Crypto Protocols, part 2
Crypto primitives

Today’s talk includes slides from:
Bart Preneel, Jonathan Millen, and Dan Wallach

Comp527 status items

- Install the smart card software
  - Bring CDs back to Dan’s office (DH3004)
- Today
  - Finish crypto protocols from Monday
  - Start on crypto primitives

Example - Needham-Schroeder

- The Needham-Schroeder symmetric-key protocol [NS78]
  - A -> S: A, B, Na
  - S -> A: (Na, B, Kc, (Kc, A)Kb)Ka
  - A -> B: (Kc, A)Kb
  - B -> A: (Nb)Kc
  - A -> B: (Nb-1)Kc
- A, B are “principals;” S is a trusted key server
- Ka, Kb are secret keys shared with S
- (X, Y)K means: X concatenated with Y, encrypted with K
- Na, Nb are “nonces;” fresh (not used before)
- Kc is a fresh connection key

Denning-Sacco Attack

- Assumes that the attacker has recorded a previous session, and compromised the connection key Kx used in that one.
  - A -> B: (Kx, A)Kb attacker replayed old message
  - B -> A: (Nb)Kx
  - A -> B: (Nb-1)Kx forged by attacker
- B now believes he shares a fresh secret key Kx with A.
- Denning-Sacco moral: use a timestamp (calendar clock value) to detect replay of old messages.
Belief Logic

- Burrows, Abadi, and Needham (BAN) Logic [BAN90a]
  - Modal logic of belief ("belief" as local knowledge)
  - Special constructs and inference rules
    - Protocol messages are "idealized" into logical statements
    - Objective is to prove that both parties share common beliefs

Constructs

- $P \text{ bel } X$ \quad P believes $X$
- $P \text{ sees } X$ \quad P received $X$ in a message
- $P \text{ said } X$ \quad P once said $X$
- $P \text{ controls } X$ \quad P has jurisdiction over $X$
- $\text{fresh}(X)$ \quad $X$ has not been used before
- $P \leftrightarrow K \leftrightarrow Q$ \quad P and Q may use key $K$ for private communication
- $K \leftrightarrow P$ \quad P has $K$ as public key
- $P \leftrightarrow X \leftrightarrow Q$ \quad $X$ is a secret shared by $P$ and $Q$
- $(X)K$ \quad $X$ encrypted under $K$
- $\langle X \rangle Y$ \quad $X$ combined with $Y$
- $K^{-1}$ \quad inverse key to $K$

(This symbolism is not quite standard)

BAN Inference Rules

- These inferences are supposed to be valid despite attacker interference.
  (1) Message-meaning rules
    - $P \text{ bel } Q \leftarrow K \leftarrow P$ \quad $P$ sees $(X)K$
    - $P \text{ bel } Q \leftarrow P$ \quad $P$ sees $(X)$
    - $P \text{ bel } K \leftarrow Q$, $P$ sees $(X)K$
    - $P \text{ bel } K \leftarrow Y \rightarrow P$, $P$ sees $(X)Y$
  (2) Nonce-verification
    - $P \text{ bel fresh}(X)$, $P$ bel $Q$ said $X$ \quad $P$ bel $Q$ bel $X$
  (3) Jurisdiction
    - $P \text{ bel } Q$ controls $X$, $P$ bel $Q$ bel $X$ \quad $P$ bel $X$

More BAN Rules

(4) Sees rules
  - $P \text{ sees } (X, Y)$ \quad $P$ sees $X$, $P$ sees $Y$
  - $P \text{ sees } \langle X \rangle Y$ \quad $P$ sees $X$
  - $P \text{ bel } Q \leftarrow K \leftarrow P$, $P$ sees $(X)K$
  - $P \text{ bel } K \leftarrow P$, $P$ sees $(X)K$
  - $P \text{ bel } K \leftarrow Q$, $P$ sees $(X)K^{-1}$
(5) Freshness
  - $P \text{ bel fresh}(X)$ \quad $P$ bel fresh$(X, Y)$ (inside encryption)

