

Lecture 6

October 4, 2021

Clark-Wilson Integrity Model

- Integrity defined by a set of constraints
 - Data in a *consistent* or valid state when it satisfies these
- Example: Bank
 - D today's deposits, W withdrawals, YB yesterday's balance, TB today's balance
 - Integrity constraint: $D + YB - W$
- *Well-formed transaction* move system from one consistent state to another
- Issue: who examines, certifies transactions done correctly?

Entities

- CDIs: constrained data items
 - Data subject to integrity controls
- UDIs: unconstrained data items
 - Data not subject to integrity controls
- IVPs: integrity verification procedures
 - Procedures that test the CDIs conform to the integrity constraints
- TPs: transaction procedures
 - Procedures that take the system from one valid state to another

Certification Rules 1 and 2

- CR1 When any IVP is run, it must ensure all CDIs are in a valid state
- CR2 For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state
- Defines relation *certified* that associates a set of CDIs with a particular TP
 - Example: TP balance, CDIs accounts, in bank example

Enforcement Rules 1 and 2

- ER1 The system must maintain the certified relations and must ensure that only TPs certified to run on a CDI manipulate that CDI.
- ER2 The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. The TP cannot access that CDI on behalf of a user not associated with that TP and CDI.
- System must maintain, enforce certified relation
 - System must also restrict access based on user ID (*allowed* relation)

Users and Rules

CR3 The allowed relations must meet the requirements imposed by the principle of separation of duty.

ER3 The system must authenticate each user attempting to execute a TP

- Type of authentication undefined, and depends on the instantiation
- Authentication *not* required before use of the system, but *is* required before manipulation of CDIs (requires using TPs)

Logging

CR4 All TPs must append enough information to reconstruct the operation to an append-only CDI.

- This CDI is the log
- Auditor needs to be able to determine what happened during reviews of transactions

Handling Untrusted Input

- CR5 Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.
- In bank, numbers entered at keyboard are UDIs, so cannot be input to TPs. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI

Separation of Duty In Model

- ER4 Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.
- Enforces separation of duty with respect to certified and allowed relations

Comparison With Requirements

1. Users can't certify TPs, so CR5 and ER4 enforce this
2. Procedural, so model doesn't directly cover it; but special process corresponds to using TP
 - No technical controls can prevent programmer from developing program on production system; usual control is to delete software tools
3. TP does the installation, trusted personnel do certification

Comparison With Requirements

4. CR4 provides logging; ER3 authenticates trusted personnel doing installation; CR5, ER4 control installation procedure
 - New program UDI before certification, CDI (and TP) after
5. Log is CDI, so appropriate TP can provide managers, auditors access
 - Access to state handled similarly

Comparison to Biba

- Biba
 - No notion of certification rules; trusted subjects ensure actions obey rules
 - Untrusted data examined before being made trusted
- Clark-Wilson
 - Explicit requirements that *actions* must meet
 - Trusted entity must certify *method* to upgrade untrusted data (and not certify the data itself)

UNIX Implementation

- Considered “allowed” relation

(user, TP, { CDI set })

- Each TP is owned by a different user
 - These “users” are actually locked accounts, so no real users can log into them; but this provides each TP a unique UID for controlling access rights
 - TP is setuid to that user
- Each TP’s group contains set of users authorized to execute TP
- Each TP is executable by group, not by world

CDI Arrangement

- CDIs owned by *root* or some other unique user
 - Again, no logins to that user's account allowed
- CDI's group contains users of TPs allowed to manipulate CDI
- Now each TP can manipulate CDIs for single user

Examples

- Access to CDI constrained by user
 - In “allowed” triple, *TP* can be any TP
 - Put CDIs in a group containing all users authorized to modify CDI
- Access to CDI constrained by TP
 - In “allowed” triple, *user* can be any user
 - CDIs allow access to the owner, the user owning the TP
 - Make the TP world executable

