CS 155 Spring 2006

Cryptography Overview

John Mitchell

Cryptography

- Is
  - A tremendous tool
  - The basis for many security mechanisms
- Is not
  - The solution to all security problems
  - Reliable unless implemented properly
  - Reliable unless used properly
  - Something you should try to invent yourself unless
    - you spend a lot of time becoming an expert
    - you subject your design to outside review

Basic Cryptographic Concepts

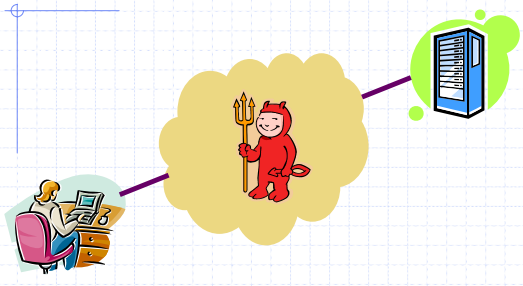
- Encryption scheme:
  - functions to encrypt, decrypt data
  - key generation algorithm
- Secret key vs. public key
  - Public key: publishing *key* does not reveal key^{-1}
  - Secret key: more efficient, generally $key = key^{-1}$
- Hash function, MAC
  - Map input to short hash; ideally, no collisions
  - MAC (keyed hash) used for message integrity
- Signature scheme
  - Functions to sign data, verify signature

Five-Minute University

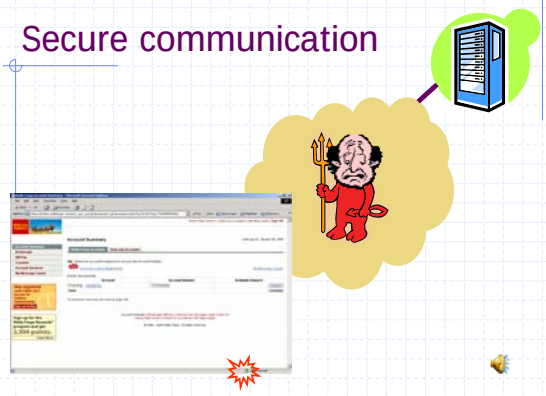
Father Guido Sarducci

- Everything you could remember, five years after taking CS255 ... ?

Web Purchase



Secure communication



Secure Sockets Layer / TLS

- Standard for Internet security
  - Originally designed by Netscape
  - Goal: "… provide privacy and reliability between two communicating applications"
- Two main parts
  - Handshake Protocol
    - Establish shared secret key using public-key cryptography
    - Signed certificates for authentication
  - Record Layer
    - Transmit data using negotiated key, encryption function

SSL/TLS Cryptography

- Public-key encryption
  - Key chosen secretly (handshake protocol)
  - Key material sent encrypted with public key
- Symmetric encryption
  - Shared (secret) key encryption of data packets
- Signature-based authentication
  - Client can check signed server certificate
  - And vice-versa, in principal
- Hash for integrity
  - Client, server check hash of sequence of messages
  - MAC used in data packets (record protocol)

Example cryptosystems

- One-time pad
  - "Theoretical idea," but leads to stream cipher
- Feistel construction for symmetric key crypto
  - Iterate a "scrambling function"
  - Examples: DES, Lucifer, FREAL, Khufu, Khafre, LOKI, GOST, CAST, Blowfish, ...
  - AES (Rijndael) is also block cipher, but different
- Complexity-based public-key cryptography
  - Modular exponentiation is a "one-way" fctns
  - Examples: RSA, El Gamal, elliptic curve systems, ...

One-time pad

- Secret-key encryption scheme (symmetric)
  - Encrypt plaintext by xor with sequence of bits
  - Decrypt ciphertext by xor with same bit sequence
- Scheme for pad of length n
  - Set P of plaintexts: all n-bit sequences
  - Set C of ciphertexts: all n-bit sequences
  - Set K of keys: all n-bit sequences
  - Encryption and decryption functions

encrypt(key, text) = key $\oplus$ text (bit-by-bit)
decrypt(key, text) = key $\oplus$ text (bit-by-bit)

Evaluation of one-time pad

- Advantages
  - Easy to compute encrypt, decrypt from key, text
  - As hard to break as possible
    - This is an information-theoretically secure cipher
    - Given ciphertext, all possible plaintexts are equally likely, assuming that key is chosen randomly
- Disadvantage
  - Key is as long as the plaintext
    - How does sender get key to receiver securely?