- Symmetry of $\leftarrow K$ and $\leftarrow X$ is implicitly used
- Conjunction is handled implicitly
  - $P \text{ bel } (X, Y)$ \quad $P$ bel $X$ and $P$ bel $Y$
  - $P \text{ bel } Q$ said $(X, Y)$ \quad $P$ bel $Q$ said $X$, $P$ bel $Q$ said $Y$
**Protocol Idealization**

- Convert a protocol into a collection of statements
  - Assumptions
  - Message idealizations
  - Security goals
- Message idealization conveys intent of message
  - Example: $A \rightarrow B$: $(A, (T, B, Kab))_Kbs$
  - Idealized: $B$ sees $(A \leftarrow Kab \rightarrow B)_Kbs$
- **Note:** only encrypted fields are retained in the idealization.

**Example – Wide-Mouthed Frog**

\[A \rightarrow S: A, (T, B, Kab)_Kas \rightarrow (M1) S \text{ sees } (T, A \leftarrow Kab \rightarrow B)_Kas\]
\[S \rightarrow B: (T, A, Kab)_Kbs \rightarrow (M2) B \text{ sees } (T, A \text{ bel } A \leftarrow Kab \rightarrow B)_Kbs\]

- (A1) $P$ bel fresh($T$), for $P = A, B, S$
- (A2) $B$ bel $A$ controls $A \leftarrow Kab \rightarrow B$
- (A3) $S$ bel $A \leftarrow Kas \rightarrow S, B$ bel $B \leftarrow Kbs \rightarrow S$
- (A4) $B$ bel $S$ controls $A$ bel $A \leftarrow Kab \rightarrow B$
- (A5) $A$ bel $A \leftarrow Kab \rightarrow B$

$T$ is a timestamp
$A$ generates $Kab$
$Kab, Kbs$ are shared with $S$
$S$ should check this
$A$ justifies $A$ said $A \leftarrow Kab \rightarrow B$

**Analysis**

- **Goal:** prove that $B$ bel $A \leftarrow Kab \rightarrow B$.
- **Proof:**
  - $B$ sees $(T, A \text{ bel } A \leftarrow Kab \rightarrow B)_Kbs$
  - $B$ bel $S$ said $(T, A \text{ bel } A \leftarrow Kab \rightarrow B)$
  - $B$ bel fresh($T$, $A$ bel $A \leftarrow Kab \rightarrow B$)
  - $B$ bel $S$ bel $T$, $A$ bel $A \leftarrow Kab \rightarrow B$
  - $B$ bel $S$ bel $A$ bel $A \leftarrow Kab \rightarrow B$
  - $B$ bel $A$ bel $A \leftarrow Kab \rightarrow B$
  - $B$ bel $A \leftarrow Kab \rightarrow B$
- **Exercises:**
  - Prove that $S$ bel $A$ bel $A \leftarrow Kab \rightarrow B$
  - Add the message $B \rightarrow A$: $(T)Kab$ (M3) and show that $A$ bel $B$ bel $A \leftarrow Kab \rightarrow B$

**Nessett’s Critique**

- Awkward example in [Nes90]
  - $A \rightarrow B$: $(T, Kab)_Kab^t \leftarrow B$ sees $(T, A \leftarrow Kab \rightarrow B)_Kab^t$
- **Assumptions**
  - (A1) $B$ bel $Kab$ to $A$
  - (A2) $A$ bel $A \leftarrow Kab \rightarrow B$
  - (A3) $B$ bel fresh($T$)
  - (A4) $B$ bel $A$ controls $A \leftarrow Kab \rightarrow B$
- **Goal:** $B$ bel $A \leftarrow Kab \rightarrow B$
- **Proof:**
  - $B$ bel $A$ said $(T, A \leftarrow Kab \rightarrow B)$
  - $B$ bel fresh($T$, $A \leftarrow Kab \rightarrow B$)
  - $B$ bel $A$ bel $(T, A \leftarrow Kab \rightarrow B)$
  - $B$ bel $A \leftarrow Kab \rightarrow B$
- **Problem:** $Ka$ is a public key, so $Kab$ is exposed.
**Observations**