Problems

- 2 different users cannot use same copy of TP to access 2 different CDIs
 - Need 2 separate copies of TP (one for each user and CDI set)
- TPs are setuid programs
 - As these change privileges, want to minimize their number
- *root* can assume identity of users owning TPs, and so cannot be separated from certifiers
 - No way to overcome this without changing nature of *root*

Cryptosystem

- Quintuple $(\mathcal{E}, \mathcal{D}, \mathcal{M}, \mathcal{K}, \mathcal{C})$
 - \mathcal{M} set of plaintexts
 - \mathcal{K} set of keys
 - \mathcal{C} set of ciphertexts
 - \mathcal{E} set of encryption functions $e: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$
 - \mathcal{D} set of decryption functions $d: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$

Example

- Example: Cæsar cipher
 - $\mathcal{M} = \{ \text{sequences of letters} \}$
 - $\mathcal{K} = \{ i \mid i \text{ is an integer and } 0 \leq i \leq 25 \}$
 - $\mathcal{E} = \{ E_k \mid k \in \mathcal{K} \text{ and for all letters } m, E_k(m) = (m + k) \bmod 26 \}$
 - $\mathcal{D} = \{ D_k \mid k \in \mathcal{K} \text{ and for all letters } c, D_k(c) = (26 + c - k) \bmod 26 \}$
 - $C = \mathcal{M}$

Attacks

- Opponent whose goal is to break cryptosystem is the *adversary*
 - Assume adversary knows algorithm used, but not key
- Three types of attacks:
 - *ciphertext only*: adversary has only ciphertext; goal is to find plaintext, possibly key
 - *known plaintext*: adversary has ciphertext, corresponding plaintext; goal is to find key
 - *chosen plaintext*: adversary may supply plaintexts and obtain corresponding ciphertext; goal is to find key

Basis for Attacks

- Mathematical attacks
 - Based on analysis of underlying mathematics
- Statistical attacks
 - Make assumptions about the distribution of letters, pairs of letters (digrams), triplets of letters (trigrams), *etc.*
 - Called *models of the language*
 - Examine ciphertext, correlate properties with the assumptions.

Symmetric Cryptography

- Sender, receiver share common key
 - Keys may be the same, or trivial to derive from one another
 - Sometimes called *secret key cryptography*
- Two basic types
 - Transposition ciphers
 - Substitution ciphers
 - Combinations are called *product ciphers*

Transposition Cipher

- Rearrange letters in plaintext to produce ciphertext

- Example (Rail-Fence Cipher)

- Plaintext is HELLO WORLD

- Rearrange as

HLOOL

ELWRD

- Ciphertext is HLOOL ELWRD

Attacking the Cipher

- Anagramming
 - If 1-gram frequencies match English frequencies, but other n -gram frequencies do not, probably transposition
 - Rearrange letters to form n -grams with highest frequencies

Example

- Ciphertext: HLOOLELWRD
- Frequencies of 2-grams beginning with H
 - HE 0.0305
 - HO 0.0043
 - HL, HW, HR, HD < 0.0010
- Frequencies of 2-grams ending in H
 - WH 0.0026
 - EH, LH, OH, RH, DH ≤ 0.0002
- Implies E follows H

Example

- Arrange so the H and E are adjacent

HE

LL

OW

OR

LD

- Read across, then down, to get original plaintext

Substitution Ciphers

- Change characters in plaintext to produce ciphertext
- Example (Caesar cipher)
 - Plaintext is HELLO WORLD
 - Change each letter to the third letter following it (X goes to A, Y to B, Z to C)
 - Key is 3, usually written as letter 'D'
 - Ciphertext is KHOOR ZRUOG

Attacking the Cipher

- Exhaustive search
 - If the key space is small enough, try all possible keys until you find the right one
 - Caesar cipher has 26 possible keys
- Statistical analysis
 - Compare to 1-gram model of English

Statistical Attack

- Compute frequency of each letter in ciphertext:

G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

- Apply 1-gram model of English
 - Frequency of characters (1-grams) in English is on next slide

