Introduction to Computer Security

Lecture 5
RBAC,
Policy Composition
Basic Cryptography

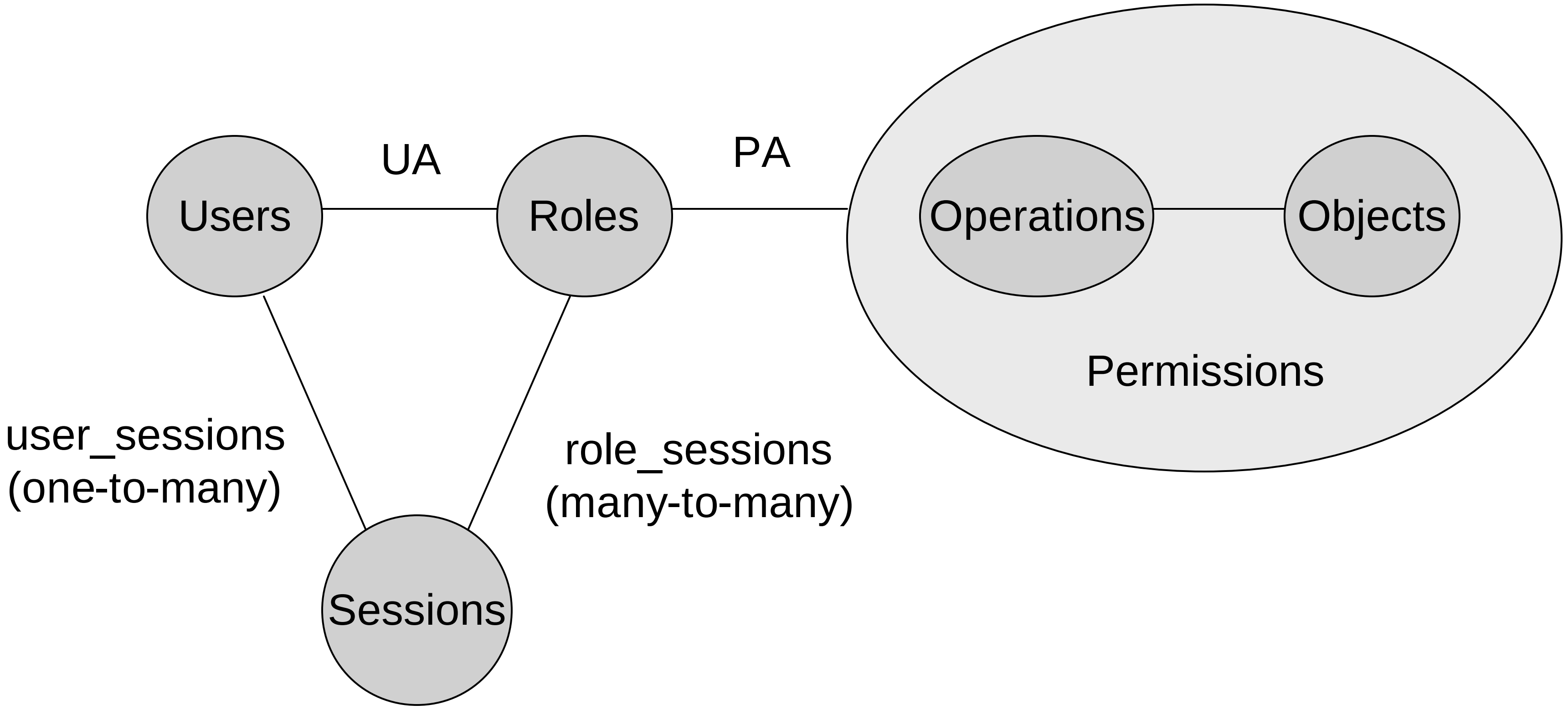
September 25, 2003

Announcements

- TA: Rachata Peechavanish
- Office hours: Tuesdays, 2pm-4pm
- Email: rapst49@pitt.edu
- Place: 2nd Floor Lounge

- HW2: Due tomorrow
  - Drop in Room 719, or
  - Email me by that time

RBAC (NIST Standard)

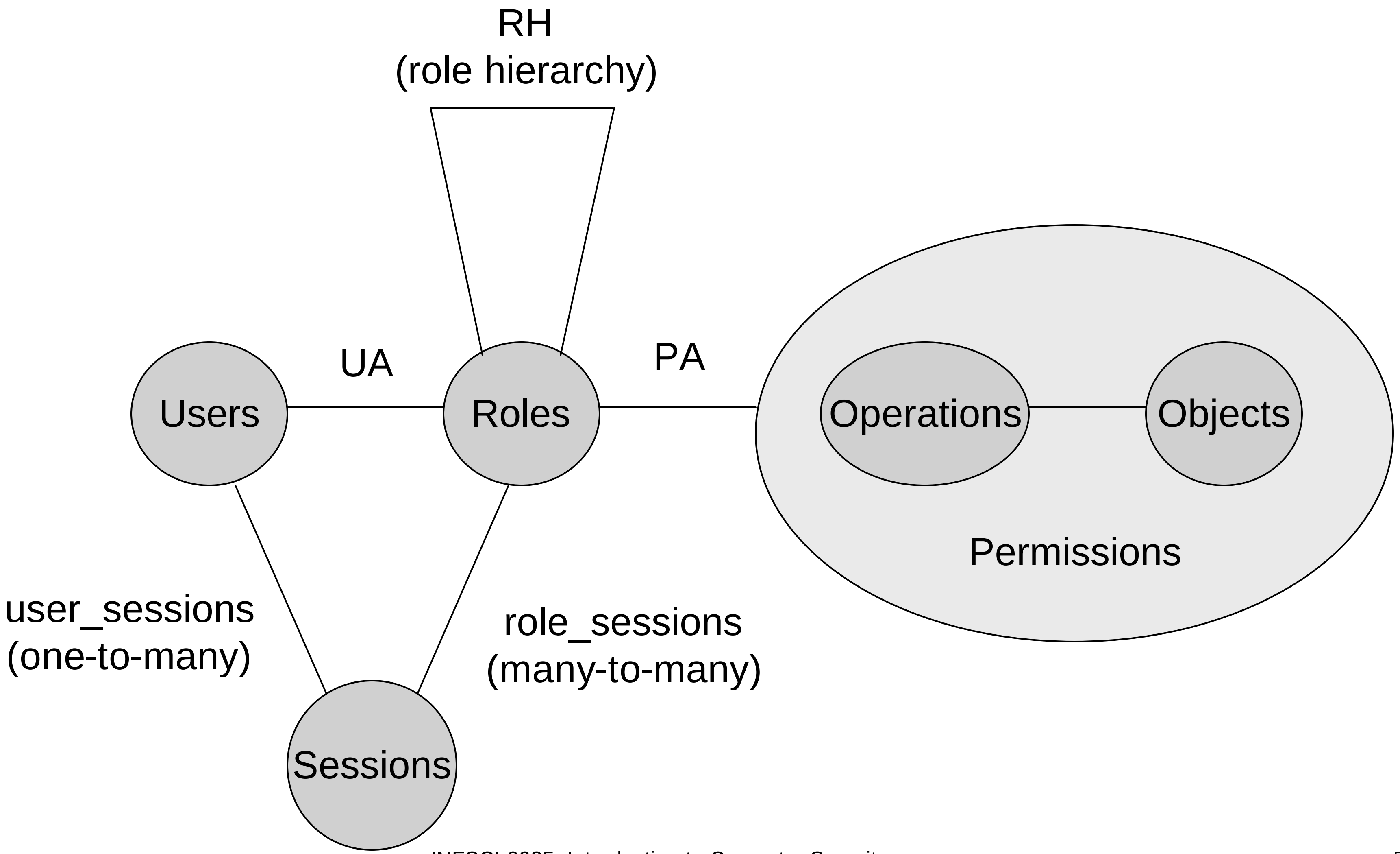


An important difference from classical models is that Subject in other models corresponds to a Session in RBAC

Core RBAC (relations)

- Permissions = $2^{\text{Operations x Objects}}$
- UA ? Users x Roles
- PA ? Permissions x Roles
- *assigned_users*: Roles $\rightarrow$ 2^{Users}
- *assigned_permissions*: Roles $\rightarrow$ $2^{\text{Permissions}}$
- *Op*(p): set of operations associated with permission p
- *Ob*(p): set of objects associated with permission p
- *user_sessions*: Users $\rightarrow$ 2^{Sessions}
- *session_user*: Sessions $\rightarrow$ Users
- *session_roles*: Sessions $\rightarrow$ 2^{Roles}
    - *session_roles*(*s*) = {*r* | (session_user(*s*), *r*) $\in$ UA)}
- *avail_session_perms*: Sessions $\rightarrow$ $2^{\text{Permissions}}$