- According to "Rejoinder" [BAN90b], "There is no attempt to deal with ... unauthorized release of secrets"
- The logic is monotonic: if a key is believed to be good, the belief cannot be retracted
- The protocol may be inconsistent with beliefs about confidentiality of keys and other secrets
- More generally - one should analyze the protocol for consistency with its idealization
- Alternatively - devise restrictions on protocols and idealization rules that guarantee consistency

**Subsequent Developments**

- Discussions and semantics, e.g., [Syv91]
- More extensive logics, e.g., GNY (Gong-Needham-Yahalom) [GNY90] and SVO [SvO94]
- GNY extensions:
  - Uncrypted fields retained
  - "P possesses X" construct and possession rules
  - "not originated here" operator
  - Rationality rule: if X |- Y then P bel X |- P bel Y
  - "message extension" links fields to assertions
- Mechanization of inference, e.g., [KW96, Bra96]
  - User still does idealization
- Protocol vs. idealization problem still unsolved

**Model-Checking**

- Application of software tools designed for hardware CAD: Verification by state space exploration - exhaustive on model
- Like earlier Prolog tool approach, but
  - Forward search rather than reverse search
  - Special algorithms (DDs, etc.)
  - A priori finite model (no unbounded recursion)
  - Fully automatic once protocol is encoded
- Practitioners:
  - Roscoe [Ros95], using FDR (the first)
  - Mitchell, et al, using Murphi (MM95)
  - Marrero, et al, using SMV [MCI97]
  - Denker, et al, using Maude [DMT98]
  - and more

**Model-Checking Observations**

- Very effective at finding flaws, but
- No guarantee of correctness, due to artificial finite bounds
- Setup and analysis is quick when done by experts
- Automatic translation from simple message-list format to model-checker input is possible [Low98a, Mi97]
- "Killer" example: Lowe attack on Needham-Schroeder public-key protocol, using FDR [Low96]
**NSPK Protocol**

- Na, Nb are nonces; PKA, PKB are public keys
- The protocol – final handshake
  - A → B: (Na, A)PKR
  - B → A: (Na, Nb)PKR
  - A → B: (Nb)PKR
- Exercise: use BAN Logic to prove
  - B bel A bel A → Nb → B  [BAN90a]

**Lowe Attack on NSPK**

- X is the attacker acting as a principal
- X masquerades as A for B

Session 1: A to X
A → X: (Na, A)PKR

Session 2: X (as A) to B
A(X) → B: (Na, A)PKB
B → A(X): (Na, Nb)PKR
X → A: (Na, Nb)PKR
A → X: (Nb)PKR
A(X) → B: (Nb)PKR

(Lowe’s modification to fix it: B → A: (Na, Nb, B)PKA)

**Finiteness Limitation**

- How many sessions must be simulated to ensure coverage?
  - Lowe attack needed two sessions
  - Example 1.3 in Dolev-Yao (DY83) needed three sessions
    - A → B: ((M)PKb, A)PKb
    - B → A: ((M)PKa, B)PKa
- No algorithmically determined bound is possible for all cases
  - Because of undecidability for the model
- Possible bounds for limited classes of protocols
  - Lowe “small system” result [Low98b]: one honest agent per role, one time, if certain restrictions are satisfied:
    - Encrypted fields are distinguishable
    - Principal identities in every encrypted field
    - No temporary secrets
    - No forwarding of encrypted fields