Character Frequencies

a	0.07984	h	0.06384	n	0.06876	t	0.09058
b	0.01511	i	0.07000	o	0.07691	u	0.02844
c	0.02504	j	0.00131	p	0.01741	v	0.01056
d	0.04260	k	0.00741	q	0.00107	w	0.02304
e	0.12452	l	0.03961	r	0.05912	x	0.00159
f	0.02262	m	0.02629	s	0.06333	y	0.02028
g	0.02013					z	0.00057

Statistical Analysis

- $f(c)$ frequency of character c in ciphertext
- $\varphi(i)$ correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is i
 - $\varphi(i) = \sum_{0 \leq c \leq 25} f(c)p(c - i)$ so here,
$$\varphi(i) = 0.1 p(6 - i) + 0.1 p(7 - i) + 0.1 p(10 - i) + 0.3 p(14 - i) + 0.2 p(17 - i) + 0.1 p(20 - i) + 0.1 p(25 - i)$$
 - $p(x)$ is frequency of character x in English

Correlation: $\varphi(i)$ for $0 \leq i \leq 25$

i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$
0	0.0469	7	0.0461	13	0.0505	19	0.0312
1	0.0393	8	0.0194	14	0.0561	20	0.0287
2	0.0396	9	0.0286	15	0.0215	21	0.0526
3	0.0586	10	0.0631	16	0.0306	22	0.0398
4	0.0259	11	0.0280	17	0.0386	23	0.0338
5	0.0165	12	0.0318	18	0.0317	24	0.0320
6	0.0676					25	0.0443

The Result

- Most probable keys, based on φ :
 - $i = 6, \varphi(i) = 0.0676$
 - plaintext EBIIIL TLOLA
 - $i = 10, \varphi(i) = 0.0631$
 - plaintext AXEEH PHKEW
 - $i = 14, \varphi(i) = 0.0561$
 - plaintext WTAAD LDGAS
 - $i = 3, \varphi(i) = 0.0586$
 - plaintext HELLO WORLD
- Only English phrase is for $i = 3$
 - That's the key (3 or 'D')

Caesar's Problem

- Key is too short
 - Can be found by exhaustive search
 - Statistical frequencies not concealed well
 - They look too much like regular English letters
- So make it longer
 - Multiple letters in key
 - Idea is to smooth the statistical frequencies to make cryptanalysis harder

Vigènere Cipher

- Like Caesar cipher, but use a phrase
 - So it's effectively multiple Caesar ciphers
- Example
 - Message A LIMERICK PACKS LAUGHS ANATOMICAL
 - Key BENCH
 - Encipher using Caesar cipher for each letter:

key	BENCHBENCHBENCHBENCHBENCHBENCH
plain	ALIMERICKPACKSLAUGHSANATOMICAL
cipher	BPVOLSMPMWBGXUSBYTJZBRNVVNMPCS

Relevant Parts of Tableau

<i>A</i>	<i>B</i>	<i>C</i>	<i>E</i>	<i>H</i>	<i>N</i>
<i>C</i>	<i>B</i>	<i>C</i>	<i>E</i>	<i>H</i>	<i>N</i>
<i>E</i>	<i>D</i>	<i>E</i>	<i>G</i>	<i>J</i>	<i>P</i>
<i>G</i>	<i>F</i>	<i>G</i>	<i>I</i>	<i>L</i>	<i>R</i>
<i>H</i>	<i>H</i>	<i>I</i>	<i>K</i>	<i>N</i>	<i>T</i>
<i>I</i>	<i>I</i>	<i>J</i>	<i>L</i>	<i>O</i>	<i>U</i>
<i>K</i>	<i>J</i>	<i>K</i>	<i>M</i>	<i>P</i>	<i>V</i>
<i>L</i>	<i>L</i>	<i>M</i>	<i>O</i>	<i>R</i>	<i>X</i>
<i>M</i>	<i>M</i>	<i>N</i>	<i>P</i>	<i>S</i>	<i>Y</i>
<i>N</i>	<i>N</i>	<i>O</i>	<i>Q</i>	<i>T</i>	<i>Z</i>
<i>O</i>	<i>O</i>	<i>P</i>	<i>R</i>	<i>U</i>	<i>A</i>
<i>P</i>	<i>P</i>	<i>Q</i>	<i>S</i>	<i>V</i>	<i>B</i>
<i>R</i>	<i>Q</i>	<i>R</i>	<i>T</i>	<i>W</i>	<i>C</i>
<i>S</i>	<i>S</i>	<i>T</i>	<i>V</i>	<i>Y</i>	<i>E</i>
<i>T</i>	<i>T</i>	<i>U</i>	<i>W</i>	<i>Z</i>	<i>F</i>
<i>U</i>	<i>U</i>	<i>V</i>	<i>X</i>	<i>A</i>	<i>G</i>
	<i>V</i>	<i>W</i>	<i>Y</i>	<i>B</i>	<i>H</i>

