Introduction to Cryptography

Definition

- process data into unintelligible form, reversibly, without data loss ➠ typically digitally
- usually one-to-one in size ↔ compression
- analog cryptography: voice changers, shredder
- other services:
  - integrity checking: no tampering
  - authentication: not an impostor

Cryptography Caveats

- Cannot *prove* that code is secure ➠ assume until otherwise
  but: can prove (some) systems/protocols secure (assuming secure code)
- Difficult to explain algorithm securely ➠ Cryptographic system = algorithm (published or secret) + secret value (*key*)
- Assume Trudy has algorithm

Computational Difficulty

- algorithm needs to be efficient ➠ may use inefficient for short key
- brute-force cryptanalysis: try all keys until "looks like" plaintext
- any scheme can be broken ➠ depends on \$ = $f(t)$
- longer key ➠ more secure:
  - encryption: $O(N+1)$
  - brute-force cryptanalysis: $O(2^{N+1})$ ➠ twice as hard
- 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 widely used: reverse engineering, social engineering
- commercial: published ➠ wide review, trust
- military: avoid giving enemy good ideas (not just messages)

Trivial Codes

Caesar cipher: substitution cipher: A $\rightarrow$ D, B $\rightarrow$ E

Captain Midnight secret Decoder ring: shift by variable n: IBM ➠ HAL ➠ only 26 possibilities

monoalphabetic cipher: generalization ➠ arbitrary mapping letter to letter ➠ $26! = 4 \cdot 10^{26}$ possibilities ➠ statistical analysis of letter frequencies ➠ larger codebook

Cryptanalysis

Ciphertext only: ➠ exhaustive search until “recognizable plaintext” (unless limited base set) ➠ need enough ciphertext

Known plaintext: secret may be revealed (by spy, time) ➠ pair (ciphertext, plaintext) ➠ great for monoalphabetic ciphers

Chosen plaintext: choose text, get encrypted ➠ useful if limited set of messages or initial strings

Some Large Numbers

Time to next ice age	14,000 yrs
DES 56 bits	$7 \cdot 10^{16}$ keys
probability of MD5 collision	$1/3 \cdot 10^{38}$
Age of planet	10^{9} yrs
Time until sun goes nova	10^{14} yrs
Age of universe	10^{10} yrs
Number of atoms in universe	10^{77}

Brute Force Attacks

- Number of encryptions/sec: 1 million to 1 billion bits/sec
- 1999: 56-bit key broken in 22.5 h with 1,800 chips ($250,000) (245 $\cdot 10^9$ keys/s, see eff.org); helped by distributed.net
- 1995: 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 $3.1 \cdot 10^7$ processors ($1.7B)
- 128-bit key broken in 1 week with $5.6 \cdot 10^{26}$ processors
- Chinese Lottery:
  With machines that test at the rate of a million keys every second, take 64 seconds to break DES with a billion such machines running in parallel.

- DES'osaur:
  With suitable advances in biotechnology, a 10^{14} celled DES'osaur can break DES in 0.2 secs.

