

Unit 10

Security and Integrity

Contents

- ❑ 10.1 Introduction
- ❑ 10.2 Security
- ❑ 10.3 Integrity
- ❑ 10.4 Security and Integrity in INGRES
- ❑ 10.5 Security in Statistical Databases
- ❑ 10.6 Data Encryption

10.1 Introduction

Security and Integrity

- Objective
 - **Security**: protect data against unauthorized disclosure, alteration, or destruction.
 - **Integrity**: ensure data valid or accurate.
- Problem
 - **Security**: allowed or not ?
 - **Integrity**: correct or not ?
- Similarity between Security and Integrity
 - the system needs to be aware of certain **constraints** that the users must not violate.
 - **constraints** must be specified (by DBA) in some languages.
 - **constraints** must be maintained in the **system catalog** (or dictionary).
 - DBMS must monitor user interactions.

10.2 Security

General Considerations

■ Aspects of the security problem:

- Legal, social, ethical
- Physical control
- O.S. security
- **DBMS**

■ The unit of data for security purpose

- an entire database
- a relation
- a specific row-and-column data

	S.S#	S.Status	S. ...	P	SP
Charley	1	1	1 ...	0	0

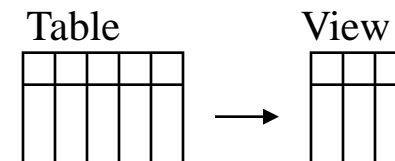
■ Access Control Matrix

- A given user typically have different access rights on different objects.
- Different users may have different access rights on the same object.
- Access right on object: read, write, owner, ...

Security on SQL: View Mechanism

- Two features

- View mechanism: hide sensitive data.
- Authorization subsystem: specify access right.



- **View Mechanism**

<e.g.1> For a user permitted access only supplier records located in Paris :

```
CREATE VIEW PARIS_SUPPLIERS
AS SELECT S#, SNAME, STATUS, CITY
FROM S
WHERE city = 'Paris'; /*value dependent*/
```

- Users of this view see a horizontal subset, similar views can be created for vertical subset or row-and-column subset.

<e.g.2> For a user permitted access to catalog entries for tables created by that user:

```
CREATE VIEW MY_TABLES
AS SELECT *
FROM SYSTABLE
WHERE CREATOR = USER; /*context dependent*/
```

Security on SQL (cont.)

<e.g.3> For a user permitted access to average shipment quantities per supplier, but not to any individual quantities :

```
CREATE VIEW AVQ ( S#, AVGQTY )
AS SELECT S# , AVG(QTY)
FROM SP
GROUP BY S#; /*statistical summary*/
```

- **Advantages of view mechanism**
 - Provide security for free.
 - many authorization checks can be applied at compile time.

Security on SQL: Authorization Subsystem

- In DB2, the installation procedure

- specify a privilege user as the system administrator.
- the system administrator is given a special authority SYSADM, means the holder can perform every operation the system support.
- the system administrator grant rights to other user.
- use access control matrix

	S.S#	S.Status	S. ...	P	SP
Charley	1	1	1 ...	0	0

- <e.g.1> [GRANT]

```
GRANT SELECT ON TABLE S TO CHARLEY;
GRANT SELECT, UPDATE ( STATUS, CITY ) ON TABLE S TO JUDY, JACK, JOHN;
GRANT ALL ON TABLE S, P, SP TO FRED, MARY;
GRANT SELECT ON TABLE P TO PUBLIC;
GRANT INDEX ON TABLE S TO PHIL;
```

- <e.g.2> [REVOKE]

```
REVOKE SELECT ON TABLE S FROM CHARLEY;
REVOKE UPDATE ON TABLE S FROM JOHN;
REVOKE INSERT, DELETE ON TABLE SP FROM NANCY, JACK;
REVOKE ALL ON TABLE S, P, SP FROM SAM
```

Security on SQL: Authorization Subsystem

- **The rights that apply to tables (both base tables and views):**

- SELECT
- UPDATE: can specify column
- DELETE
- INSERT

- **The rights that apply to base tables only**

- ALTER: right to execute ALTER TABLE
- INDEX: right to execute CREATE INDEX

- **The GRANT option**

User U1: GRANT SELECT ON TABLE S TO U2 WITH GRANT OPTION;