**Inductive Proofs**

- Approach: like proofs of program correctness
  - Induction to prove “loop invariant”
- State-transition model, objective is security invariant
- General-purpose specification/verification system support
  - Kemmerer, using Ina Jo and ITP [Kem99] (the first)
  - Paulson, using Isabelle [Paul98] (the new wave)
  - Dutertre and Schneider, using PVS [DSS97]
  - Bagnara, using Coq [Bol97]
- Can also be done manually [Sch98, THG98]
  - Contributed to better understanding of invariants
  - Much more complex than belief logic proofs
- Full guarantee of correctness (with respect to model)
  - Proofs include confidentiality
Summary

- Cryptographic protocol verification is based on models where
  - Encryption is perfect (strong encryption)
  - The attacker intercepts all messages (strong attacker)
  - Security is undecidable in general, primarily because the number
    of sessions is unbounded.
- Belief logic analysis:
  - Requires "idealization" of the protocol
  - Does not address confidentiality
  - Can be performed easily, manually or with automated support
- State-exploration approaches
  - Use model-checking tools
  - Are effective for finding flaws automatically
  - Are limited by finiteness

Summary, cont’d

- Inductive proofs
  - Can prove correctness
  - Require substantial effort
  - Can be done manually, but preferably with verification tools
- Protocol security verification is still a research area
  - But experts can do it fairly routinely
- "Real" protocols are difficult to analyze for practical reasons
  - Specifications are not precise
  - They use operators with more complex properties than simple
    abstract encryption
  - Flow of control is more complex – protocols negotiate alternative
    encryption algorithms and other parameters
  - Messages have many fields not relevant to provable security

Crypto primitives

- The building blocks of everything else
  - Stream ciphers
  - Block ciphers (& cipher modes)
- Far more material than we can ever cover
  - In addition to your book…
    - Nice reference, lots of details:
      - http://home.ecn.ab.ca/~jsvard/crypto/jscript.htm

Model of a practical stream cipher
LFSR based stream cipher

+ good randomness properties
+ mathematical theory
+ compact in hardware
- too linear: easy to predict after 2L output bits

A5/1 stream cipher (GSM)

A5/1 attacks
- exhaustive key search: $2^{64}$ (or rather $2^{54}$)
- search 2 smallest registers: $2^{45}$ steps
- [BWS00] 2 seconds of plaintext: 1 minute on a PC
  - $2^{48}$ precomputation, 146 GB storage

Clock control: registers agreeing with majority are clocked (2 or 3)

A5/1 stream cipher (GSM)

Bluetooth stream cipher

- best known shortcut attack: $2^{70}$ rather than $2^{128}$
Cryptanalysis of stream ciphers

- exhaustive key search (key of \( k \) bits)
  - \( 2^k \) encryptions, about \( k \) known plaintext bits
- time-memory trade-off (memory of \( m \) bits)
  - \( 2^m \) short output sequences
  - \( 2^{m-t} \) precomputation and memory

- linear complexity
- divide and conquer
- fast correlation attacks (decoding problem)


Generate key stream which is added to plaintext

\[
\begin{align*}
i & := i + 1 \\
j & := (j + S[i]) \mod 256 \\
\text{swap } S[i] \text{ and } S[j] \\
t & := (S[i] + S[j]) \mod 256 \\
\text{output } S[t]
\end{align*}
\]

<table>
<thead>
<tr>
<th>000</th>
<th>001</th>
<th>002</th>
<th>093</th>
<th>094</th>
<th>095</th>
<th>254</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>205</td>
<td>162</td>
<td>013</td>
<td>...</td>
<td>033</td>
<td>092</td>
<td>079</td>
<td>...</td>
</tr>
<tr>
<td>099</td>
<td>143</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


for \( i = 0 \) to 255 \( S[i] := i \)
\( j := 0 \)
for \( i = 0 \) to 255
  \( j := (j + S[i] + K[i]) \mod 256 \)
  swap \( S[i] \) and \( S[j] \)
\( i := 0, \ j := 0 \)

RC4: weaknesses

- often used with 40-bit key
  - US export restrictions until Q4/2000
- best known general shortcut attack: \( 2^{700} \)
- weak keys and key setup (shuffle theory)
- some statistical deviations
  - e.g., 2nd output byte is biased
  - solution: drop first 256 bytes of output
- problem with resynchronization modes (WEP)
Block cipher

- larger data units: 64...128 bits
- memoryless
- repeat simple operation (round) many times

How NOT to use a block cipher: ECB mode

An example plaintext

Encrypted with AES in ECB mode
Encrypted with AES in CBC mode

How to use a block cipher: CBC mode

CBC mode decryption

Secure encryption

- What is a secure block cipher anyway?
- What is secure encryption anyway?