RBAC with General Role Hierarchy



RBAC with General Role Hierarchy

- *authorized_users*: Roles$\rightarrow 2^{\text{Users}}$

  $authorized\_users(r) = \{u \mid r' = r \,\&(r', u) \in UA\}$

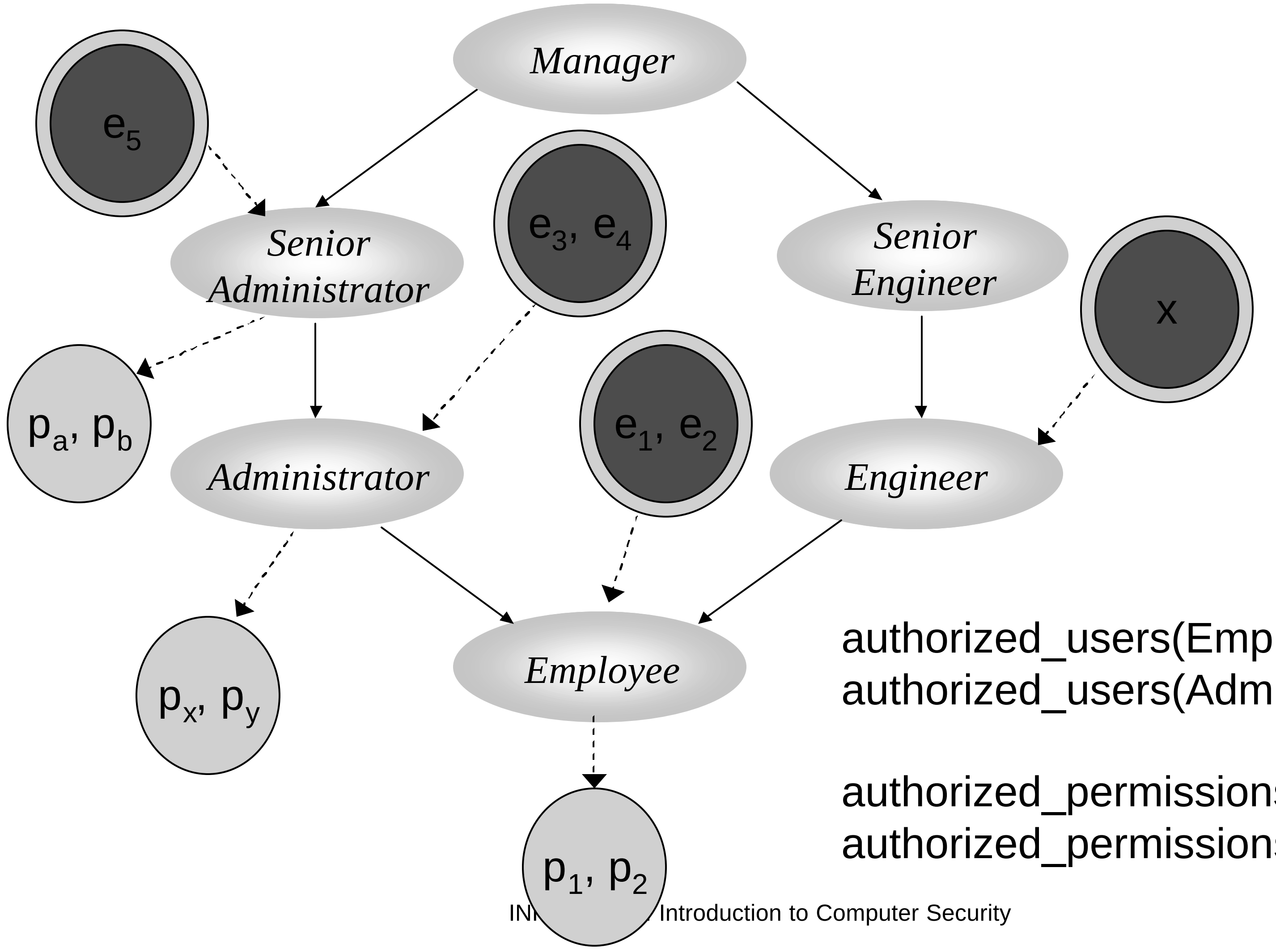
- *authorized_permissions*: Roles$\rightarrow 2^{\text{Permissions}}$

  $authorized\_users(\mathrm{r}) = \{p \mid r' = r \,\&(p, r') \in PA\}$

- RH ? Roles x Roles is a partial order
  - called the inheritance relation
  - written as =.

  $(r_1 = r_2) \rightarrow authorized\_users(r_1)$? $authorized\_users(r_2)$ & $authorized\_permisssions(r_2)$? $authorized\_permisssions(r_1)$

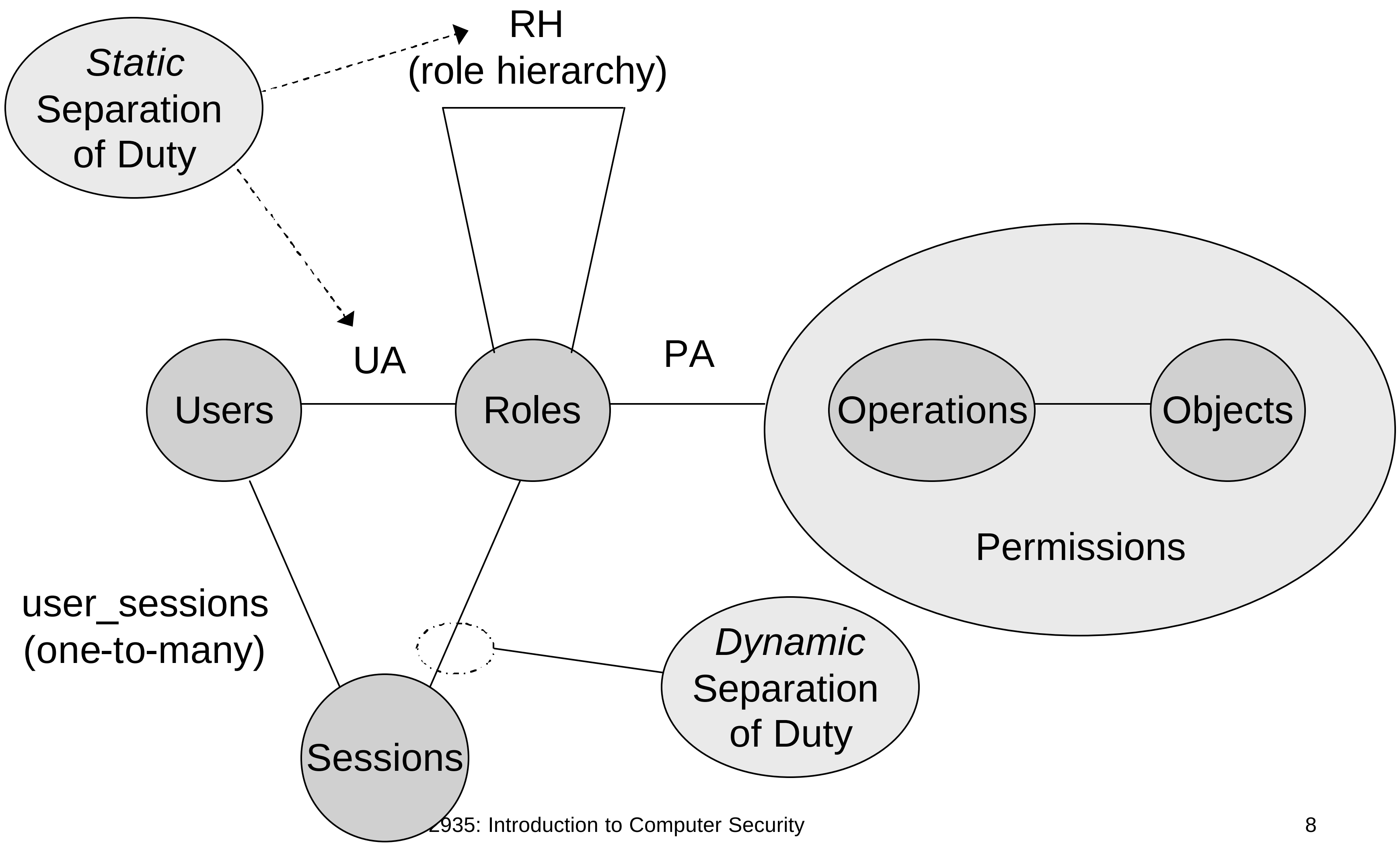
Example



authorized_users(Employee)?
authorized_users(Administrator)?

authorized_permissions(Employee)?
authorized_permissions(Administrator)?

Constrained RBAC



Static Separation of Duty

- *SSD* ? 2^{Roles} x N
- In absence of hierarchy
  - Collection of pairs (*RS*, *n*) where *RS* is a role set, *n* = 2;
    for all (*RS*, *n*) $\in$ *SSD*, *for all t* ? *RS*:

    $$|t| = n \rightarrow \text{n}_{r \in t}\, \textit{assigned\_users}(\text{r}) = \emptyset$$
- In presence of hierarchy
  - Collection of pairs (RS, n) where RS is a role set, n = 2;

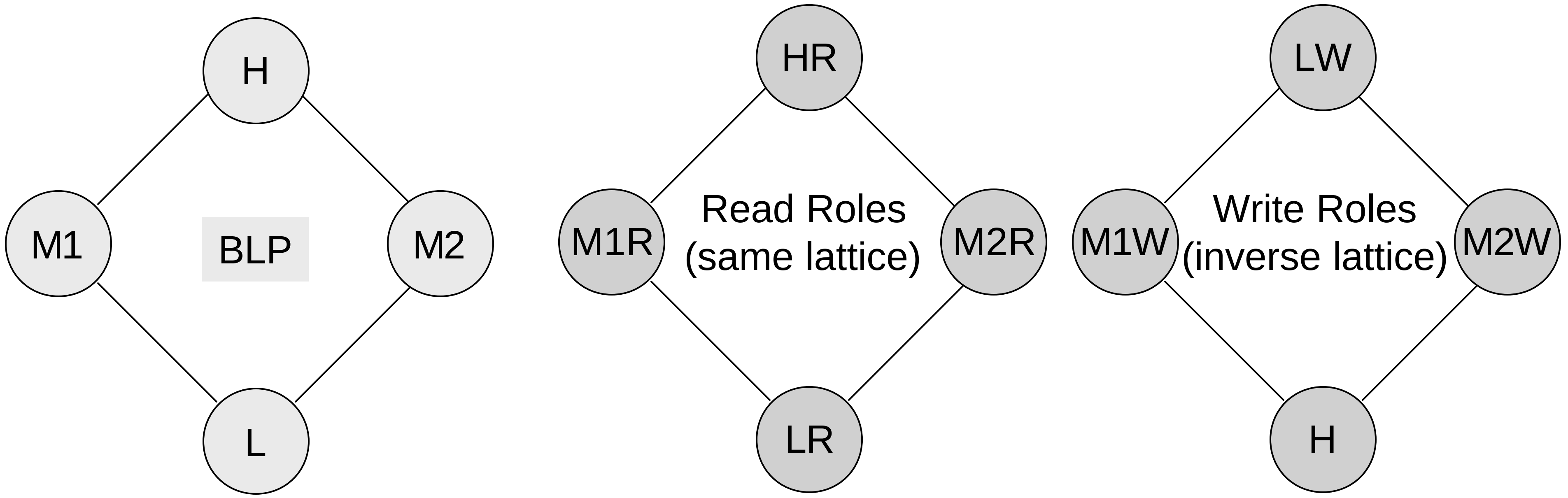
    for all (*RS*, *n*) $\in$ *SSD*, *for all t* ? *RS*:

    $$|t| = n \rightarrow \text{n}_{r \in t}\, \textit{authorized\_uers}(r) = \emptyset$$

Dynamic Separation of Duty

- *DSD* ? 2^{Roles} x N
  - Collection of pairs (*RS*, *n*) where *RS* is a role set, *n* = 2;
  - A user cannot activate *n* or more roles from RS
  - Formally?? [HW3?]
  - What if both SSD and DSD contains (*RS*, *n*)?
    - Consider (*RS*, *n*) = ({r_1, r_2, r_3}, 2)?
    - If SSD – can r_1, r_2 *and* r_3 be assigned to *u*?
    - If DSD – can r_1, r_2 *and* r_3 be assigned to *u*?