User U2: GRANT SELECT ON TABLE S TO U3 WITH GRANT OPTION;

User U3: GRANT SELECT ON TABLE S TO U4 WITH GRANT OPTION;

- **The REVOKE will cascade**

User U1: REVOKE SELECT ON TABLE S FROM U2;

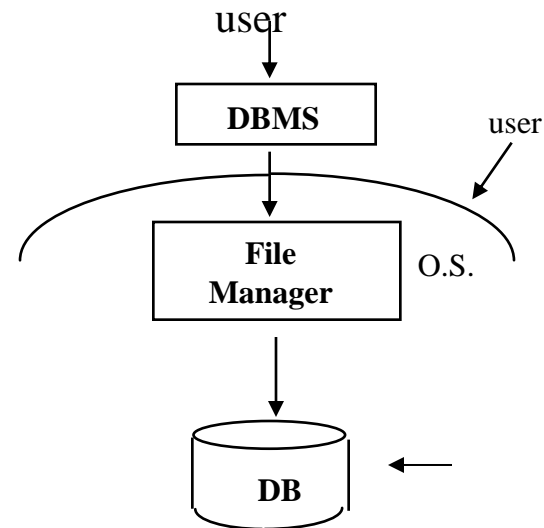
- U3, U4, are revoked automatically!

- **Authorization can be queried (recorded in system catalog).**

Aspects of Security

- **Other Aspects of Security**

- The total system should be secure.
- Not to assume the security system is perfect
 - Audit Trail: keep track of all operations' information.
<e.g.> terminal, user, date, time, ...
- Statistical Databases: 10.5
- Data Encryption: 10.6
- Access Control Schemes (papers)



10.3 Integrity

General Considerations of Integrity

- “**Integrity**” refers to accuracy or correctness.
- Most of **integrity checking** today is still done by user-written procedure.
- It is preferable to specify **integrity constraints** in declarative fashion and have the system to do the check.
- Integrity constraint can be regarded as a condition that all correct states of the database required to satisfy.

<e.g.> FORALL SX (SX.STATUS > 0)

- If an integrity constraint is violated, either

<1> Reject, or

<2> Perform **compensating action** to ensure the correctness.

- Language for specifying integrity constraints should include

<1> the ability to specify arbitrary **conditions**.

<2> the ability to specify **compensating actions**.

```
CREATE INTEGRITY RULE R1
ON INSERT S.STATUS,
UPDATE S.STATUS;
CHECK FORALL S ( S.STATUS > 0 )
ELSE REJECT;
```

Types of Integrity Constraints

- Domain Constraints

values of an attribute are required to belong to a pool of legal values (domain).

<e.g.> S.STATUS > 0

SP.QTY > 0

- Primary and Foreign key constraints

<e.g.> S.S# must be unique

SP.S# must be contained in S.S#.

- FD, MVD, JD

<e.g.> S.S# => S.CITY

$\forall SX \forall SY (\text{IF } SX.S\# = SY.S\# \text{ THEN } SX.CITY = SY.CITY)$

- Format Constraints

<e.g.> ID number: A999999999

- Range Constraints

<e.g.> SALARY in (10,000 ~ 100,000).

A Hypothetical Integrity Language

- Two statements

<1> CREATE INTEGRITY RULE

<2> DROP INTEGRITY RULE

- <e.g.1> STATUS values must be positive:

```
CREATE INTEGRITY RULE R1
ON INSERT S.STATUS,
UPDATE S.STATUS;
CHECK FORALL S ( S.STATUS > 0 )
ELSE REJECT;
```

- Rule name: R1.
- Checking times: ON INSERT S.STATUS, UPDATE S.STATUS.
- Constraint: FORALL S (S.STATUS > 0)
- Violation Response: REJECT.

- Default

```
CREATE INTEGRITY RULE R1
CHECK S.STATUS > 0;
```

A Hypothetical Integrity Language (cont.)

