Lecture 9, April 17, 2026

Role-Based Access Control

- Access depends on function, not identity
  - Example:
    - Allison, bookkeeper for Math Dept, has access to financial records.
    - She leaves.
    - Betty hired as the new bookkeeper, so she now has access to those records
  - The role of “bookkeeper” dictates access, not the identity of the individual.

Definitions

- Role *r*: collection of job functions
  - *trans*(*r*): set of authorized transactions for *r*
- Active role of subject *s*: role *s* is currently in
  - *actr*(*s*)
- Authorized roles of a subject *s*: set of roles *s* is authorized to assume
  - *authr*(*s*)
- *canexec*(*s*, *t*) iff subject *s* can execute transaction *t* at current time

Axioms

Let S be the set of subjects and T the set of transactions.

- *Rule of role assignment*: $(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow actr(s) \neq \varnothing]$.
  - If s can execute a transaction, it has a role
  - This ties transactions to roles
- *Rule of role authorization*: $(\forall s \in S)\ [actr(s) \subseteq authr(s)]$.
  - Subject must be authorized to assume an active role (otherwise, any subject could assume any role)

Axiom

- *Rule of transaction authorization*:

$$(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow t \in trans(actr(s))].$$

  - If a subject *s* can execute a transaction, then the transaction is an authorized one for the role *s* has assumed

Containment of Roles

- Trainer can do all transactions that trainee can do (and then some). This means role *r* contains role *r′* (*r* > *r′*). So:

  $(\forall s \in S)[\ r' \in authr(s) \wedge r > r' \rightarrow r \in authr(s)\ ]$

Separation of Duty

- Let *r* be a role, and let *s* be a subject such that $r \in auth(s)$. Then the predicate *meauth*(*r*) (for mutually exclusive authorizations) is the set of roles that *s* cannot assume because of the separation of duty requirement.
- Separation of duty:

$$(\forall r_1, r_2 \in R)\ [\ r_2 \in meauth(r_1) \rightarrow [\ (\forall s \in S)\ [\ r_1 \in authr(s) \rightarrow r_2 \notin authr(s)\ ]\ ]\ ]$$

Role Engineering

- *Role engineering*: defining roles and determining needed permissions
- Often used when two organizations using RBAC merge
  - Roles in one organization rarely overlap with roles in other
  - Job functions often do overlap
- *Role mining*: analyzing existing roles, permission assignments to determine optimal assignment of permissions to roles
  - *NP*-complete, but in practice optimal solutions can be approximated or produced

Cryptosystem

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

Example

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

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
  - $\varphi(i) = \Sigma_{0 \le c \le 25} f(c)p(c - i)$ 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 $0 \le i \le 25$

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 `EBIIL 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`

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