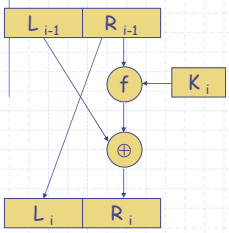
Idea for stream cipher: use pseudo-random generators for key...

Feistel networks

- Many block algorithms are *Feistel networks*
  - A block cipher encrypts data in blocks
    - Encryption of block n+1 *may* depend on block n
  - Feistel network is a standard construction for
    - Iterating a function f on parts of a message
    - Producing an invertible transformation
- AES (Rijndael) is related
  - Also a block cipher with repeated rounds
  - Not a Feistel network

Feistel network: One Round

Divide n-bit input in half and repeat



- Scheme requires
  - Function $f(R_{i-1}, K_i)$
  - Computation for K_i
    - e.g., permutation of key K
- Advantage
  - Systematic calculation
    - Easy if f is table, etc.
  - Invertible if K_i known
    - Get R_{i-1} from L_i
    - Compute $f(R_{i-1}, K_i)$
    - Compute L_{i-1} by ⊕

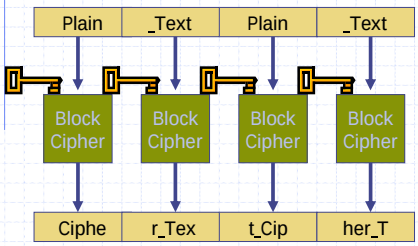
Data Encryption Standard

- Developed at IBM, some input from NSA, widely used
- Feistel structure
  - Permute input bits
  - Repeat application of a *S-box* function
  - Apply inverse permutation to produce output
- Worked well in practice (but brute-force attacks now)
  - Efficient to encrypt, decrypt
  - Not provably secure
- Improvements
  - Triple DES, AES (Rijndael)

Block cipher modes (for DES, AES, ...)

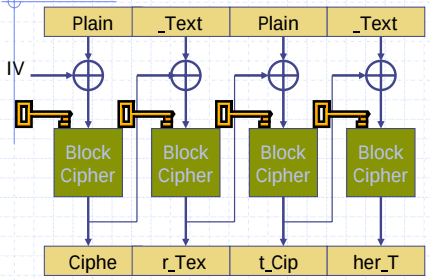
- ECB – Electronic Code Book mode
  - Divide plaintext into blocks
  - Encrypt each block independently, with same key
- CBC – Cipher Block Chaining
  - XOR each block with encryption of previous block
  - Use initialization vector IV for first block
- OFB – Output Feedback Mode
  - Iterate encryption of IV to produce stream cipher
- CFB – Cipher Feedback Mode
  - Output block y_i = input x_i ⊕ $encyrpt_K(y_{i-1})$

Electronic Code Book (ECB)



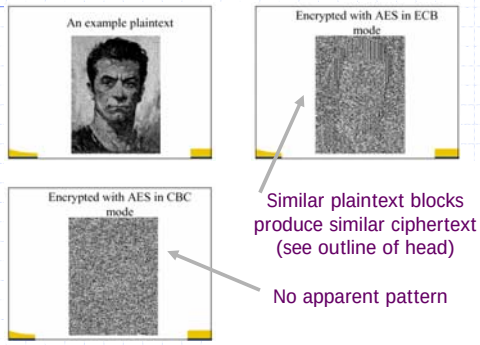
Problem: Identical blocks encrypted identically
No integrity check

Cipher Block Chaining (CBC)



Advantages: Identical blocks encrypted differently
Last ciphertext block depends on entire input

Comparison (for AES, by Bart Preneel)



Similar plaintext blocks produce similar ciphertext (see outline of head)

No apparent pattern

RC4 stream cipher – "Ron's Code"