- When a CREATE INTEGRITY RULE statement is executed:
 - (1) the system check if the current database state satisfied the specified constraint,
 - YES => accept the rule
 - NO => reject the rule
 - (2) the accepted rule is saved in system catalog,
 - (3) DBMS monitor all operations at the specified checking times.
- The integrity rule can be dropped
`DROP INTEGRITY RULE R1;`

A Hypothetical Integrity Language (cont.)

- <e.g.2> [The constraints can be arbitrary complex]

Assume that **SP** includes MONTH, DAY, YEAR, each is CHAR(2), representing the date of the shipment.

```
CREATE INTEGRITY RULE R2
  CHECK IS_INTEGER (SP.YEAR)
  AND IS_INTEGER (SP.MONTH)
  AND IS_INTEGER (SP.DAY)
  AND NUM (SP.YEAR) BETWEEN 0 AND 99
  AND NUM (SP.MONTH) BETWEEN 1 AND 12
  AND NUM (SP.DAY) > 0
  AND IF NUM (SP.MONTH) IN (1,3,5,7,8,10,12)
    THEN NUM (SP.DAY) < 32
  AND IF NUM (SP.MONTH) IN (4,6,9,11)
    THEN NUM (SP.DAY) < 31
  AND IF NUM (SP.MONTH) = 2
    THEN NUM (SP.DAY) < 30
  AND IF NUM (SP.MONTH) = 2
  AND NUM (SP.DAY) <> 0
  AND MOD (NUM(SP.YEAR),4) = 0
    THEN NUM (SP.DAY) < 29
```

A Hypothetical Integrity Language (cont.)

- <e.g.3> Status values must never decrease

```
CREATE INTEGRITY RULE R3
  BEFORE UPDATE OF S.STATUS FROM NEW_STATUS:
  CHECK NEW_STATUS > S.STATUS;
```

- <e.g.4> The average supplier must supply greater than 25

```
CREATE INTEGRITY RULE R4
  CHECK IF EXISTS S( ) THEN AVG(S.STATUS) > 25
```

- <e.g.5> Every London supplier must supply part p2

```
CREATE INTEGRITY RULE R5
  AT COMMIT :
  CHECK IF S.CITY = 'London' THEN
    EXISTS SP (SP.S# = S.S# AND SP.P# = 'P2')
  ELSE ROLLBACK;
```

Note: the constraint must be checked at commit time, otherwise, it's never possible to INSERT a new S record for a supplier in 'London'.

A Hypothetical Integrity Language (cont.)

- <e.g.6> Field S# is the primary key for S.

```
CREATE INTEGRITY RULE R6
  BEFORE INSERT OF S FROM NEW_S,
    UPDATE OF S.S# FROM NEW_S.S# :
  CHECK NOT (IS_NULL(NEW_S.S#))
  AND NOT EXISTS SX ( SX.S# =NEW_S.S#)
```

Note: The syntax in SQL: PRIMARY KEY(S#) is a much better alternative.

```
CREATE TABLE S
  ( S# CHAR(20)
    :
    :
    PRIMARY KEY (S#));
```

A Hypothetical Integrity Language (cont.)

- <e.g.7> S# is a foreign key in SP, matching the primary key of S.

```
CREATE INTEGRITY RULE R7A  
  BEFORE INSERT OF SP, UPDATE OF SP.S# :  
  CHECK EXISTS S (S.S#=SP.S#);
```

```
CREATE INTEGRITY RULE R7B  
  BEFORE DELETE OF S, UPDATE OF S.S# :  
  CHECK NOT EXISTS SP (SP.S#=S.S#);
```

||

The foreign key rule: DELETE OF S RESTRICTED
UPDATE OF S.S# RESTRICTED

A Hypothetical Integrity Language (cont.)