MAC using RBAC



Transformation rules

- R = {L_1R, L_2R,…, L_nR, L_1W, L_2W,…, L_nW}
- Two separate hierarchies for {L_1R, L_2R,…, L_nR} and { L_1W, L_2W,…, L_nW}
- Each user is assigned to exactly two roles: xR and LW
- Each session has exactly two roles yR and yW
- Permission (o, r) is assigned to xR iff (o, w) is assigned to xW)

RBAC’s Benefits

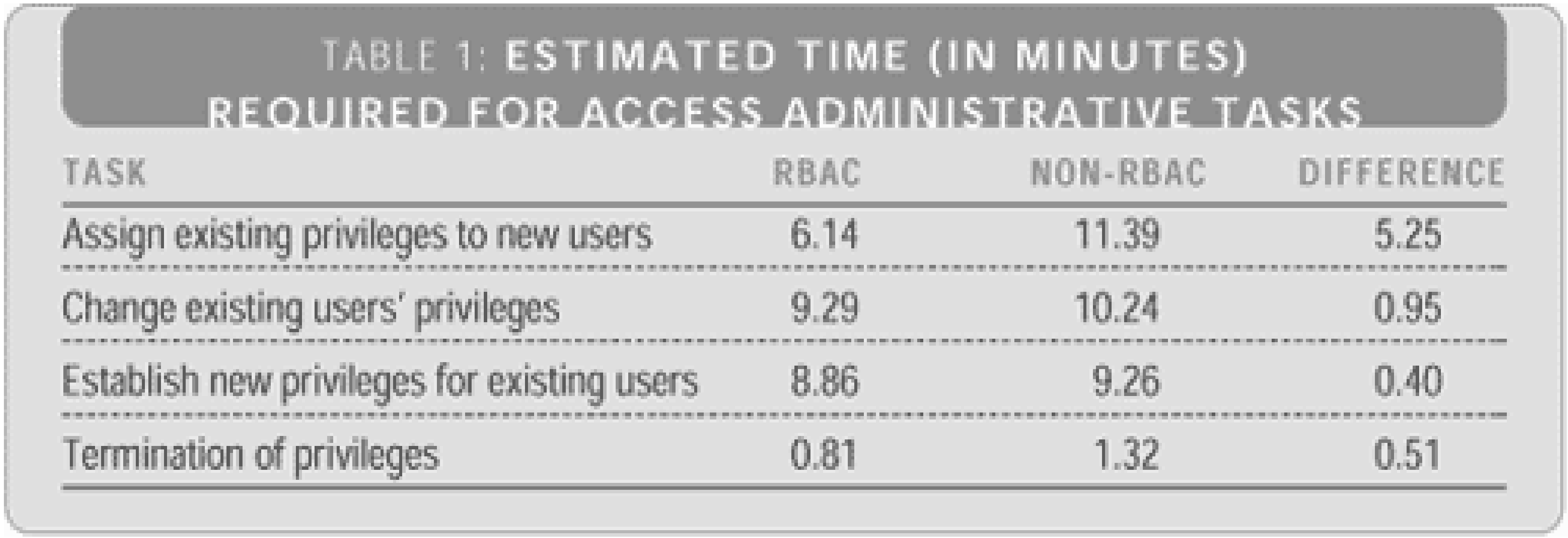


TABLE 1: ESTIMATED TIME (IN MINUTES) REQUIRED FOR ACCESS ADMINISTRATIVE TASKS

TASK	RBAC	NON-RBAC	DIFFERENCE
Assign existing privileges to new users	6.14	11.39	5.25
Change existing users' privileges	9.29	10.24	0.95
Establish new privileges for existing users	8.86	9.26	0.40
Termination of privileges	0.81	1.32	0.51

Cost Benefits

- Saves about 7.01 minutes per employee, per year in administrative functions
  - Average IT amin salary - $59.27 per hour
  - The annual cost saving is:
    - $6,924/1000; $692,471/100,000
- Reduced Employee downtime
  - if new transitioning employees receive their system privileges faster, their productivity is increased
  - 26.4 hours for non-RBAC; 14.7 hours for RBAC
  - For average employee wage of $39.29/hour, the annual productivity cost savings yielded by an RBAC system:
    - $75000/1000; $7.4M/100,000

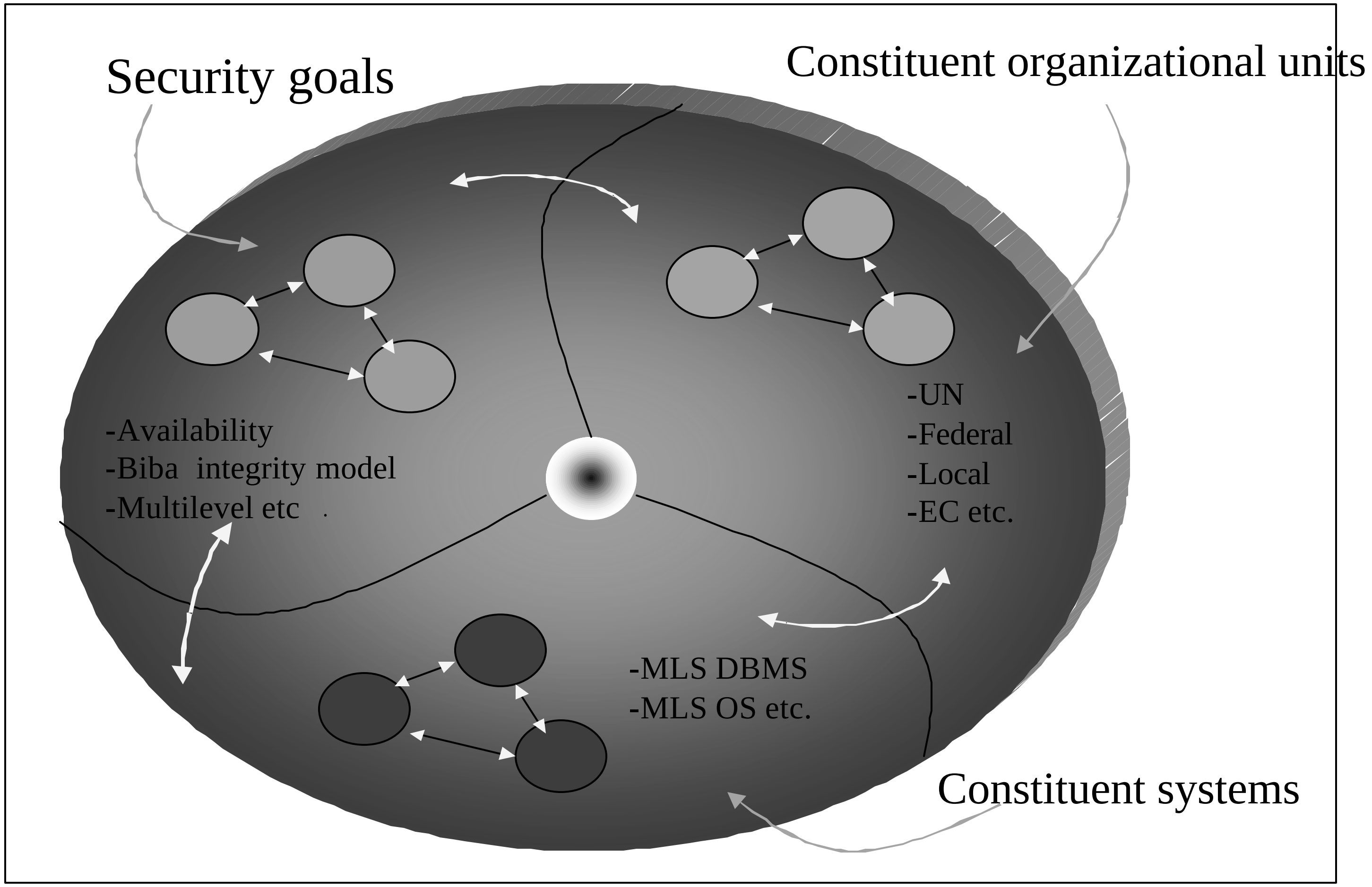
Policy Composition

Problem: *Consistent* Policies

- Policies defined by different organizations
  - Different needs
  - But sometimes subjects/objects overlap
- Can all policies be met?
  - Different categories
    - Build lattice combining them
  - Different security levels
    - Need to be *levels* – thus must be able to order
  - What if different DAC and MAC policies need to be integrated?

Multidomain Environments

- Heterogeneity exists at several levels



Multidomain Challenges

Key challenges

- Semantic heterogeneity
- Secure interoperation
- Assurance and risk propagation
- Security Management

Semantic heterogeneity

- Different systems use different security policies
  - e.g., Chinese wall, BLP policies etc.
- Variations of the same policies
  - e.g., BLP model and its variations
- Naming conflict on security attributes
  - Similar roles with different names
  - Similar permission sets with different role names
- Structural conflict
  - different multilevel lattices / role hierarchies
- Different Commercial-Off-The-Self (COTS) products

Secure Interoperability

- Principles of secure interoperation [Gong, 96]

  Principle of autonomy

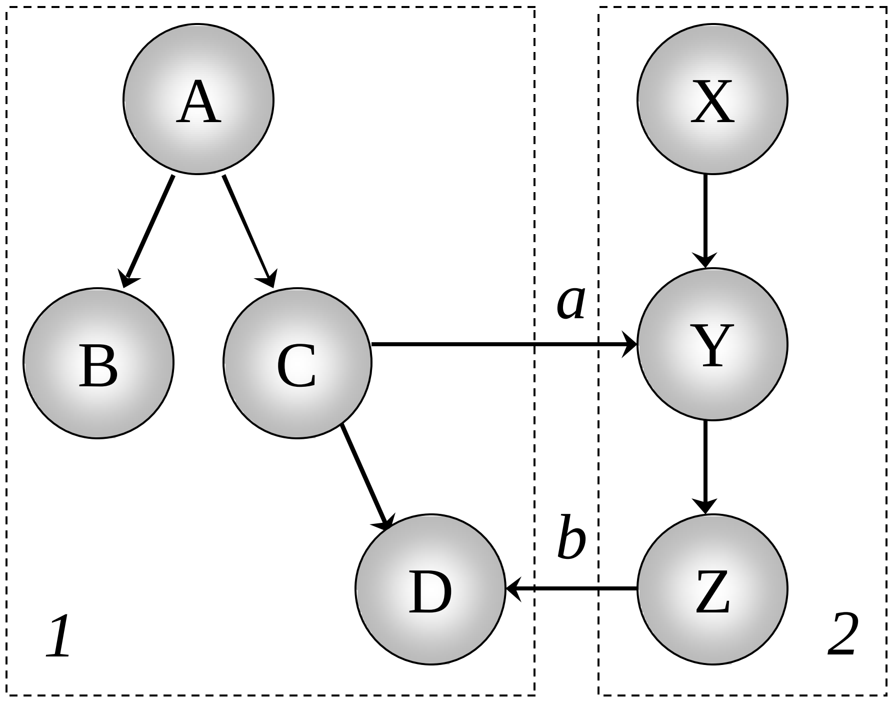
  - If an access is permitted within an individual system, it must also be permitted under secure interoperation