- Tableau shown has relevant rows, columns only
 - Columns correspond to letters from the key
 - Rows correspond to letters from the message
- Example encipherments:
 - key B, letter R: follow B column down to R row (giving “S”)
 - Key H, letter L: follow H column down to L row (giving “S”)

Useful Terms

- *period*: length of key
 - In earlier example, period is 3
- *tableau*: table used to encipher and decipher
 - Vigenere cipher has key letters on top, plaintext letters on the left
- *polyalphabetic*: the key has several different letters
 - Caesar cipher is monoalphabetic

Attacking the Cipher

- Approach
 - Establish period; call it n
 - Break message into n parts, each part being enciphered using the same key letter
 - Solve each part; you can leverage one part from another
- We will show each step

The Target Cipher

- We want to break this cipher:

ADQYS MIUSB OXKKT MIBHK IZOOO EQOOG IFBAG KAUMF
VVTAA CIDTW MOCIO EQOOG BMBFV ZGGWP CIEKQ HSNEW
VECNE DLAAV RWKXS VNSVP HCEUT QOIOF MEGJS WTPCH
AJMOC HIUIX

Establish Period

- Kasiski: *repetitions in the ciphertext occur when characters of the key appear over the same characters in the plaintext*
- Example:

```
key      VIGVIGVIGVIGVIGV
plain    THEBOYHASTHEBALL
cipher   OPKWECIYOPKWIRG
```

Note the key and plaintext line up over the repetitions (underlined). As distance between repetitions is 9, the period is a factor of 9 (that is, 1, 3, or 9)

Repetitions in Example

Letters	Start	End	Gap Length	Gap Length Factors
OEQOOG	24	54	30	2, 3, 5
MOC	50	122	72	2, 2, 2, 3, 3

Estimate of Period

- OEQOOG is probably not a coincidence
 - It's too long for that
 - Period may be 1, 2, 3, 5, 6, 10, 15, or 30
- MOC is also probably not a coincidence
 - Period may be 1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, or 72
- Period of 2 or 3 is probably too short (but maybe not)
- Begin with period of 6

Check on Period

- Index of coincidence is probability that two randomly chosen letters from ciphertext will be the same
- Tabulated for different periods:

1 0.0660

2 0.0520

3 0.0473

6 0.0427

Compute IC for an Alphabet

- $IC = [n(n-1)]^{-1} \sum_{0 \leq i \leq 25} [F_i(F_i-1)]$
 - where n is length of ciphertext and F_i the number of times character i occurs in ciphertext
- For the given ciphertext, $IC = 0.0433$
 - Indicates a key of length 5 or 6
 - A statistical measure, so it can be in error, but it agrees with the previous estimate (which was 6)

Splitting Into Alphabets

alphabet 1: AIKHOIATTOBGEEERNEOSAI

alphabet 2: DUKKEFUAWEMGKWDWSUFWJU

alphabet 3: QSTIQBMAMQBWQVLKVTMTMI

alphabet 4: YBMZOAFCOOFPHEAXPQEPOX

alphabet 5: SOIOOGVICOVCSVASHOGCC

alphabet 6: MXBOGKVDIGZINNVVCIJHH

- ICs (#1, 0.0692; #2, 0.0779; #3, 0.0779; #4, 0.0562; #5, 0.1238; #6, 0.0429) indicate all alphabets have period 1, except #4 (between 1 and 2) and #6 (between 5 and 6); assume statistical variance