- Definition of security
  - security assumption
  - security goal
  - capability of opponent
**Security assumption:**
the block cipher is a pseudo-random permutation

- It is hard to distinguish a block cipher from a random permutation
- Advantage of a distinguisher
  \[
  \text{Adv}_{\text{AES,PRP}} = \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]
  \]

**Security goal:** “encryption”

- **semantic security:** adversary with limited computing power cannot gain any extra information on the plaintext by observing the ciphertext
- **indistinguishability (real or random) [IND-ROR]:** adversary with limited computing power cannot distinguish the encryption of a plaintext \( P \) from a random string of the same length

\[
\begin{align*}
  x_0 &= \text{AES}_K(P) \\
  x_1 &= \text{PRP}(P) \\
  b = \text{?} \\
  b' = 0 \text{ or } 1?
\end{align*}
\]

**Cryptanalysis of block ciphers**

- exhaustive key search (key of \( k \) bits)
  - \( 2^k \) encryptions, \( k/n \) known plaintexts
- code book attack (block of \( n \) bits)
  - collect \( 2^n \) encryptions
- time-memory trade-off:
  - \( k/n \) chosen plaintexts
  - \( 2^k \) encryptions (precomputation)
  - on-line: \( 2^{2k/3} \) encryptions and memory
- differential cryptanalysis
- linear cryptanalysis

**Time-memory trade-off [Hellman]**

- \( f(x) \) is a one-way function: \( \{0,1\}^n \to \{0,1\}^n \)
- easy to compute, but hard to invert
- \( f(x) \) has \((\varepsilon, t)\) preimage security iff
  - choose \( x \) uniformly in \( \{0,1\}^n \)
  - let \( M \) be an adversary that on input \( f(x) \) needs time \( \leq t \) and outputs \( M(f(x)) \) in \( \{0,1\}^n \)
  - \( \Pr\{f(M(f(x))) = f(x) < \varepsilon \} \),
  - where the probability is taken over \( x \) and over all the random choices of \( M \)
- \( t/\varepsilon \) should be large
Time-memory trade-off (2)

- Consider the functional graph of $f$

\[ x \xrightarrow{f} f(x) \]

Time-memory trade-off (3)

- Choose $m$ different starting points and iterate for $t$ steps

! problem: collisions: $m \cdot t \ll 2^n$

The birthday paradox

- Given a set with $S$ elements
- Choose $q$ elements at random (with replacements) with $q \ll S$
- The probability $p$ that there are at least 2 equal elements is $1 - \exp(-q(q-1)/2S)$

- $S$ large, $q = \sqrt{S}$, $p = 0.39$
- $S = 365$, $q = 23$, $p = 0.50$

DES properties

- design: IBM + NSA (1977)
- 64-bit block cipher with a 56-bit key
- 16 iterations of a relatively simple mapping
- optimized for mid 1970ies hardware
- FIPS 41: US government standard for sensitive but unclassified data
- worldwide de facto standard since early 80ies
- surrounded by controversy: key length
Data Encryption Standard

Security of DES (56-bit key)

- PC: trying 1 DES key: 0.25 μs
- Trying all keys on 4000 PCs:
  1 month: $2^{22} \times 2^{16} \times 2^5 = 2^{55}$
- M. Wiener’s estimate (1993):
  1,000,000 $ machine: 35 minutes