  Principle of security

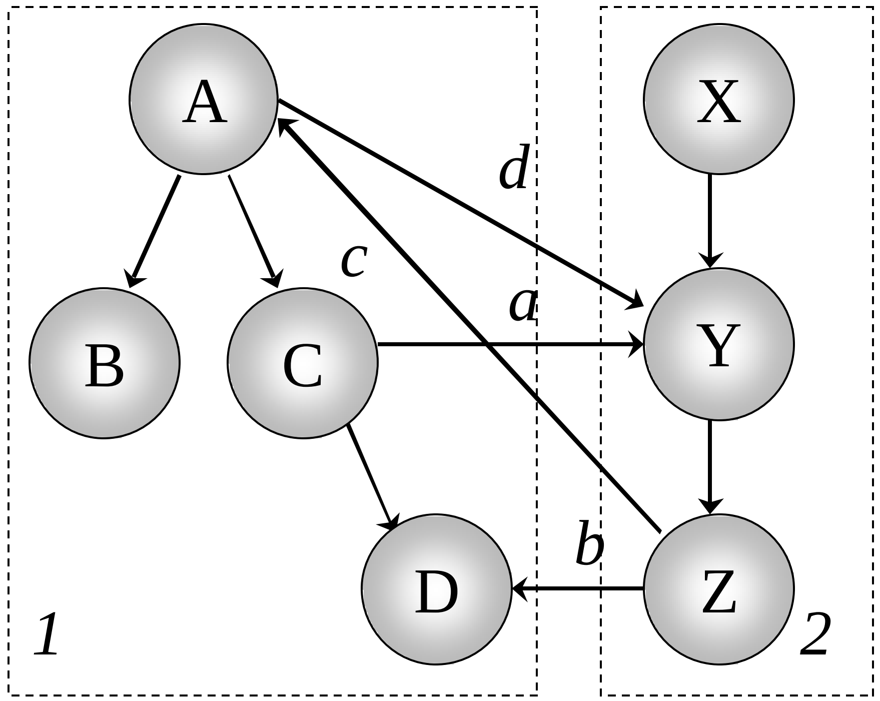
  - If an access is not permitted within an individual system, it must not be permitted under secure interoperation

- Interoperation of secure systems can create new security breaches

Secure Interoperability (Example)



F_{12} = {a, b}



F_{12} = {a, b, c, d}

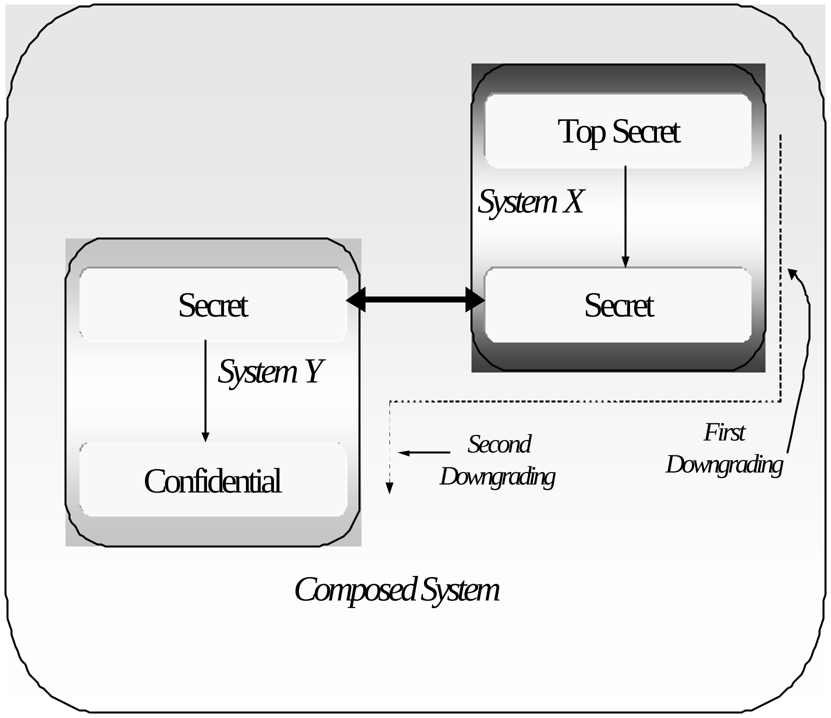
F_{12} - permitted access between systems 1 and 2

(1) F_{12} = {a, b, d}
Direct access

(2) F_{12} = {c}
Indirect access

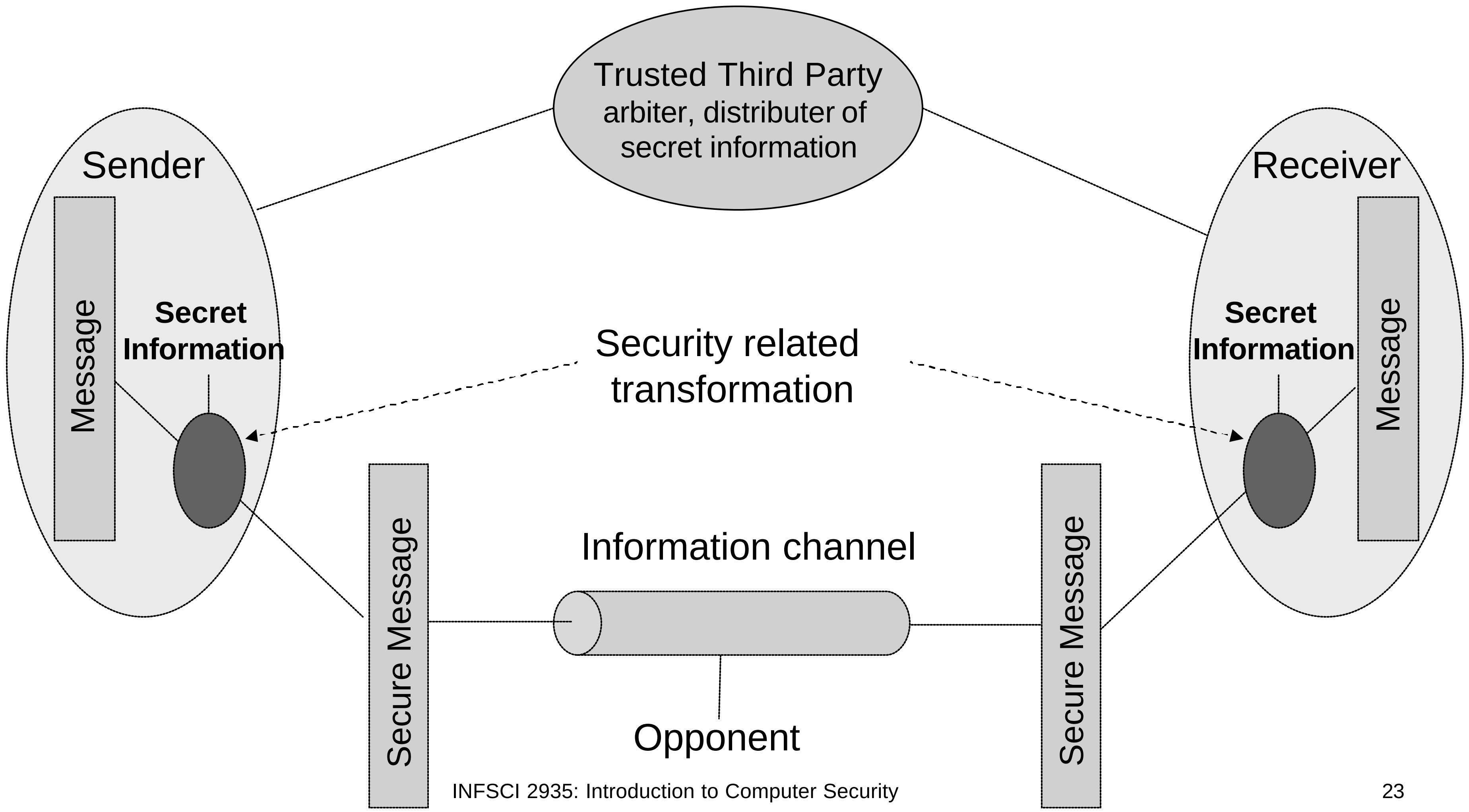
Assurance and Risk Propagation & Security Management

- Assurance and Risk propagation
  - A breach in one component affects the whole environment
  - Cascading problem
- Management
  - Centralized/Decentralized
  - Managing metapolicy
  - Managing policy evolution

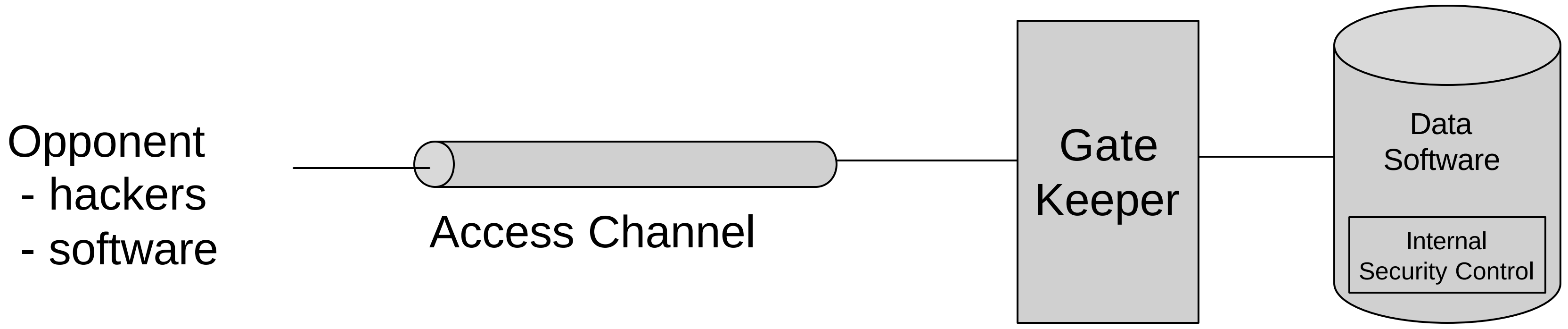


Cryptography & Network Security

Secure Information Transmission (network security model)