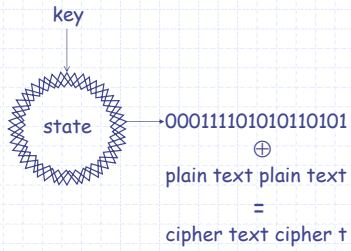
- Design goals (Ron Rivest, 1987):
  - speed
  - support of 8-bit architecture
  - simplicity (circumvent export regulations)
- Widely used
  - SSL/TLS
  - Windows, Lotus Notes, Oracle, etc.
  - Cellular Digital Packet Data
  - OpenBSD pseudo-random number generator

RSA Trade Secret

- History
  - 1994 – leaked to cypherpunks mailing list
  - 1995 – first weakness (USENET post)
  - 1996 – appeared in Applied Crypto as "alleged RC4"
  - 1997 – first published analysis

Weakness is predictability of first bits; best to discard them

Encryption/Decryption



Stream cipher: one-time pad based on pseudo-random generator

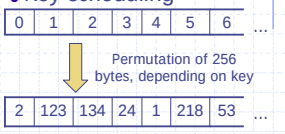
Security

- Goal: indistinguishable from random sequence
  - given part of the output stream, it is impossible to distinguish it from a random string
- Problems
  - Second byte [MS01]
    - Second byte of RC4 is 0 with twice expected probability
  - Related key attack [FMS01]
    - Bad to use many related keys (see WEP 802.11b)
- Recommendation
  - Discard the first 256 bytes of RC4 output [RSA, MS]

Complete Algorithm (all arithmetic mod 256)

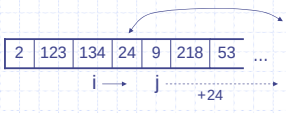
```
for i := 0 to 255    S[i] := i
j := 0
for i := 0 to 255
    j := j + S[i] + key[i]
    swap (S[i], S[j])
```

- Key scheduling

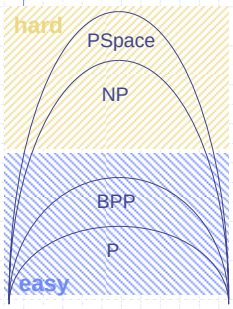



```
i, j := 0
repeat
    i := i + 1
    j := j + S[i]
    swap (S[i], S[j])
    output (S[ S[i] + S[j] ])
```

- Random generator



Complexity Classes



Answer in polynomial space
may need exhaustive search

If yes, can guess and check in polynomial time

Answer in polynomial time, with high probability

Answer in polynomial time
compute answer directly

One-way functions

- A function f is one-way if it is
  - Easy to compute f(x), given x
  - Hard to compute x, given f(x), for most x
- Examples (we believe they are one way)
  - f(x) = divide bits x = y@z and multiply f(x)=y*z
  - f(x) = 3^x mod p, where p is prime
  - f(x) = x^3 mod pq, where p,q are primes with |p|=|q|

One-way trapdoor

- A function f is *one-way trapdoor* if
  - Easy to compute f(x), given x
  - Hard to compute x, given f(x), for most x
  - Extra "trapdoor" information makes it easy to compute x from f(x)
- Example (we believe)
  - f(x) = x^3 mod pq, where p,q are primes with |p|=|q|
  - Compute cube root using (p-1)*(q-1)

Public-key Cryptosystem

- Trapdoor function to encrypt and decrypt
  - encrypt(key, message)

    key pair
  - decrypt(key $^{-1}$, encrypt(key, message)) = message
- Resists attack
  - Cannot compute m from encrypt(key, m) and key, unless you have key^{-1}

Example: RSA

- Arithmetic modulo pq
  - Generate secret primes p, q
  - Generate secret numbers a, b with $x^{ab} \equiv x \bmod \overbrace{pq}^{n}$
- Public encryption key $\langle n, a \rangle$
  - Encrypt($\langle n, a \rangle$, x) = x^a mod n
- Private decryption key $\langle n, b \rangle$
  - Decrypt($\langle n, b \rangle$, y) = y^b mod n
- Main properties
  - This works
  - Cannot compute b from n,a
    - *Apparently,* need to factor n = pq

How RSA works (quick sketch)

- Let p, q be two distinct primes and let n=p*q
  - Encryption, decryption based on group Z_n^*
  - For n=p*q, order $\phi(n) = (p-1)*(q-1)$
    - Proof: (p-1)*(q-1) = p*q - p - q + 1
- Key pair: $\langle a, b \rangle$ with $ab \equiv 1 \bmod \phi(n)$
  - Encrypt(x) = x^a mod n
  - Decrypt(y) = y^b mod n
  - Since $ab \equiv 1 \bmod \phi(n)$, have $x^{ab} \equiv x \bmod n$
    - Proof: if gcd(x,n) = 1, then by general group theory, otherwise use "Chinese remainder theorem".

