like a text compressor in reverse
  - Can be made arbrtrarily close to real text

Generating Synthetic Data (ctd)

- Use a grammatical model of actual text to build plausible-sounding data
  - "{Steganography|Stego} provides a {means|mechanism} for {hiding|encoding} {hidden|secret} {data|information} in {plain|open} {view|sight}".
  - More work than the statistical model method, but can provide a virtually undetectable channel

Problems with steganography

- The better the steganography, the lower the bandwidth

Main use is as an argument against crypto restrictions

Watermarking

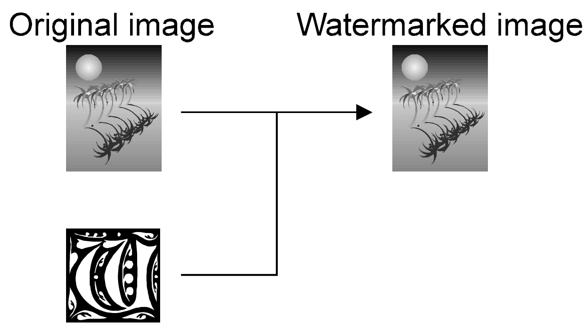
Uses redundancy in image/sound to encode information

Requirements

- Invisibility
- Little effect on compressability
- Robustness
- High detection reliability
- Security
- Inexpensive

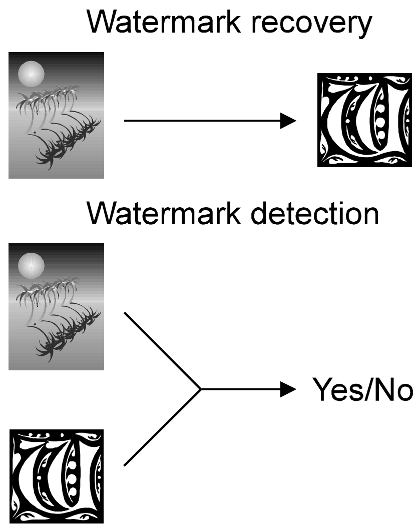
Watermarking (ctd)

Watermark insertion



Watermarking (ctd)

Watermark detection/checking



Watermarking (ctd)

Public watermarking

- Anyone can detect/view the watermark (and try to remove it)

Private watermarking

- Creator can demonstrate ownership using a secret key

Copy Protection Working Group (CPTWG) looking at standardisation, `http://www.dvcc.com/dhsg`

Defeating Watermarking

Lossy compression (JPEG)

Resizing

Noise insertion (print+scan)

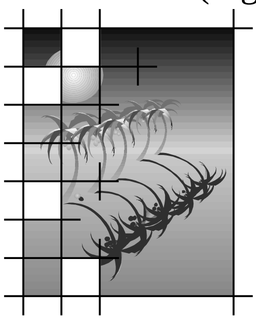
Cropping

Interpretation attacks (neutralise ownership evidence)

Automated anti-watermarking software available (eg UnZign)

Defeating Watermarking (ctd)

Presentation attacks (segmented images)



Watermarking is still in its infancy

- No watermarking standards
- No indication of security/benchmarks
- No legal recognition

Other Crypto Applications

Hashcash

- Requires finding a collision for *n* bits of a hash function
  - “Find a message for which the last 16 bits of the SHA-1 hash are 1F23”
- Forces a program to expend a (configurable) amount of effort before access is granted to a system or service
- Useful for stopping denial-of-service attacks
  - *n* varies as the system load goes up or down
  - Can be used as a spam-blocker

Other Crypto Applications (ctd)

PGP Moose

- Signs all postings to moderated newsgroups
  - Signature is added to the message as an `X-Auth` header
- Unsigned messages (spam, forgeries) are automatically cancelled
- Has so far proven 100% effective in stopping newsgroup spam/forgeries