

CSE3323: Risks, Security, Reliability, Cryptography

Les Kitchen

19 October 2000

1 Computer Risks

1.1 Overview

- What is risk, in general, and how to quantify it?
- Tour of computer risks, and the issues

1.2 Risk

What is risk?

- Probability of something bad happening?
- Value of what might be lost?
- Expected value of loss/gain: probability times value

1.3 Expected value worksheet

$$\begin{aligned} E(\text{leap}) &= P(\text{death}|\text{leap}) \times -V(\text{life}) \\ &\quad + P(\text{catch}|\text{leap}) \times V(\text{catch}) \\ &= 0.5 \times -10,000 + 0.5 \times 1 \\ &= -4,999.5 \end{aligned}$$

$$\begin{aligned} E(\neg\text{leap}) &= P(\neg\text{catch}|\neg\text{leap}) \times -V(\text{catch}) \\ &\quad + P(\text{catch}|\neg\text{leap}) \times V(\text{catch}) \end{aligned}$$

1.4 More to it?

- Maximizing expected gain is, in a sense, optimal rational long-term strategy
- But not all there is to it...
- People often want to trade-off expectation against variance/predictability

• Insurance:

- Expected small loss (premiums) for greater predictability...
- ...reduced risk of large loss

• Gambling

- Expected (we hope) small loss for less predictability...
- ...small chance of big win

• Nuclear versus Coal Power Plants

- Expected cost of nuclear less than coal (maybe)
- BUT, (small) risk of catastrophe

1.5 Decision Analysis

Ideal

- Specify exclusive and exhaustive sets of actions and states
- Specify *value* of each action/state pair (outcome)
- Specify *probability* of each state
- Compute expected value of each action
- Choose action with maximum expected value

1.6 Leap Example

	Die	Live Catch	Live not Catch	Expect-ation
Leap	-10,000 (P=0.5)	1 (P=0.5)	-1 (P=0)	-4,999.5
Climb	-10,000 (P=0)	1 (P=0.5)	-1 (P=0.5)	0

In this case actions and probabilities of states are dependent

1.7 Decision Analysis in Practice

- In many (most) cases:
 - Probabilities not available
 - States, actions, utilities unclear
- Nevertheless, effort to specify them is often very useful:
 - Focus attention on what's important
 - Sensitivity analysis may reveal degree of precision is OK
- Even though difficult, useful numbers are often available

1.8 Example

What is the dollar value of a human life?

- The question can be answered (at least in some contexts)
- Courts often answer it explicitly
 - Perhaps: Earning power times working life expectancy plus pain and suffering compensation
- Implicit answer (e.g.):
 - Government safety expenditure divided by number of accidental deaths
 - Estimate of value of life to society by government
 - Should be reduced to take account of injuries and other safety costs to society
 - So, such an estimate is an upper bound

1.9 Computer Risks

Complicated issue, but useful categories are:

- Computer system failures
- Computer “Abuse” (Crime?)

1.10 Computer system failures

- hardware
- software
- actual damage, injuries, or death (direct or consequential) down to inconvenience

1.11 Computer “Abuse” (Crime?)

- hackers or crackers?
- physical damage to hardware
- destruction or corruption of data (accidental or malicious)
- theft of data, especially sensitive data
- theft of services (CPU, network, disk)
- use of computers in other crimes
- impersonation (identity theft)
- breach of privacy
- denial of service
- computer trespass (some jurisdictions)
- extortion

1.12 The Titanic Effect

The severity with which a system fails is directly proportional to the intensity of the designer's belief that it cannot.

1.13 Reliability and Availability

Reliability Probability that system will fail over some time interval

- Risk also involves cost of failure

Availability Fraction of time system is available

$$\frac{MTBF}{MTBF + MTTR}$$

MTBF Mean Time Between Failures

MTTR Meant Time To Repair

1.14 Fault Tolerance

Continued operation of system in presence of faults

- Fault tolerance *improves* reliability and availability
 - system does not fail in response to fault, but recovers and continues
 - downtime is zero, or minimal
- Fault tolerance not the same as reliability or availability
 - reliability & availability can be improved by fault elimination (e.g. engineering practices, debugging) without fault tolerance
- Fault tolerance requires redundancy
 - temporal redundancy: re-execution of failed process
 - physical redundancy: duplicate components