How well does RSA work?

- Can generate modulus, keys fairly efficiently
  - Efficient rand algorithms for generating primes p,q
    - May fail, but with low probability
  - Given primes p,q easy to compute n=p*q and $\phi(n)$
  - Choose a randomly with gcd(a, $\phi(n)$)=1
  - Compute b = a-1 mod $\phi(n)$ by Euclidean algorithm
- Public key n, a does not reveal b
  - This is not proven, but believed
- But if n can be factored, all is lost …

Public-key crypto is significantly slower than symmetric key crypto

Message integrity

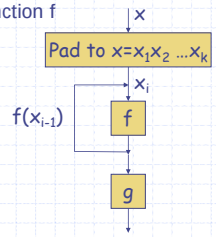
- For RSA as stated, integrity is a weak point
  - encrypt(k*m) = $(k*m)^e = k^e * m^e$
    = encrypt(k)*encrypt(m)
  - This leads to "chosen ciphertext" form of attack
    - If someone will decrypt *new* messages, then can trick them into decrypting m by asking for decrypt(k^e *m)
- Implementations reflect this problem
  - "The PKCS#1 ... RSA encryption is intended primarily to provide confidentiality. ... It is not intended to provide integrity." RSA Lab. Bulletin
- Additional mechanisms provide integrity

Cryptographic hash functions

- Length-reducing function h
  - Map arbitrary strings to strings of fixed length
- One way ("preimage resistance")
  - Given y, hard to find x with h(x)=y
- Collision resistant
  - Hard to find any distinct m, m' with h(m)=h(m')
- Also useful: 2nd preimage resistance
  - Given x, hard to find x'≠x with h(x')=h(x)
  - Collision resistance ⇒ 2nd preimage resistance

Iterated hash functions

- Repeat use of block cipher or custom function
  - Pad input to some multiple of block length
  - Iterate a length-reducing function f
    - f : 2^{2k} -> 2^k reduces bits by 2
    - Repeat h_0= some seed
      $h_{i+1} = f(h_i, x_i)$
  - Some final function g completes calculation



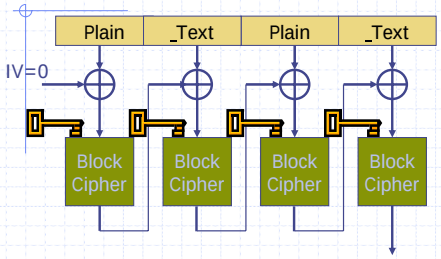
Applications of one-way hash

- Password files (one way)
- Digital signatures (collision resistant)
  - Sign hash of message instead of entire message
- Data integrity
  - Compute and store hash of some data
  - Check later by recomputing hash and comparing
- Keyed hash for message authentication
  - MAC – Message Authentication Code

MAC: Message Authentication Code

- General pattern of use
  - Sender sends Message & MAC(Message), M1
  - Receiver receives both parts
  - Receiver makes his own MAC(Message),M2
  - If M2 != M1, data has been corrupted
  - If M2 == M1, data is valid
- Need for shared key
  - Suppose an attacker can compute MAC(x)
  - Intercept M and Hash(M) and resend as M' and Hash(M')
  - Receiver cannot detect that message has been altered.