EFF Deep Crack (July 1999)
250,000 $ machine: 50 hours…

Solution to DES key length

- Moore’s “law”: speed of computers doubles every 18 months
  - Conclusion: key lengths need to grow in time
- Use new algorithms with longer keys
- Or replace DES by triple-DES (168-bit key):

AES (Advanced Encryption Standard)

- Open competition launched by US government (’97)
- 21 contenders, 15 in first round, 5 finalists
- decision October 2, 2000
- 128-bit block cipher with long key (128/192/256 bits)
- five finalists:
  - MARS (IBM, US)
  - RC6 (RSA Inc, US)
  - Rijndael (KULeuven/PWI, BE)
  - Serpent (DK/IL/UK)
  - Twofish (Counterpane, US)
AES properties

- Rijndael: design by V. Rijmen (COSIC) and J. Daemen (Proton World, ex-COSIC)
- 128-bit block cipher with a 128/192/256-bit key
- 10/12/14 iterations of a relatively simple mapping
- optimized for software for 8/16/32/64-bit machines, also suitable for hardware

Design trade-off

security

low high

speed

O’Connor versus Massey

- Luke O’Connor
  “most ciphers are secure after sufficiently many rounds”
- James L. Massey
  “most ciphers are too slow after sufficiently many rounds”

AES Status

- FIPS 197 published on 6 December 2001
- Revised FIPS on modes of operation
- Rijndael has more options than AES
- fast adoption in the market
  – early 2002, 74 products are using AES
  – standardization: ISO, IETF, …
- slower adoption in financial sector
AES/Rijndael: 1 round

- SubBytes
- ShiftRows
- MixColumn
- AddRoundKey

1 round consists of 4 operations

State: 16 bytes = 128 bits

Rijndael round: SubBytes

256 byte table

mapping $x^{-1}$ over $GF(2^8)$, plus some affine transformation over $GF(2)$

Rijndael round: ShiftRows

Rijndael round: MixColumn

MixColumn

$$
\begin{pmatrix}
p_0 & p_1 & p_2 & p_3 \\
p_4 & p_5 & p_6 & p_7 \\
p_8 & p_9 & p_{10} & p_{11} \\
p_{12} & p_{13} & p_{14} & p_{15}
\end{pmatrix} =
\begin{pmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{pmatrix} \cdot
\begin{pmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
p_6 \\
p_7
\end{pmatrix}$$
Rijndael round: AddRoundKey

Rijndael design strategy

- simple and elegant
- no integer arithmetic
- wide trail strategy:
  - strong resistance against linear and differential attacks
  - over 4 rounds, sum of number of “active” input and output bytes equals 25
- diffusion based on (8,4) MDS code with minimum distance 5
  \[ [p_1 \ p_2 \ p_3 \ p_4 \mid p_1' \ p_2' \ p_3' \ p_4'] \]

Performance reference data
(Pentium)

Differential cryptanalysis [Biham Shamir90]

choose the difference \( P, P' \)

try to predict the difference \( C_{r-1}, C_{r-1}' \)

this leaks information on \( K_r \)

key setup (cycles)
Linear cryptanalysis [Matsui93]

for a non-perfect cipher, there exist values $\delta_0$, $\delta_{r-1}$ s.t.
$P.\delta_0 \oplus C_{r-1}.\delta_{r-1} = 0$
with probability $p \neq 1/2$

or $P.\delta_0 \oplus \text{round}^{\text{round}-1}(K_r, C).\delta_{r-1} = 0$
with probability $p \neq 1/2$

this leaks information on $K_r$

Linear and differential cryptanalysis

- hard to find good linear or differential attacks
  – it is even harder to prove that it is impossible to find good linear or differential attacks
  – for some ciphers, this proof exists
- there exist many optimizations and generalizations
  – it is even harder to show that none of these work for a particular cipher
- analysis requires some heuristics
- DES: linear analysis needs $2^{43}$ known texts and differential analysis needs $2^{47}$ chosen texts