Frequency Examination

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	3	1	0	0	4	0	1	1	3	0	1	0	0	1	3	0	0	1	1	2	0	0	0	0	0	0
2	1	0	0	2	2	2	1	0	0	1	3	0	1	0	0	0	0	0	1	0	4	0	4	0	0	0
3	1	2	0	0	0	0	0	0	2	0	1	1	4	0	0	0	4	0	1	3	0	2	1	0	0	0
4	2	1	1	0	2	2	0	1	0	0	0	0	1	0	4	3	1	0	0	0	0	0	0	2	1	1
5	1	0	5	0	0	0	2	1	2	0	0	0	0	0	5	0	0	0	3	0	0	2	0	0	0	0
6	0	1	1	1	0	0	2	2	3	1	1	0	1	2	1	0	0	0	0	0	0	3	0	1	0	1
	H	M	M	M	H	M	M	H	H	M	M	M	M	H	H	M	L	H	H	H	M	L	L	L	L	L

The last row has general letter frequencies (H high, M medium, L low)

Begin Decryption

- First matches characteristics of unshifted alphabet
- Third matches if I shifted to A
- Sixth matches if V shifted to A
- Substitute into ciphertext (bold are substitutions)

ADIYS RIUKB OCKKL MIGHK AZOTO EIOOL IFTAG
PAUEF VATAS CIITW EOCNO EIOOL BMTFV EGGOP
CNEKI HSSEW NECSE DDAAA RWCXS ANSNP HHEUL
QONOF EEGOS WLPCM AJEOC MIUAX

Look For Clues

- **AJE** in last line suggests “are”, meaning second alphabet maps A into S:

ALIYS RICKB OCKSL MIGHS AZOTO MIOOL INTAG
PACEF VATIS CIITE EOCNO MIOOL BUTFV EGOOP
CNESI HSSEE NECSE LDAAA RECXS ANANP HHECL
QONON EEGOS ELPCM AREOC MICAX

Next Alphabet

- **MICAX** in last line suggests “mical” (a common ending for an adjective), meaning fourth alphabet maps O into A:

ALIMS RICKP OCKSL AIGHS ANOTO MICOL INTOG
PACET VATIS QIITE ECCNO MICOL BUTTV EGOOD
CNESI VSSEE NSCSE LDOAA RECLS ANAND HHECL
EONON ESGOS ELDCM ARECC MICAL

Got It!

- **QI** means that U maps into I, as Q is always followed by U:

ALIME RICKP ACKSL AUGHS ANATO MICAL INTOS
PACET HATIS QUITE ECONO MICAL BUTTH EGOOD
ONESI VESEE NSOSE LDOMA RECLE ANAND THECL
EANON ESSOS ELDOM ARECO MICAL

One-Time Pad

- A Vigenère cipher with a random key at least as long as the message
 - Provably unbreakable
 - Why? Look at ciphertext DXQR. Equally likely to correspond to plaintext DOIT (key AJIY) and to plaintext DONT (key AJDY) and any other 4 letters
 - Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
 - Approximations, such as using pseudorandom number generators to generate keys, are *not* random

Overview of the DES

- A block cipher:
 - encrypts blocks of 64 bits using a 64 bit key
 - outputs 64 bits of ciphertext
- A product cipher
 - basic unit is the bit
 - performs both substitution and transposition (permutation) on the bits
- Cipher consists of 16 rounds (iterations) each with a 48 bit round key generated from the user-supplied key

Structure of the DES

- Input is first permuted, then split into left half (L) and right half (R), each 32 bits
- Round begins; R and round key run through function f , then xor'ed with L
 - f expands R to 48 bits, xors with round key, and then each 6 bits of this are run through S-boxes (substitution boxes), each of which gives 4 bits of output
 - Those 32 bits are permuted and this is the output of f
- R and L swapped, ending the round
 - Swapping does not occur in the last round
- After last round, L and R combined, permuted, forming DES output