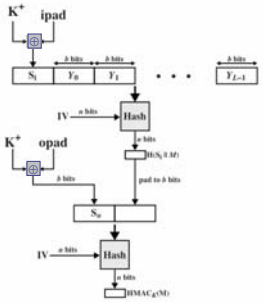
Basic CBC-MAC



CBC block cipher, discarding all but last output block
Additional post-processing (e.g, encrypt with second key) can improve output

HMAC: Keyed Hash-Based MAC

- Internet standard RFC2104
- Uses hash of key, message:
  $HMAC_K(M)$
  = Hash[(K^+ XOR opad) ||
  Hash[(K^+ XOR ipad)||M)]]
- Low overhead
  - opad, ipad are constants
- Any of MD5, SHA-1, RIPEMD-160, ... can be used



K+ is the key padded out to size

Slides: A.K. Lenstra, B. de Weger

Hash cryptanalysis before Aug '04

- MD4 considered broken: Den Boer, Bosselaers, and Dobbertin,
  - 1996, 'meaningful' collisions
- MD5 potentially weak: Dobbertin,
  - 1996, collisions in the MD5 compression function
- Iterated hash functions for which compression function
  - fixed points can be found (i.e., all hashes in the SHA family):
  - Drew Dean et al. (1999) found 2nd preimage weakness
    - (hidden in Dean's thesis, never published)
- MD5 and up (128-bit keys or greater):
  - security of practical applications not seriously questioned
- Strong belief in effectiveness of tweaks

Subsequent developments

- August 2004:
  - X. Wang et al.: actual random collisions in MD4 ('no time'),
    - MD5 in time $\approx 2^{39}$, etc., for any IV
  - A. Joux: cascading of iterated L-bit and perfect M-bit hash
    - does not result in L+M-bit hash – as commonly believed
  - A. Joux: actual random collision for SHA-0 in time $\approx 2^{51}$
  - E. Biham: cryptanalysis of SHA-1 variants
- October 2004, Kelsey/Schneier (based on Joux):
  - 2nd preimage weakness in any iterated hash (improving Dean)
- Feb 14, 2005, X. Wang et al. (based on Wang/Joux/Biham):
  - actual random collision for SHA-0 in time $\approx 2^{39}$
  - random collision possibility for SHA-1 in time $\approx 2^{69}$ (or 2^{66})
    (advantage: $2^{69} < 2^{80}$)

Digital Signatures

- Public-key encryption
  - Alice publishes encryption key
  - Anyone can send encrypted message
  - Only Alice can decrypt messages with this key
- Digital signature scheme
  - Alice publishes key for verifying signatures
  - Anyone can check a message signed by Alice
  - Only Alice can send signed messages

Properties of signatures

- Functions to sign and verify
  - Sign(Key^{-1}, message)
  - Verify(Key, x, m) = { *true* if x = Sign(Key^{-1}, m); *false* otherwise }
- Resists forgery
  - Cannot compute Sign(Key^{-1}, m) from m and Key
  - Resists existential forgery:
    given Key, cannot produce Sign(Key^{-1}, m)
    for any random or otherwise arbitrary m

RSA Signature Scheme

- Publish decryption instead of encryption key
  - Alice publishes decryption key
  - Anyone can decrypt a message encrypted by Alice
  - Only Alice can send encrypt messages
- In more detail,
  - Alice generates primes p, q and key pair $\langle a, b \rangle$
  - Sign(x) = $x^a \bmod n$
  - Verify(y) = $y^b \bmod n$
  - Since $ab \equiv 1 \bmod \phi(n)$, have $x^{ab} \equiv x \bmod n$

Public-Key Infrastructure (PKI)

- Anyone can send Bob a secret message
  - Provided they know Bob's public key
- How do we know a key belongs to Bob?
  - If imposter substitutes another key, read Bob's mail
- One solution: PKI
  - Trusted root authority (VeriSign, IBM, United Nations)
    - Everyone must know the verification key of root authority
    - Check your browser; there are hundreds!!
  - Root authority can sign certificates
  - Certificates identify others, including other authorities
  - Leads to certificate chains

Slides: A.K. Lenstra, B. de Weger

X.509 certificate

- X.509 allows data with this format to be hashed and signed: $p_1 || m || p_2$

  where
  - p_1 contains header, distinguished names, and
    - header of public key part,
    - may assume that $p1$ consists of whole number of blocks
  - m is an RSA modulus
  - p_2 contains public exponent, other data