Security of Information Systems (Network access model)



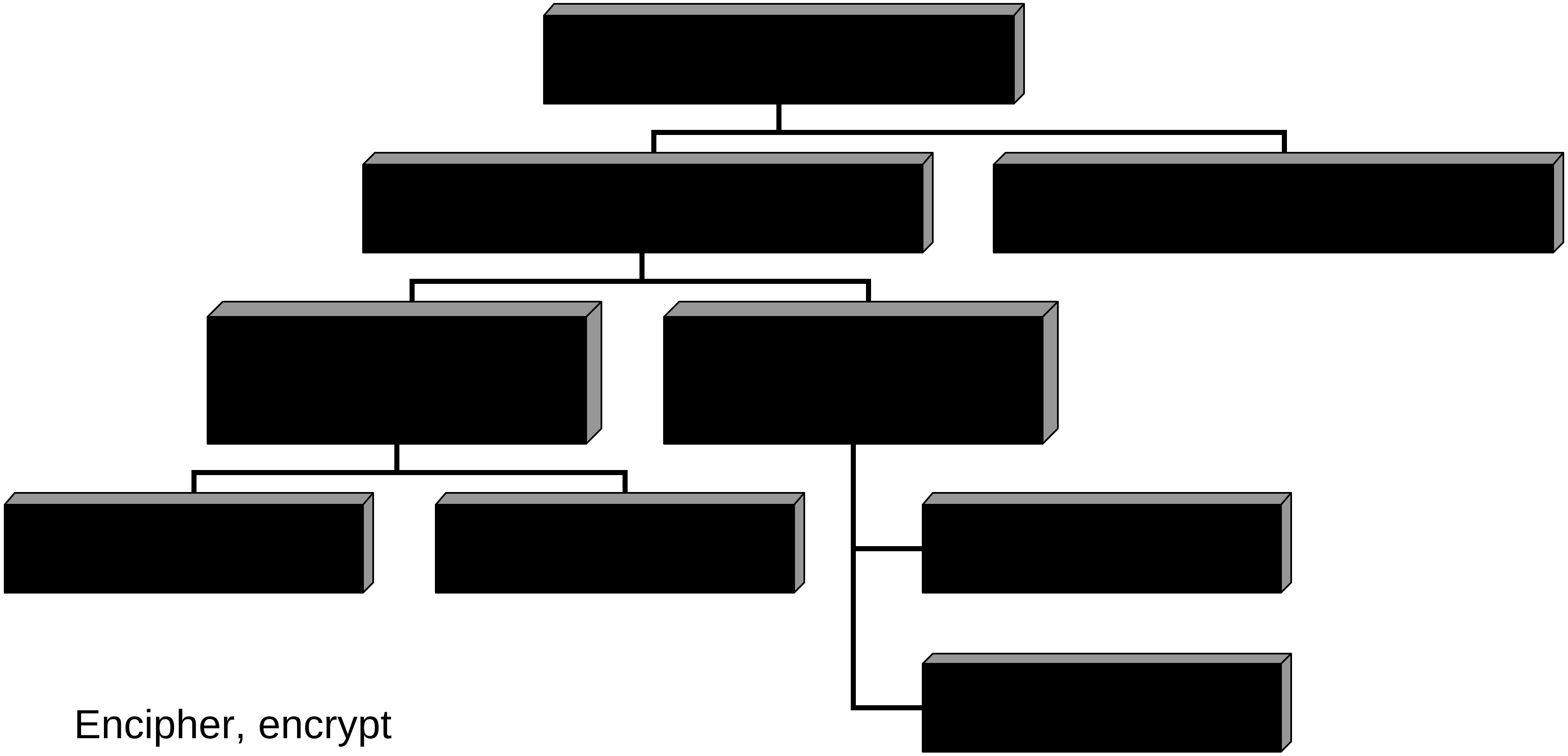
Gatekeeper – firewall or equivalent, password-based login

Internal Security Control – Access control, Logs, audits, virus scans etc.

Issues in Network security

- Distribution of secret information to enable secure exchange of information is important
- Effect of communication protocols needs to be considered
- Encryption (cryptography) *if used cleverly and correctly*, can provide several of the security services
- Physical and logical placement of security mechanisms
- Countermeasures need to be considered

Cryptology



Encipher, encrypt
Decipher, decrypt

The modulo operation

- What is 27 mod 5?
- Definition
  - Let *a*, *r*, *m* be integers and let $m > 0$
  - We write $a \equiv r \bmod m$ if *m* divides $r – a$ (or $a – r$) and $0 \leq r < m$
  - *m* is called the modulus
  - *r* is called the remainder
    - Note that *r* is positive or zero
  - Note that $a = m.q + r$ where *q* is another integer (quotient)
- Example: $42 \equiv 6 \bmod 9$
  - 9 divides 42-6 = 36
  - 9 also divides 6-42 = -36
  - Note that 42 = 9.4 + 6
    - (q = 4)

Elementary Number Theory

- Natural numbers N = {1,2,3,…}
- Whole numbers W = {0,1,2,3, …}
- Integers Z = {…,-2,-1,0,1,2,3, …}
- Divisors
  - A number *b* is said to divide *a* if $a = mb$ for some *m* where $a,b,m \in$ Z
  - We write this as $b \mid a$
    - Read as "*b* divides *a*"

Divisors

- Some common properties
  - If $a \mid 1$, $a = +1$ or -1
  - If $a|b$ and $b|a$ then $a = +b$ or $-b$
  - Any $b \in Z$ divides 0 if $b \neq 0$
  - If $b|g$ and $b|h$ then $b|(mg + nh)$ where $b,m,n,g,h \in Z$
- Examples:
  - The positive divisors of 42 are 1,2,3,6,7,14,21,42
  - 3|6 and 3|21 => 3|21m+6n for $m,n \in Z$

Prime Numbers

- An integer *p* is said to be a prime number if its only positive divisors are 1 and itself
  - 1, 3, 7, 11, ..
- Any integer can be expressed as a ***unique*** product of prime numbers raised to positive integral powers
- Examples
  - $7569 = 3 \times 3 \times 29 \times 29 = 3^2 \times 29^2$
  - $5886 = 2 \times 27 \times 109 = 2 \times 3^3 \times 109$
  - $4900 = 7^2 \times 5^2 \times 2^2$
  - 100 = ?
  - 250 = ?
- This process is called ***Prime Factorization***

Greatest common divisor (GCD)

- Definition: Greatest Common Divisor
  - This is the largest divisor of *both* *a* and *b*
- Given two integers *a* and *b*, the positive integer *c* is called their GCD or greatest common divisor if and only if
  - *c* | *a* and *c* | *b*
  - Any divisor of both *a* and *b* also divides *c*
- Notation: gcd(*a*, *b*) = *c*
- Example: gcd(49,63) = ?

Relatively Prime Numbers

- Two numbers are said to be relatively prime if their gcd is 1
  - Example: 63 and 22 are relatively prime
- How do you determine if two numbers are relatively prime?
  - Find their GCD or
  - Find their prime factors
    - If they do not have a common prime factor other than 1, they are relatively prime
  - Example: $63 = 9 \times 7 = 3^2 \times 7$ and $22 = 11 \times 2$

Modular Arithmetic Again

- We say that $a \equiv b \bmod m$ if $m \mid a - b$
  - Read as: *a* is congruent to *b* modulo *m*
  - *m* is called the modulus
  - Example: $27 \equiv 2 \bmod 5$
- Note that *b* is the *remainder* after dividing *a* by *m* BUT
  - Example: $27 \equiv 7 \bmod 5$ and $7 \equiv 2 \bmod 5$
- $a \equiv b \bmod m \Rightarrow b \equiv a \bmod m$
  - Example: $2 \equiv 27 \bmod 5$
- We usually consider the *smallest positive remainder* which is sometimes called the residue

Modulo Operation

- The modulo operation “reduces” the infinite set of integers to a finite set
- Example: modulo 5 operation
  - We have five sets
    - {…,-10, -5, 0, 5, 10, …} => $a \equiv 0 \bmod 5$
    - {…,-9,-4,1,6,11,…} => $a \equiv 1 \bmod 5$
    - {…,-8,-3,2,7,12,…} => $a \equiv 2 \bmod 5$, etc.
  - The set of residues of integers modulo 5 has five elements {0,1,2,3,4} and is denoted Z_5.