1.15 Examples of Physical Redundancy

- Parity bits, CRC, checksums
 - fault detection and correction
- Triple Modular Redundancy (TMR, von Neuman, 1956)
 - e.g., military and space applications
 - three units vote on each action
 - if any deviation, majority wins, loser disabled
 - if subsequent failure, system fails
 -

$$R_c [R_m^3 + 3R_m^2(1 - R_m)]$$

- intended for hardware
- Hot Backup
 - e.g., IBM's airline reservation systems, disaster recovery systems
 - systems operate in parallel

- fault detection built in
- on detected failure, backup takes over

- All involve extra cost: especially TMR and hot backup

1.16 Pathogens

- Virus
 - Intentionally destructive, self-propagating program
- Worm
 - Self-propagating program (particularly on network)
 - Not intentionally destructive...
 - ... but may go out of control (e.g. famous Internet Worm of 1988) if only accidental denial of service
- Trojan Horse
 - A malicious program that gets run by pretending to be something legitimate
 - e.g. executable called `ls` on `/tmp`
 - e.g. fake login-screen to capture passwords
 - most viruses get started this way
- Time Bomb
 - Code to cause some malicious operation
 - set to go off at some particular time
 - or on some particular condition
 - e.g. when programmer is no longer on the payroll
- Backdoor
 - Code in otherwise legitimate software to allow some secret unauthorized access
 - e.g. special key sequence allows programmer to change their bank balance
- NOTE, these schemes can be quite intricate in execution

1.17 Exploits

Main ones:

- Trojan horses
- Buffer over-runs
 - never use `scanf()`'s `%s` format or `gets()`!
- Password guessing/cracking
- IP spoofing
- Opportunism: unattended terminals or comms lines; unintentionally revealed sensitive data

1.18 Improving Computer Security

- Organizational
 - foster security awareness
- Procedural
 - establish procedures and responsibilities
 - * prevention
 - * monitoring/detection
 - * response
 - * recovery
 - * forensics
 - * reporting
 - * disposal
 - * ...
- Technical, e.g.
 - physical security of machines and communications channels
 - safe backups of data and tools
 - cryptographic (e.g. MD5) file checksums
 - upgrading—security bug fixes
 - limitation of services
 - authorized challenges: e.g. Satan, cracklib
 - firewalls
 - encryption

- passwords and other verification mechanisms (e.g. biometric, challenge and response)
- inspection of computer and network log files
- ...

1.19 Cryptography

- plain text, key, cypher text
- cryptanalysis (code-breaking) relies on redundancy
 - e.g. WW2 German Enigma Machine
 - shorter the key relative to corpus of cypher text, better chance of breaking
 - compression with encryption
- “One Time Pad”
 - Problem: transmission of key
- *Strong Cryptography*, e.g., RSA
 - in principle, computationally intractable to break, with respect to current mathematical knowledge
 - in practice, often pushed by computer power
- Public-Key Cryptography
- cryptographic protocols
- e-cash—verifiability and anonymity
- steganography
- traffic analysis
- PGP
- Are cyphers armaments?
- civil liberty issues
 - general use of encryption
 - government access
 - * legitimate law-enforcement versus snooping
 -

1.20 Public-Key Cryptography

- Cryptographic system with two keys: a message encrypted with one key can only be decrypted with the other key, and vice-versa.
- One is *public key*; other is *private key*
- A wants to send secret message to B over insecure communication channel:
 1. A encrypts message with B's public key
 2. A sends encrypted message to B
 3. B decrypts message with B's private key
 4. only B could decrypt the message
- How does B know the message actually came from A, and not an impostor?
- Digital signatures
 1. A encrypts message with B's public key (as before)
 2. A adds to the message a signature, then encrypts the whole lot (encrypted message plus signature) with A's private key
 3. A sends this message to B
 4. B decrypts the outer message with A's public key, and by seeing the signature, knows the message must have come from A
 5. B then decrypts the inner message with B's private key (as before)
 6. Third party C, could intercept message, and unpack it, but couldn't decrypt inner message to tamper with it, nor could they put message back together with a different signature
 7. signatures can also be applied to plain-text message
- How does A know that this actually is B's public key, and not of some impostor?
- Trust, and key certificates