Trick: can choose m cleverly to get collision

Constructing a collision

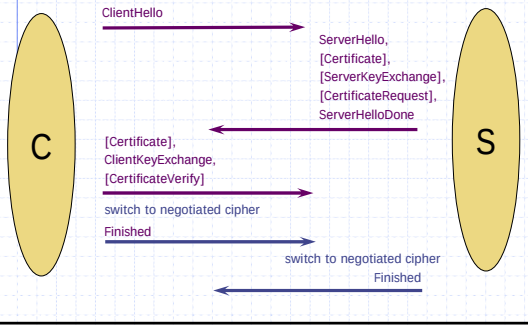
- If collisions can be found for any IV, then collisions can be concocted such that they have same prescribed initial blocks
- Proper (and identical) data appended to random data pairs turns random pair plus appendix into pair of valid RSA moduli
- Arbitrarily selected data can be appended to colliding messages of same length, and they will still collide

1 & 3: due to iterative nature of hashes
2: a new trick for RSA moduli construction

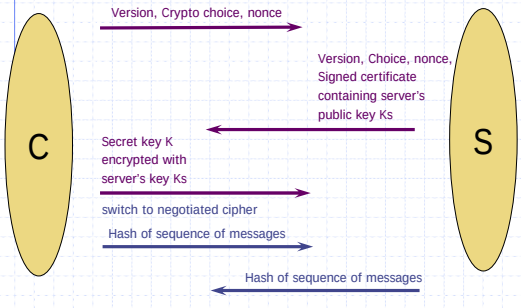
Some details

- Construct colliding $p_1 || m || p_2$ and $p_1 || m' || p_2$ as follows:
- Prepend:
  - pick properly formatted p1 with names etc., whole # blocks
  - compute p_1's intermediate hash value h
  - ask X. Wang to find random collision m_1, m_2 with h as IV
  - $p_1||m_1$ and $p_1||m_2$ now collide as well
- Promote:
  - find m_3 s.t. $m_1||m_3$ = m and $m_2||m_3$ = m' are RSA moduli
  - random m_1, m_2 extended to meaningful $m_1||m_3$ and $m_2||m_3$
- Append:
  - $p_1||m_1||m_3 = p_1||$ m and $p_1||m_2||m_3 = p_1||$ m' still collide and so do $p_1||$ m $||p_2$ and $p_1||$ m' $||p_2$ for any p_2

Back to TLS



Use of cryptography



More detail ...

ClientHello C → S C, Ver_C, $Suite_C$, N_C

ServerHello S → C Ver_S, $Suite_S$, N_S, $sign_{CA}\{ S, K_S \}$

ClientVerify C → S $sign_{CA}\{ C, V_C \}$
$\{ Ver_C, Secret_C \} K_S$
$sign_C \{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1)) \}$

(Change to negotiated cipher)

ServerFinished S → C $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + S + Master(N_C, N_S, Secret_C) + Pad_1))$
$\} Master(N_C, N_S, Secret_C)$

ClientFinished C → S $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1))$
$\} Master(N_C, N_S, Secret_C)$

Crypto Summary

- Encryption scheme:
  - encrypt(key, plaintext) decrypt(key^{-1},ciphertext)
- Secret vs. public key
  - Public key: publishing key does not reveal key^{-1}
  - Secret key: more efficient; can have key = key^{-1}
- Hash function
  - Map long text to short hash; ideally, no collisions
  - Keyed hash (MAC) for message authentication
- Signature scheme
  - Private key^{-1} and public key provide authentication

Limitations of cryptography

- Most security problems are not crypto problems
  - This is good
    - Cryptography works!
  - This is bad
    - People make other mistakes; crypto doesn't solve them
- Examples
  - Deployment and management problems [Anderson]
  - Ineffective use of cryptography
    - Example 802.11b WEP protocol

Why cryptosystems fail [Anderson]

- Security failures not publicized
  - Government: top secret
  - Military: top secret
  - Private companies
    - Embarrassment
    - Stock price
    - Liability
- Paper reports problems in banking industry
  - Anderson hired as consultant representing unhappy customers in 1992 class action suit

Anderson study of bank ATMs

- US Federal Reserve regulations
  - Customer not liable unless bank proves fraud
- UK regulations significantly weaker
  - Banker denial and negligence
  - Teenage girl in Ashton under Lyme
    - Convicted of stealing from her father, forced to plead guilty, later determined to be bank error
  - Sheffield police sergeant
    - Charged with theft and suspended from job; bank error
- 1992 class action suit