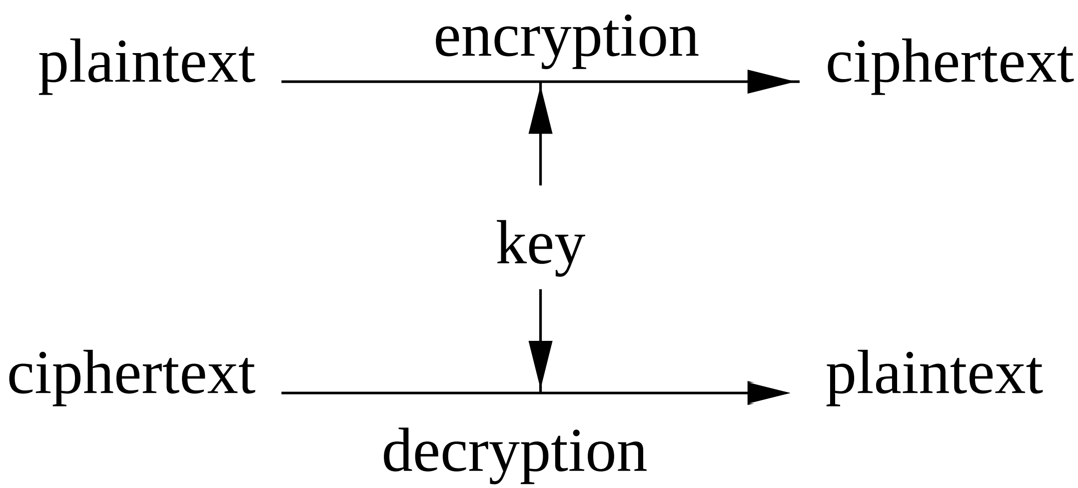
Types of Cryptography

hash functions: no key

secret key cryptography: one key

public key cryptography: two keys – public, private

Secret Key Cryptography



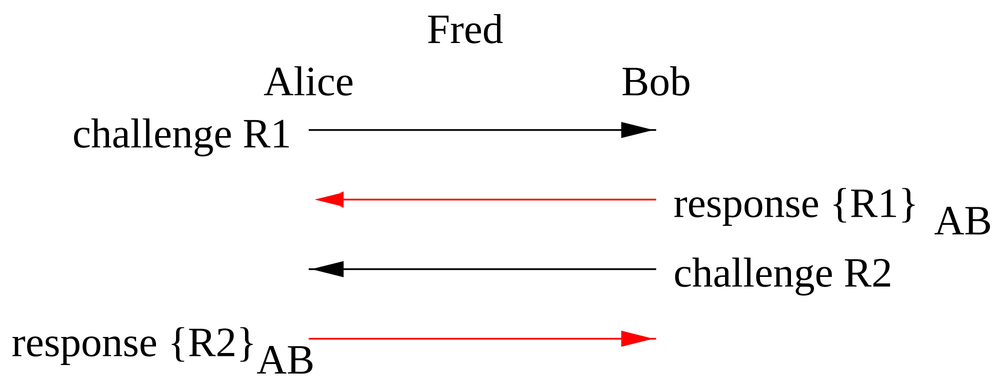
- ciphertext ≈ same length as plaintext
- symmetric cryptography
- substitution codes, DES, IDEA

Message transmission: agree on key (how?), communicate over insecure channel

Secure storage: `crypt` ➠ dangerous, no indication of trouble, no redundancy

Strong Authentication

= prove knowledge of key without revealing it

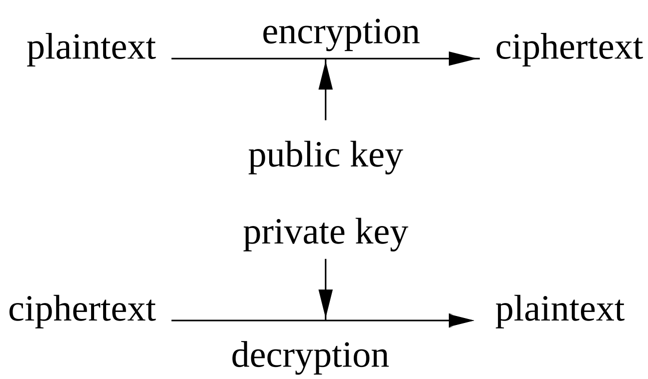


- Fred: obtain chosen plaintext, ciphertext pairs
- not completely secure!

Integrity check = fixed-length checksum for message
CRC not sufficient ➠ easy to pick new message with same CRC
encrypt MIC (*message integrity check*)

Public Key Cryptography

- asymmetric cryptography
- publicly invented in 1975
- two keys: private (d), public (e)
- much slower than secret key cryptography



Public Key Cryptography

Data transmission:

Alice		Bob
encrypt m_A using e_B	$\longrightarrow$	decrypt to m_A using d_B
decrypt to m_B using d_A	$\longleftarrow$	encrypt m_B using e_A

Storage: safety copy: use public key of trusted person

Authentication:

- secret keys: need secret key for every person to communicate with
- secret key: Alice could share key with enemies of Bob
- need to store no secrets:

Alice		Bob
encrypt r using e_B	$\longrightarrow$	decrypt to r using d_B
	$\longleftarrow$	r

Digital Signatures

encrypt *hash* $h(m)$ with private key ➠

- doesn’t reveal text ➠ semi-trusted party
- authorship
- integrity
- non-repudiation: can’t do with secret-key cryptography

Hash Algorithms

- = *message digest, one-way transformation* $h(m)$
- length($h(m)$) $\ll$ length(m)
- usually fixed lengths: 48 – 128 bits
- easy to compute $h(m)$
- given $h(m)$ but not m, no easy way to find m
- computationally infeasible to find m_1, m_2 with $h(m_1) = h(m_2)$
- example: $(m + c)^2$, take middle digits

Password Hashing

- don't need to know password to verify it
- ➠ store $h(p + s)$, s, with *salt* s
- salt makes dictionary attack more difficult
- compare entry with $h(p + s)$
- password file could be world-readable
- Unix: non-standard DES, 4096 salt values

Message Integrity using Hash

- agree on password
- compute $h(m|p)$, send m
- doesn't require encryption algorithm ➠ exportable!
- virus protection, downline load, Java applets: $h(\text{program})$ with *secure* program on write-once storage