Controversy

- Considered too weak
 - Diffie, Hellman said in a few years technology would allow DES to be broken in days
 - Design using 1999 technology published
- Design decisions not public
 - S-boxes may have backdoors

Undesirable Properties

- 4 weak keys
 - They are their own inverses
- 12 semi-weak keys
 - Each has another semi-weak key as inverse
- Complementation property
 - $DES_k(m) = c \Rightarrow DES_k(\bar{m}) = \bar{c}$
- S-boxes exhibit irregular properties
 - Distribution of odd, even numbers non-random
 - Outputs of fourth box depends on input to third box

Differential Cryptanalysis

- A chosen ciphertext attack
 - Requires 2^{47} plaintext, ciphertext pairs
- Revealed several properties
 - Small changes in S-boxes reduced the number of pairs needed
 - Making every bit of the round keys independent did not impede attack
- Linear cryptanalysis improves result
 - Requires 2^{43} plaintext, ciphertext pairs

DES Modes

- Electronic Code Book Mode (ECB)
 - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
 - Xor each block with previous ciphertext block
 - Requires an initialization vector for the first one
- Encrypt-Decrypt-Encrypt (2 keys: k, k')
 - $c = \text{DES}_k(\text{DES}_{k'}^{-1}(\text{DES}_k(m)))$
- Triple DES(3 keys: k, k', k'')
 - $c = \text{DES}_k(\text{DES}_{k'}(\text{DES}_{k''}(m)))$

Current Status of DES

- Design for computer system, associated software that could break any DES-enciphered message in a few days published in 1998
- Several challenges to break DES messages solved using distributed computing
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES
 - Designed to withstand attacks that were successful on DES
- DES officially withdrawn in 2005

Advanced Encryption Standard

- Competition announces in 1997 to select successor to DES
 - Successor needed to be available for use without payment (no royalties, etc.)
 - Successor must encipher 128-bit blocks with keys of lengths 128, 192, and 256
- 3 workshops in which proposed successors were presented, analyzed
- Rijndael selected as successor to DES, called the Advanced Encryption Standard (AES)
 - Other finalists were Twofish, Serpent, RC6, MARS

Overview of the AES

- A block cipher:
 - encrypts blocks of 128 bits using a 128, 192, or 256 bit key
 - outputs 128 bits of ciphertext
- A product cipher
 - basic unit is the bit
 - performs both substitution and transposition (permutation) on the bits
- Cipher consists of rounds (iterations) each with a round key generated from the user-supplied key
 - If 128 bit key, then 10 rounds
 - If 192 bit key, then 12 rounds
 - If 256 bit key, then 14 rounds

Structure of the AES: Encryption

- Input placed into a state array, which is then combined with zeroth round key
 - Treat state array as a 4x4 matrix, each entry being a byte
- Round begins; new values substituted for each byte of the state array
- Rows then cyclically shifted
- Each column independently altered
 - Not done in last round
- Result xor'ed with round key
- After last round, state array is the encrypted input

Structure of the AES: Decryption

- Round key schedule reversed
- Input placed into a state array, which is then combined with zeroth round key (of reversed schedule)
- Round begins; rows cyclically shifted, then new values substituted for each byte of the state array
 - Inverse rotation, substitution of encryption
- Result xor'ed with round key (of reversed schedule)
- Each column independently altered
 - Inverse of encryption; this is not done in last round
- After last round, state array is the decrypted input

Analysis of AES

- Designed to withstand attacks that the DES is vulnerable to
- All details of design made public, unlike with the DES
 - In particular, those of the substitutions (S-boxes) were described
- After 2 successive rounds, every bit in the state array depends on every bit in the state array 2 rounds ago
- No weak, semi-weak keys

AES Modes

- DES modes also work with AES
- EDE and “Triple-AES” not used
 - Extended block size makes this unnecessary
- New counter mode CTR added