Sources of ATM Fraud

- Internal Fraud
  - PINs issued through branches, not post
    - Bank employees know customer's PIN numbers
  - One maintenance engineer modified an ATM
    - Recorded bank account numbers and PINs
  - One bank issues "master" cards to employees
    - Can debit cash from customer accounts
  - Bank with good security removed control to cut cost
    - No prior study of cost/benefit; no actual cost reduction
    - Increase in internal fraud at significant cost
    - Employees did not report losses to management out of fear

Sources of ATM Fraud

- External Fraud
  - Full account numbers on ATM receipts
    - Create counterfeit cards
      - Attackers observe customers, record PIN
      - Get account number from discarded receipt
    - One sys: Telephone card treated as previous bank card
      - Apparently programming bug
      - Attackers observe customer, use telephone card
  - Attackers produce fake ATMs that record PIN
  - Postal interception accounts for 30% of UK fraud
    - Nonetheless, banks have poor postal control procedures
  - Many other problems
    - Test sequence causes ATM to output 10 banknotes

Sources of ATM Fraud

- PIN number attacks on lost, stolen cards
  - Bank suggestion of how to write down PIN
    - Use weak code; easy to break
  - Programmer error - all customers issued same PIN
  - Banks store encrypted PIN on file
    - Programmer can find own encrypted PIN, look for other accounts with same encrypted PIN
  - One large bank stored encrypted PIN on mag strip
    - Possible to change account number on strip, leave encrypted PIN, withdraw money from other account

Additional problems

- Some problems with encryption products
  - Special hardware expensive; software insecure
  - Banks buy bad solutions when good ones exist
    - Not knowledgeable enough to tell the difference
  - Poor installation and operating procedures
  - Cryptanalysis possible for homegrown crypto

More sophisticated attacks described in paper

Wider Implications

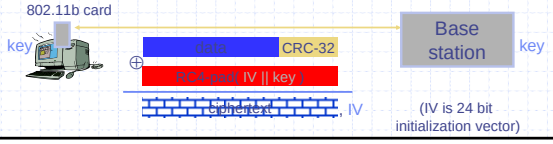
- Equipment designers and evaluators focus on technical weaknesses
  - Banking systems have some loopholes, but these do not contributed significantly to fraud
- Attacks were made possible because
  - Banks did not use products properly
  - Basic errors in
    - System design
    - Application programming
    - Administration

Summary

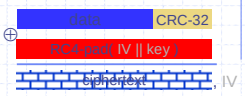
- Cryptographic systems suffer from lack of failure information
  - Understand all possible failure modes of system
  - Plan strategy to prevent each failure
  - Careful implementation of each strategy
- Most security failures due to implementation and management error
  - Program must carried out by personnel available

Last mile security: wireless Ethernet

- Many corporate wireless hubs installed without any privacy or authentication.
  - POP/IMAP passwords easily sniffed off the air.
  - Laptops in parking lot can access internal network.
- Intended "solution": use the WEP protocol (802.11b).
  - Provides 40-bit or 128-bit encryption using RC4 ...



Some mistakes in the design of WEP



- CRC-32 ⇒ no packet integrity!!
  - CRC-32 is linear
  - Attacker can easily modify packets in transit, e.g. inject "rm –rf *"
  - Should use MAC for integrity
- Prepending IV is insufficient.
  - Fluhrer-Mantin-Shamir: RC4 is insecure in prepending IV mode
    - Given 106 packets can get key.
    - Implemented by Stubblefield, AirSnort, WEPCrack, ...
  - Correct construction:
    - packet-key = SHA-1(IV || key)
    - use longer IV, random.

What to do?

- Regard 802.11b networks as public channels.
  - Use SSH, SSL, IPsec, ...
- Lesson:
  - Insist on open security reviews for upcoming standards
  - Closed standards don't work: e.g. GSM, CMEA, ...
  - Open review worked well for SSL and IPsec

Summary

- Main functions from cryptography
  - Public-key encryption, decryption, key generation
  - Symmetric encryption
    - Block ciphers, CBC Mode
    - Stream cipher
  - Hash functions
    - Cryptographic hash
    - Keyed hash for Message Authentication Code (MAC)
  - Digital signatures
- Be careful
  - Many non-intuitive properties; prefer public review
  - Need to implement, use carefully