CS1001

Lecture 24

Overview

- Encryption
- Artificial Intelligence
- Homework 4

Reading

- Brookshear, 11.6
- Brookshear, 10

Homework 4

- Check Courseworks
- Problems based on
  - Handout (Smullyan, Natural Deduction)
  - Question from Ch. 10
  - Question from Ch. 11

Some Trivial Schemes

- Caesar cipher: substitution cipher:
  - A $\rightarrow$ D, B $\rightarrow$ E
- Captain Midnight Secret Decoder rings:
  - shift variable by *n*: IBM $\rightarrow$ HAL, or :
    - (letter + offset) mod 26
  - only 26 possible ways of secret coding.
- Monoalphabetic cipher:
  - generalization, arbitrary mapping of one letter to another
  - 26!, approximately 4×10^{26}
  - statistical analysis of letter frequencies
- One-time pad
  - A random sequence of 0's and 1's XORed to plaintext

Definitions

- Process data into unintelligible form, reversible, without data loss
- Usually one-to-one (not compression)
- Other services:
  - Integrity checking: no tampering
  - Authentication: not an imposter
- Plaintext $\xrightarrow{\text{encryption}}$ ciphertext $\xrightarrow{\text{decryption}}$ plaintext

Computational Difficulty

- Algorithm needs to be efficient.
  - Otherwise only short keys can be used.
- Most schemes can be broken: depends on $$$.
  - E.G. Try all possible keys.
- Longer key is often more secure:
  - Encryption $O(N+1)$.
  - Brute-force cryptanalysis: $O(^{2N+1})$, twice as hard with each additional bit.
- Cryptanalysis tools:
  - Special-purpose hardware.
  - Parallel machines.
  - Internet coarse-grain parallelism.

Secret Key vs. Secret Algorithm

- Secret algorithm: additional hurdle
- Hard to keep secret if used widely:
  - Reverse engineering, social engineering
- Commercial: published
  - Wide review, trust
- Military: avoid giving enemy good ideas

Cryptanalysis: Breaking an Encryption Scheme

- Ciphertext only:
  - Exhaustive search until "recognizable plaintext"
  - Need enough ciphertext
- Known plaintext:
  - Secret may be revealed (by spy, time), thus <ciphertext, plaintext> pair is obtained
  - Great for monoalphabetic ciphers
- Chosen plaintext:
  - Choose text, get encrypted
  - Useful if limited set of messages

Models for Evaluating Security

- Unconditional security (perfect secrecy)
  - Observation of ciphertext provides no information
  - Uncertainty/entropy $H(p)=H(p|c)$
- Complexity-theoretic security
- Provable security
  - As difficult to break as solving well-known and *supposedly* difficult problem
- Computational security
- Ad hoc security

Brute Force Attacks

- Number of encryption/sec: 1 million to 1 billion/sec
- 56-bit key broken in 1 week with 120,000 processors ($6.7m)
- 56-bit key broken in 1 month with 28,000 processors ($1.6m)
- 64-bit key broken in 1 week with 3.1×10^7 processors ($1.7b)
- 128-bit key broken in 1 week with 5.6×10^{26} processors

Types of Cryptography

- Hash functions: no key
- Secret key cryptography: one key
- Public key cryptography: two keys - public, private

Secret Key Cryptography

- Same key is used for encryption and decryption
  - Symmetric cryptography
- Ciphertext approximately the same length as plaintext
- Substitution codes, DES, IDEA
- Message transmission:
  - Agree on key (but how?)
  - Communicate over insecure channel
- Secure storage: *crypt*

Secret Key Cryptography (Cont'd)

- Strong authentication: prove knowledge of key without revealing it:
  - Send challenge *r*, verify the returned encrypted {*r*}
  - Fred can obtain chosen plaintext, cihpertext pairs
    - Challenge should chosen from a large pool
- Integrity check: fixed-length checksum for message
  - Send MIC along with the message

Public Key Cryptography

- Asymmetric cryptography
- Invented/published in 1975
- Two keys: private (*d*), public (*e*)
  - Encryption: public key; Decryption: private key
  - Signing: private key; Verification: public key
- Much slower than secret key cryptography

Public Key Cryptography (Cont'd)

- Data transmission:
  - Alice encrypts m_a using e_B, Bob decrypts to m_a using d_b.
- Storage:
  - Can create a safety copy: using public key of trusted person.
- Authentication:
  - No need to store secrets, only need *public* keys.
  - Secret key cryptography: need to share *secret* key for every person to

Public Key Cryptography (Cont'd)

- Digital signatures
  - Encrypt *hash* $h(m)$ with private key
    - Authorship
    - Integrity
    - Non-repudiation: can't do with secret key cryptography