Brief History

- All encryption algorithms from BC till 1976 were secret key algorithms
  - Also called private key algorithms or symmetric key algorithms
  - Julius Caesar used a substitution cipher
  - Widespread use in World War II (enigma)
- Public key algorithms were introduced in 1976 by Whitfield Diffie and Martin Hellman

Cryptosystem

- (E, D, M, K, C)
  - E set of encryption functions $e: M \times K \to C$
  - D set of decryption functions $d: C \times K \to M$
  - M set of plaintexts
  - K set of keys
  - C set of ciphertexts

Example

- Example: Cæsar cipher
  - M = { sequences of letters }
  - K = { i | i is an integer and 0 = i = 25 }
  - E = { E_k | $k \in$ K and for all letters m,

    $E_k(m) = (m + k) \bmod 26$ }
  - D = { D_k | $k \in$ K and for all letters c,

    $D_k(c) = (26 + c - k) \bmod 26$ }
  - C = M

Cæsar cipher

- Let k = 9, m = “VELVET” (21 4 11 21 4 19)
  - $E_k(m)$ = (30 13 20 30 13 28) mod 26
    =“4 13 20 4 13 2” = “ENUENC”
  - $D_k(m)$ = (26 + c – k) mod 26
    = (21 30 37 21 30 19) mod 26
    = “21 4 11 21 4 19” = “VELVET”

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Attacks

- *Ciphertext only*:
  - adversary has only Y;
  - goal is to find plaintext, possibly key
- *Known plaintext*:
  - adversary has X, Y;
  - goal is to find K
- *Chosen plaintext*:
  - adversary may gets a specific plaintext enciphered;
  - goal is to find key

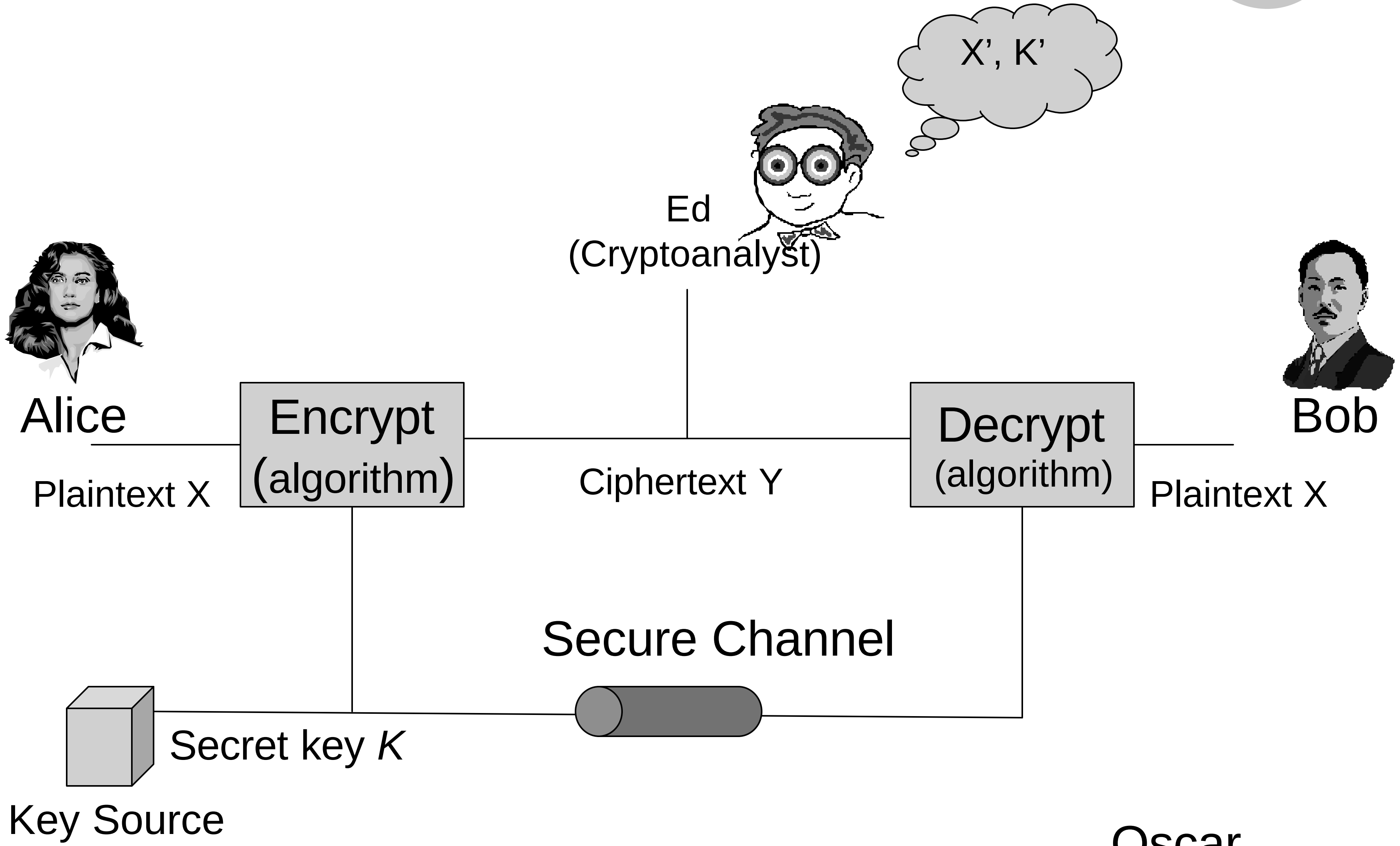
Attacking a conventional cryptosystem

- Cryptoanalysis:
  - Art/Science of breaking an encryption scheme
  - Exploits the characteristics of algorithm/ mathematcis
    - Recover plaintext from the ciphertext
    - Recover a key that can be used to break many ciphertexts
- Brute force
  - Tries all possible keys on a piece of ciphertext
    - If the *number of keys* is small, Ed can break the encryption easily

Basis for Cyptoanalysis

- 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.

Classical Cryptography



Classical Cryptography

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

Classical Cryptography

- $y = E_k(x)$:   Ciphertext → Encryption
- $x = D_k(y)$:   Plaintext → Decryption
- k = encryption and decryption key
- The functions $E_k()$ and $D_k()$ must be inverses of one another
  - $E_k(D_k(y)) = ?$
  - $D_k(E_k(x)) = ?$
  - $E_k(D_k(x)) = ?$

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 that H and E are adjacent

```
HE
LL
OW
OR
LD
```

- Read off across, then down, to get original plaintext

Substitution Ciphers

- Change characters in plaintext to produce ciphertext
- Example (Cæsar cipher)
  - Plaintext is `HELLO WORLD;`
  - Key is 3, usually written as letter ‘D’
  - Ciphertext is `KHOOR ZRUOG`

Attacking the Cipher

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

Statistical Attack

- Ciphertext is `KHOOR ZRUOG`
- 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 (Denning)

a	0.080	h	0.060	n	0.070	t	0.090
b	0.015	i	0.065	o	0.080	u	0.030
c	0.030	j	0.005	p	0.020	v	0.010
d	0.040	k	0.005	q	0.002	w	0.015
e	0.130	l	0.035	r	0.065	x	0.005
f	0.020	m	0.030	s	0.060	y	0.020
g	0.015					z	0.002

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

    $\varphi(i) = 0.1p(6 - i) + 0.1p(7 - i) + 0.1p(10 - i) + 0.3p(14 - i) + 0.2p(17 - i) + 0.1p(20 - i) + 0.1p(25 - i)$
    - $p(x)$ is frequency of character x in English
  - Look for maximum correlation!

Correlation: φ(*i*) for 0 = *i* = 25

i	j (***i***)	***i***	j (***i***)	***i***	j (***i***)	***i***	j (***i***)
0	0.0482	7	0.0442	13	0.0520	19	0.0315
1	0.0364	8	0.0202	14	0.0535	20	0.0302
2	0.0410	9	0.0267	15	0.0226	21	0.0517
3	0.0575	10	0.0635	16	0.0322	22	0.0380
4	0.0252	11	0.0262	17	0.0392	23	0.0370
5	0.0190	12	0.0325	18	0.0299	24	0.0316
6	0.0660					25	0.0430

The Result

- Ciphertext is `KHOOR ZRUOG`
- Most probable keys, based on φ:
  - *i* = 6, φ(*i*) = 0.0660
    - plaintext `EBIIL TLOLA` (K = 10, (26 + 10 - 6) mod 26 = 4 = E)
  - *i* = 10, φ(*i*) = 0.0635
    - plaintext `AXEEH PHKEW` (K = 10, (26 + 10 - 10) mod 26 = 0 = A)
  - *i* = 3, φ(*i*) = 0.0575
    - plaintext `HELLO WORLD` (K = 10, (26 + 10 - 3) mod 26 = H = E)
  - *i* = 14, φ(*i*) = 0.0535
    - plaintext `WTAAD LDGAS`
- Only English phrase is for *i* = 3
  - That’s the key (3 or ‘D’)

Cæsar’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

Vigenère Cipher

- Like Cæsar cipher, but use a phrase
- Example
  - Message `THE BOY HAS THE BALL`
  - Key `VIG`
  - Encipher using Cæsar cipher for each letter:

key	`VIGVIGVIGVIGVIGV`
plain	`THEBOYHASTHEBALL`
cipher	`OPKWWECIYOPKWIRG`

Relevant Parts of Tableau

	G	*I*	*V*
A	G	I	V
B	H	J	W
E	K	M	Z
H	N	P	C
L	R	T	G
O	U	W	J
S	Y	A	N
T	Z	B	O
Y	E	H	T

- Tableau with relevant rows, columns only
- Example encipherments:
  - key V, letter T: follow V column down to T row (giving “O”)
  - Key I, letter H: follow I column down to H row (giving “P”)

Useful Terms

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

Attacking the Cipher

- Key to attacking vigenère cipher
  - determine the key length
  - If the keyword is n, then the cipher consists of n monoalphabetic substitution ciphers

key	VIGVIGVIGVIGVIGV
plain	THEBOYHASTHEBALL
cipher	O$\underline{\text{PKWWECIYO}}$PKWIRG

key	DECEPTIVEDECEPTIVEDECEPTIVE
plain	WEAREDISCOVEREDSAVEYOURSELF
cipher	ZICV$\underline{\text{TWQNGRZGV}}$TWAVZHCQYGLMGJ