Hash Algorithms

- Message digests, one-way transformations
- Length of $h(m)$ much shorter then length of m
- Usually fixed lengths: 48-128 bits
- Easy to compute $h(m)$
- Given $h(m)$, no easy way to find m
- Computationally infeasible to find m_1, m_2 s.t. $h(m_1) = h(m_2)$
- Example: $(m+c)^2$, take middle n digits

Hash Algorithms (Cont'd)

- Password hashing
  - Doesn't need to know password to verify it
  - Store $h(p+s)$, s (salt), and compare it with the user-entered p
  - Salt makes dictionary attack less convenient
- Message integrity
  - Agree on a password p
  - Compute $h(p|m)$ and send with m
  - Doesn't require encryption algorithm, so the technology is exportable

Public Key Crypto's Trick

- Consider the Knapsack problem (p482)
- We have a knapsack filled with numbers
- The goal is to select out a series of numbers that adds to some desired number
- How do we do this *efficiently*?

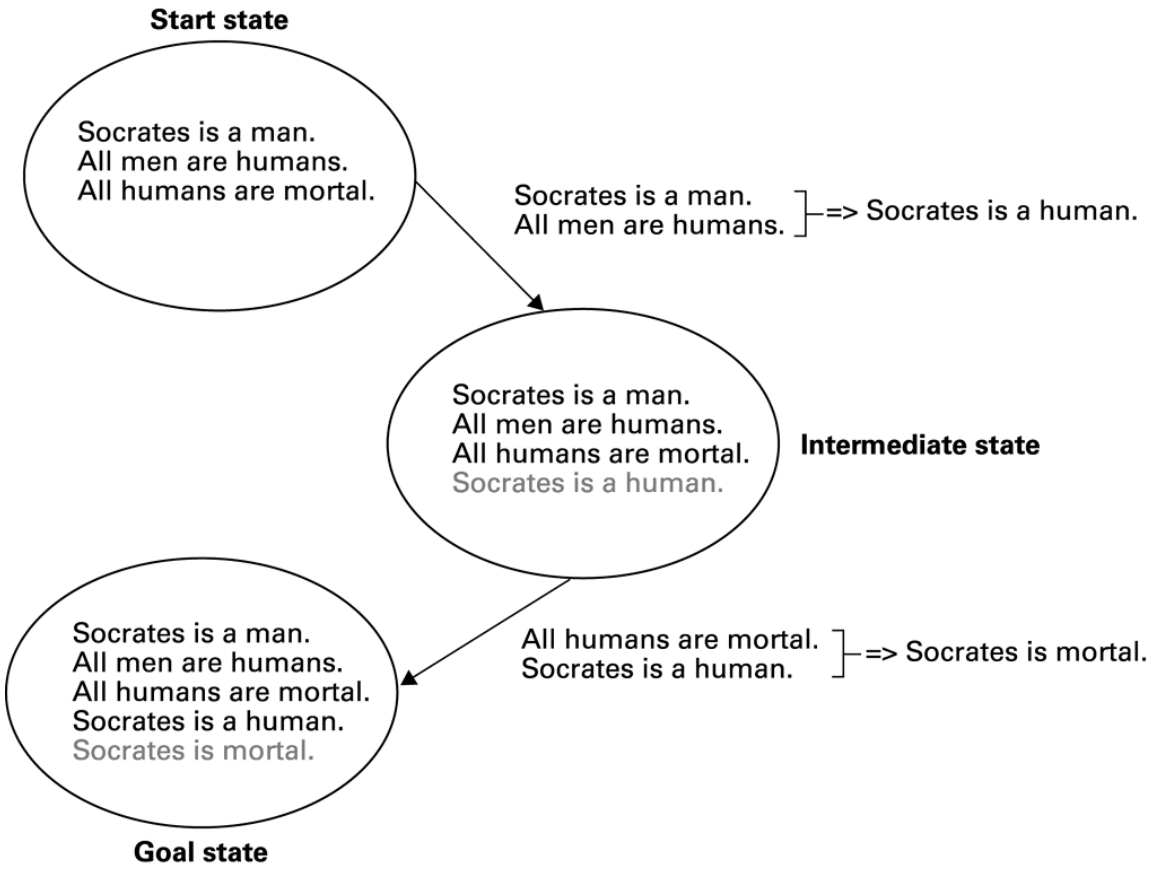
The Trick

- The trick is factoring
- You know ahead of time what certain properties of the number will be. This allows you to reduce the problem to a computable one
- Otherwise, you are dealing with a non-polynomial problem

Artificial Intelligence

- Reasoning (Production Systems
  - Goal is to *derive* a solution given facts and rules
- Searching
  - You are given facts and rules and *search* all possible combinations to find some desired solution (usually minimum/max)
- Heuristics
  - Operate based on guidelines you know to be true about a problem

A Production System



Artificial Intelligence

- Neural Networks
- Genetic Algorithms
- Machine Learning