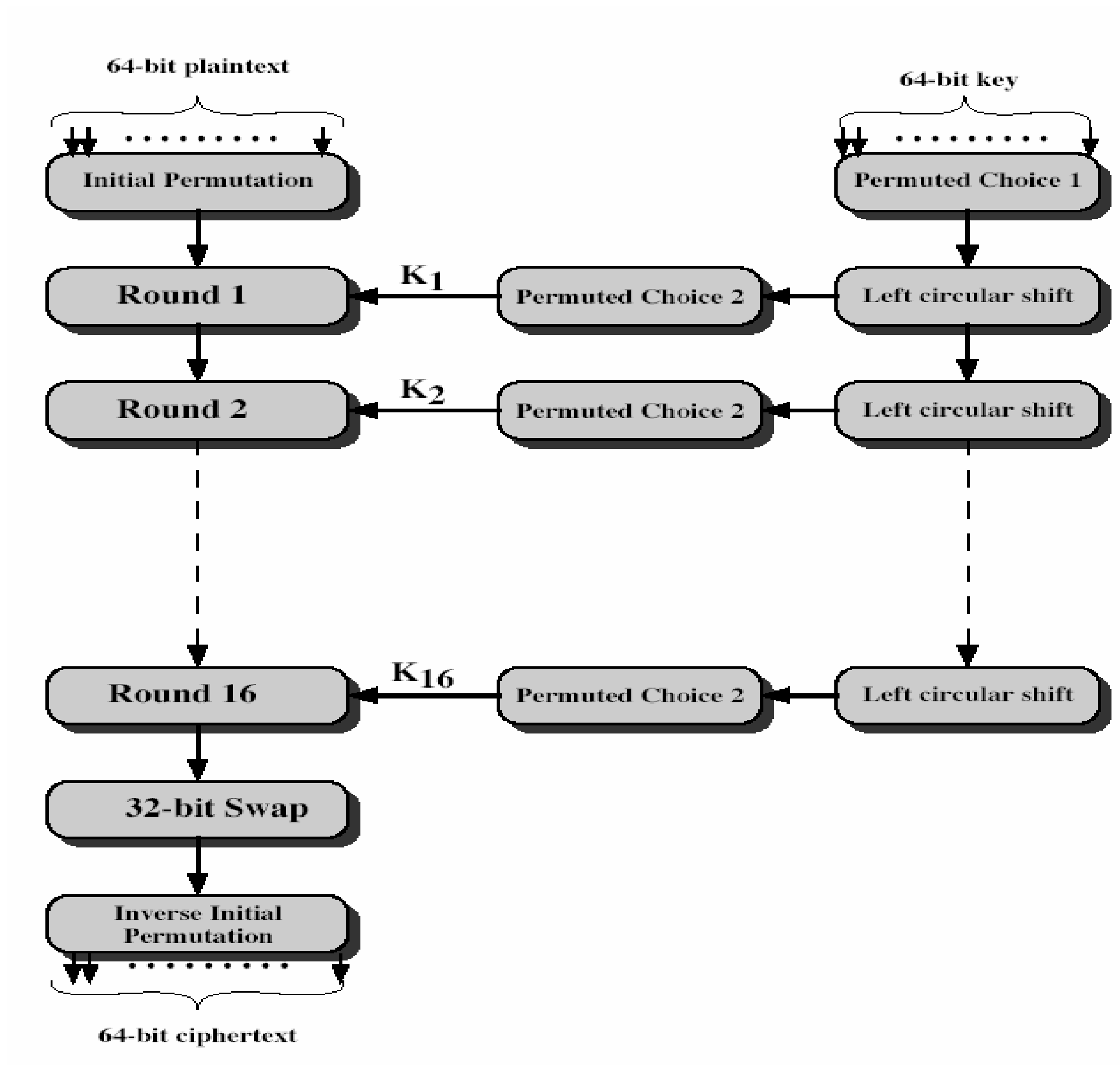
One-Time Pad

- A Vigenère cipher with a random key at least as long as the message
  - Provably unbreakable; Why?
  - Consider ciphertext `DXQR`. Equally likely to correspond to
    - plaintext `DOIT` (key `AJIY`) and
    - 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

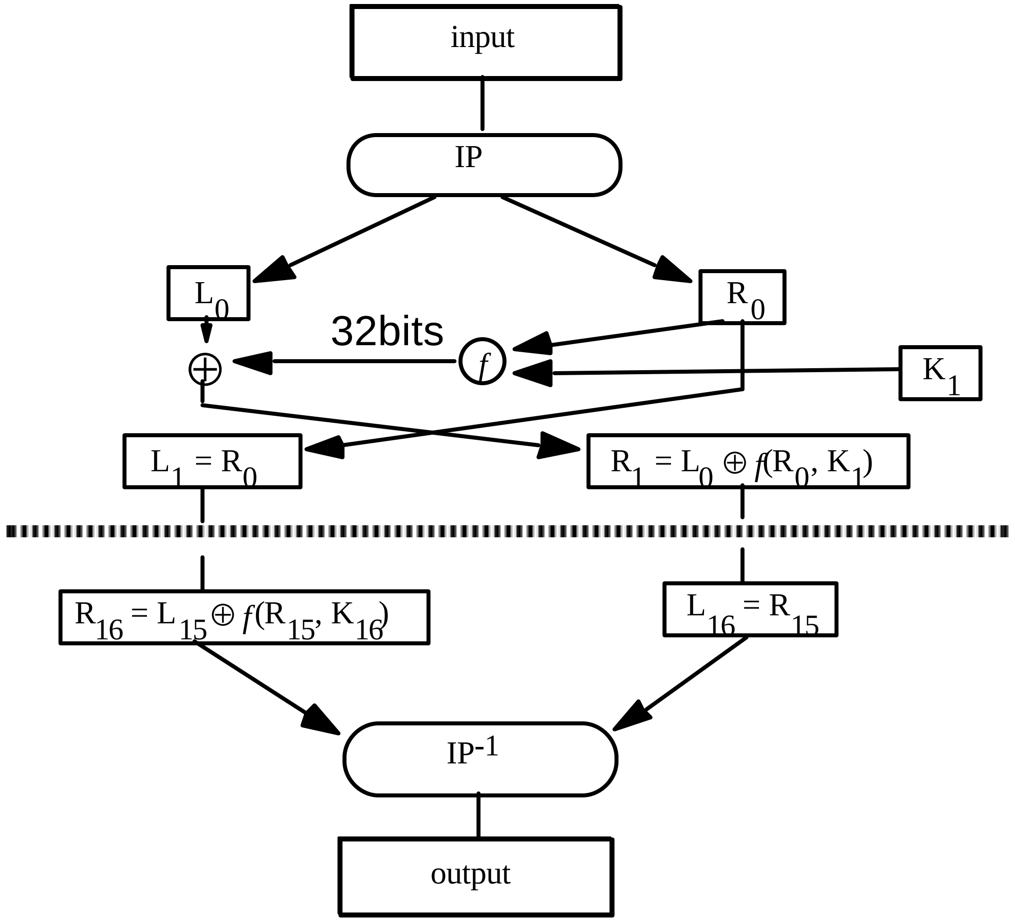
- A block cipher:
  - encrypts blocks of 64 bits using a 64 bit key
  - outputs 64 bits of ciphertext
  - A product cipher
    - performs both substitution and transposition (permutation) on the bits
  - basic unit is the bit
- Cipher consists of 16 rounds (iterations) each with a round key generated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

DES

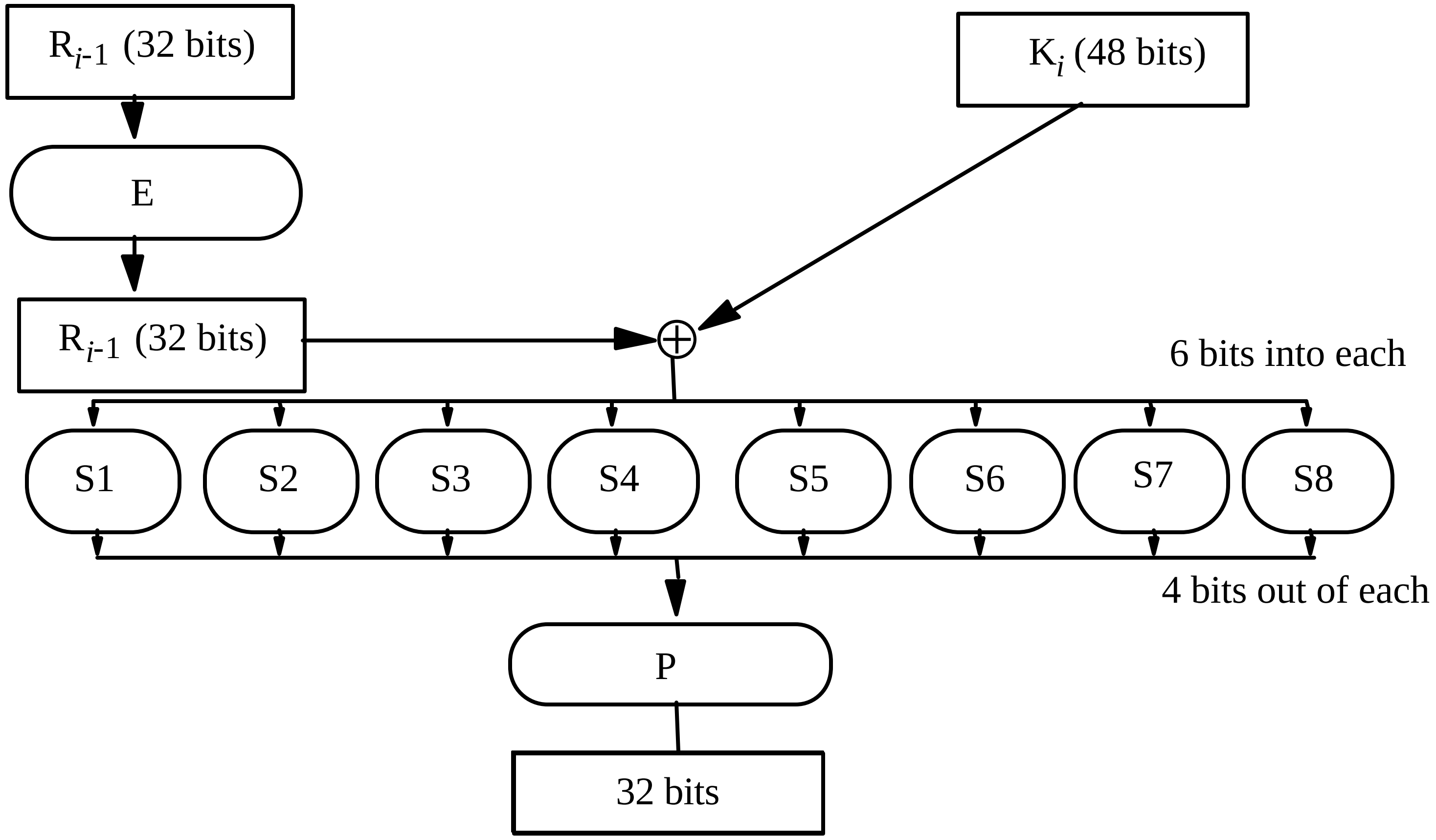


- Round keys are 48 bits each
  - Extracted from 64 bits
  - Permutation applied
- Deciphering involves using round keys in reverse

Encipherment



The *f* Function



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
  - $\mathrm{DES}_k(m) = c \Rightarrow \mathrm{DES}_{k'}(m') = c'$
- S-boxes exhibit irregular properties
  - Distribution of odd, even numbers non-random
  - Outputs of fourth box depends on input to third box

Public Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
    - Public key, private key inverses
- Idea
  - Confidentiality:
    - encipher using public key,
    - decipher using private key
  - Integrity/authentication:
    - encipher using private key,
    - decipher using public one

Requirements

1. It must be computationally easy to encipher or decipher a message given the appropriate key
2. It must be computationally infeasible to derive the private key from the public key
3. It must be computationally infeasible to determine the private key from a chosen plaintext attack

Diffie-Hellman

- Compute a common, shared key
  - Called a *symmetric key exchange protocol*
- Based on discrete logarithm problem
  - Given integers n and g and prime number p, compute k such that $n = g^k \bmod p$
  - Solutions known for small p
  - Solutions computationally infeasible as p grows large

Algorithm

- Constants known to participants
  - prime p, integer g ? 0, 1, p–1
- Anne
  - chooses private key $kAnne$,
  - computes public key $KAnne = g^{kAnne}$ mod p
- To communicate with Bob,
  - Anne computes $Kshared = KBob^{kAnne}$ mod p
- To communicate with Anne,
  - Bob computes $Kshared = KAnne^{kBob}$ mod p

Example

- Assume p = 53 and g = 17
- Alice chooses $kAlice$ = 5
  - Then $KAlice = 17^5 \bmod 53 = 40$
- Bob chooses $kBob$ = 7
  - Then $KBob = 17^7 \bmod 53 = 6$
- Shared key:
  - $KBob^{kAlice} \bmod p = 6^5 \bmod 53 = 38$
  - $KAlice^{kBob} \bmod p = 40^7 \bmod 53 = 38$

Let p = 5, g = 3
kA = 4, kB = 3

KA = ?, KB = ?,
KSshared = ?,

RSA

- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*
- Totient function $\phi(n)$
  - Number of + integers less than *n* and relatively prime to *n*
    - Relatively prime means with no factors in common with *n*
- Example: $\phi(10) = 4$
  - 1, 3, 7, 9 are relatively prime to 10
- $\phi(77)$?
- $\phi(p)$?
  - When p is a prime number
- $\phi(pq)$?
  - When p and q are prime numbers

Algorithm

- Choose two large prime numbers *p, q*
  - Let $n = pq$; then $\phi(n) = (p-1)(q-1)$
  - Choose $e < n$ relatively prime to $\phi(n)$.
  - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n); private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$

Example: Confidentiality

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
  - $17*53 \bmod 60 = ?$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends ciphertext [28 16 44 44 42]

Example

- Alice receives [28 16 44 44 42]
- Alice uses private key, *d* = 53, to decrypt message:
    - 28^{53} mod 77 = 07 H
    - 16^{53} mod 77 = 04 E
    - 44^{53} mod 77 = 11 L
    - 44^{53} mod 77 = 11 L
    - 42^{53} mod 77 = 14 O
- No one else could read it, as only Alice knows her private key and that is needed for decryption

Example: Origin Integrity/Authentication

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

Example

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$ H
  - $09^{17} \bmod 77 = 04$ E
  - $44^{17} \bmod 77 = 11$ L
  - $44^{17} \bmod 77 = 11$ L
  - $49^{17} \bmod 77 = 14$ O
- Alice sent it as only she knows her private key, so no one else could have enciphered it
- If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

Example: Confidentiality + Authentication

- Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)
  - Alice’s keys: public (17, 77); private: 53
  - Bob’s keys: public: (37, 77); private: 13
- Alice enciphers HELLO (07 04 11 11 14):
  - $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  - $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- Alice sends 07 37 44 44 14

Example: Confidentiality + Authentication

- Alice's keys: public (17, 77); private: 53
- Bob's keys: public: (37, 77); private: 13

- Bob deciphers (07 37 44 44 14):
  - $(07^{13} \bmod 77)^{17} \bmod 77 = 07$ H
  - $(37^{13} \bmod 77)^{17} \bmod 77 = 04$ E
  - $(44^{13} \bmod 77)^{17} \bmod 77 = 11$ L
  - $(44^{13} \bmod 77)^{17} \bmod 77 = 11$ L
  - $(14^{13} \bmod 77)^{17} \bmod 77 = 14$ O

Security Services

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key
- Authentication
  - Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner

More Security Services

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

Warnings

- Encipher message in blocks considerably larger than the examples here
  - If 1 character per block, RSA can be broken using statistical attacks (just like classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning
    - Example: reverse enciphered message of text ON to get NO

Security Levels

- Unconditionally Secure
  - Unlimited resources + unlimited time
  - Still the plaintext CANNOT be recovered from the ciphertext
- Computationally Secure
  - Cost of breaking a ciphertext exceeds the value of the hidden information
  - The time taken to break the ciphertext exceeds the useful lifetime of the information

Key Points

- Two main types of cryptosystems: classical and public key
- Classical cryptosystems encipher and decipher using the same key
  - Or one key is easily derived from the other
- Public key cryptosystems encipher and decipher using different keys
  - Computationally infeasible to derive one from the other

Notation

- $X \rightarrow Y : \{ Z \,||\, W \}\, k_{X,Y}$
  - *X* sends *Y* the message produced by concatenating *Z* and *W* enciphered by key $k_{X,Y}$, which is shared by users *X* and *Y*
- $A \rightarrow T : \{ Z \}\, k_A \,||\, \{ W \}\, k_{A,T}$
  - *A* sends *T* a message consisting of the concatenation of *Z* enciphered using k_A, *A*'s key, and *W* enciphered using $k_{A,T}$, the key shared by *A* and *T*
- r_1, r_2 nonces (nonrepeating random numbers)

Session, Interchange Keys

- Alice wants to send a message *m* to Bob
  - Assume public key encryption
  - Alice generates a random cryptographic key k_s and uses it to encipher *m*
    - To be used for this message *only*
    - Called a *session key*
  - She enciphers k_s with Bob's public key k_B
    - k_B enciphers all session keys Alice uses to communicate with Bob
    - Called an interchange *key*
  - Alice sends $\{ m \} k_s \{ k_s \} k_B$

Benefits

- Limits amount of traffic enciphered with single key
  - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
  - Example: Alice will send Bob message that is either “BUY” or “SELL”. Eve computes possible ciphertexts {“BUY”} k_B and {“SELL”} k_B. Eve intercepts enciphered message, compares, and gets plaintext at once

Key Exchange Algorithms

- Goal: Alice, Bob get shared key
  - Key cannot be sent in clear
    - Attacker can listen in
    - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
  - Alice, Bob may trust third party
  - All cryptosystems, protocols publicly known
    - Only secret data is the keys, ancillary information known only to Alice and Bob needed to derive keys
    - Anything transmitted is assumed known to attacker

Classical Key Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can’t send it to Bob in the clear!
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key k_A
  - Bob and Cathy share secret key k_B
- Use this to exchange shared key k_s

Simple Protocol

$$\begin{tikzcd}
\text{Alice} \arrow[rrrr, dash, "\{\text{ request for session key to Bob }\}\ k_A"] & & & & \text{Cathy}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Alice} \arrow[rrrr, dash, "\{\ k_s\ \}\ k_A \parallel \{\ k_s\ \}\ k_B"] & & & & \text{Cathy}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Alice} \arrow[rrrr, dash, "\{\ k_s\ \}\ k_B"] & & & & \text{Bob}
\end{tikzcd}$$

Problems

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- Protocols must provide authentication and defense against replay

Needham-Schroeder

Alice $\xrightarrow{\text{Alice} \parallel \text{Bob} \parallel r_1}$ Cathy

Alice $\xrightarrow{\{ \text{Alice} \parallel \text{Bob} \parallel r_1 \parallel k_s \parallel \{ \text{Alice} \parallel k_s \} k_B \} k_A}$ Cathy

Alice $\xrightarrow{\{ \text{Alice} \parallel k_s \} k_B}$ Bob

Alice $\xrightarrow{\{ r_2 \} k_s}$ Bob

Alice $\xrightarrow{\{ r_2 - 1 \} k_s}$ Bob

Argument: Alice talking to Bob

- Second message
  - Enciphered using key only she, Cathy know
    - So Cathy enciphered it
  - Response to first message
    - As r_1 in it matches r_1 in first message
- Third message
  - Alice knows only Bob can read it
    - As only Bob can derive session key from message
  - Any messages enciphered with that key are from Bob

Argument: Bob talking to Alice

- Third message
  - Enciphered using key only he, Cathy know
    - So Cathy enciphered it
  - Names Alice, session key
    - Cathy provided session key, says Alice is other party
- Fourth message
  - Uses session key to determine if it is replay from Eve
    - If not, Alice will respond correctly in fifth message
    - If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly

Denning-Sacco Modification

- Assumption: all keys are secret
- Question: suppose Eve can obtain session key. How does that affect protocol?
  - In what follows, Eve knows k_s

$$\text{Eve} \xrightarrow{\{\ \text{Alice} \,\|\, k_s\ \}\ k_B} \text{Bob}$$

$$\text{Eve} \xrightarrow{\{\ r_2\ \}\ k_s} \text{Bob}$$

$$\text{Eve} \xrightarrow{\{\ r_2 - 1\ \}\ k_s} \text{Bob}$$

Solution

- In protocol above, Eve impersonates Alice
- Problem: replay in third step
  - First in previous slide
- Solution: use time stamp *T* to detect replay
- Weakness: if clocks not synchronized, may either reject valid messages or accept replays
  - Parties with either slow or fast clocks vulnerable to replay
  - Resetting clock does *not* eliminate vulnerability

Needham-Schroeder with Denning-Sacco Modification

$$\text{Alice} \quad \overset{\text{Alice} \,\|\, \text{Bob} \,\|\, r_1}{\overline{\hspace{20em}}} \quad \text{Cathy}$$

$$\text{Alice} \quad \overset{\{\, \text{Alice} \,\|\, \text{Bob} \,\|\, r_1 \,\|\, k_s \,\|\, \{\, \text{Alice} \,\|\, T \,\|\, k_s \,\}\, k_B \,\}\, k_A}{\overline{\hspace{20em}}} \quad \text{Cathy}$$

$$\text{Alice} \quad \overset{\{\, \text{Alice} \,\|\, T \,\|\, k_s \,\}\, k_B}{\overline{\hspace{20em}}} \quad \text{Bob}$$

$$\text{Alice} \quad \overset{\{\, r_2 \,\}\, k_s}{\overline{\hspace{20em}}} \quad \text{Bob}$$

$$\text{Alice} \quad \overset{\{\, r_2 - 1 \,\}\, k_s}{\overline{\hspace{20em}}} \quad \text{Bob}$$