- The CASCADE version of foreign key rule :

```
DELETE OF S CASCADES  
UPDATE OF S.S# CASCADES
```

can be represented as :

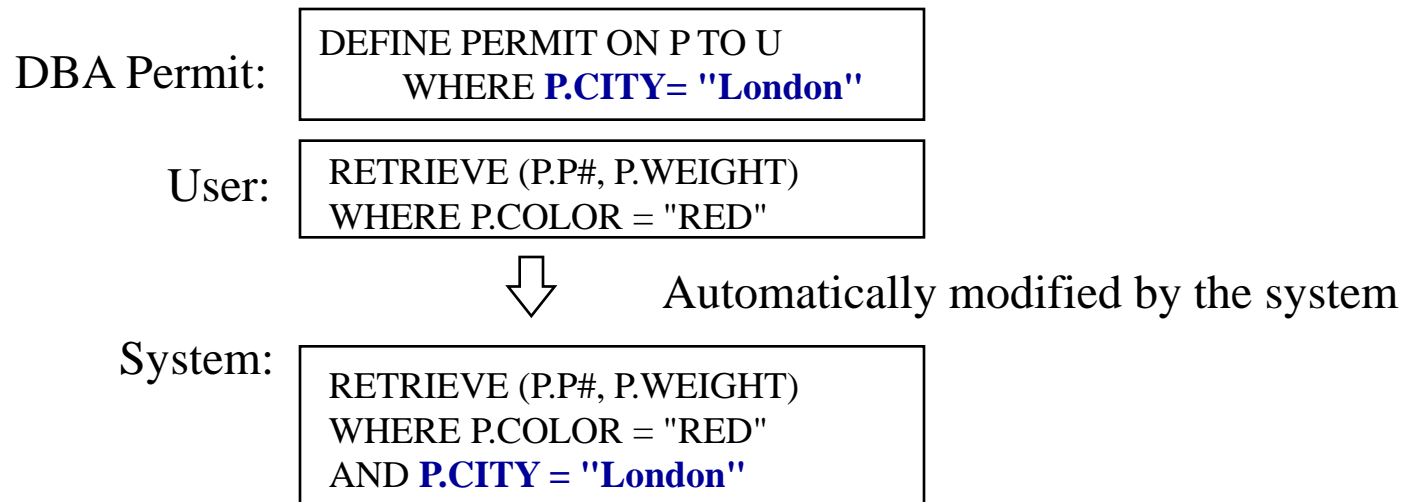
```
CREATE INTEGRITY RULE R7C  
  BEFORE DELETE OF S :  
    CHECK NOT EXISTS SP(SP.S#=S.S#)  
    ELSE DELETE SP WHERE SP.S# =S.S#;  
  
CREATE INTEGRITY RULE R7D  
  BEFORE UPDATE OF S.S# FROM NEW_S.S#  
    CHECK NOT EXISTS SP(SP.S#=S.S#)  
    ELSE UPDATE SP.S# FROM NEW_S.S#  
      WHERE SP.S#=S.S#;
```

<Note> The foreign key rule is more recommended.

10.4 Security and Integrity in INGRES

Query Modification in INGRES

- <e.g.> Suppose an user U is allowed to see parts stored in **London** only:



- The modification process is silent. (The user is not informed that there are other parts not located in London)
 - Advantages
 - easy to implement (same as **view**).
 - comparatively efficient (security overhead occurs at query **interpretation time** rather than **execution time**).
-

Security Constraint in INGRES

- Syntax:

```
DEFINE PERMIT operations
  ON table [(field-
    commalist)]
  TO user
  [ AT terminal(s) ]
  [ FROM time TO time2 ]
  [ ON day1 TO day2 ]
  [ WHERE condition ]
```

<e.g.>

```
DEFINE PERMIT RETRIEVE, REPLACE
  ON S (SNAME, CITY)
  TO Joe
  AT TTA4
  FROM 9:00 TO 17:30
  ON SAT TO SUN
  WHERE S.STATUS < 50
    AND S.S# = SP.P#
    AND SP.P# = P.P#
    AND P.COLOR = "RED"
```

- Constraints are kept in INGRES catalog.
- The constraint identifier can be discover by querying the catalog.
- To delete a constraint
<e.g.> DESTROY PERMIT S 27;

Integrity Constraint in INGRES

■ Syntax :

```
DEFINE INTEGRITY
  ON  table
  IS  condition
```

<e.g.>

```
DBA:      DEFINE INTEGRITY
           ON  S
           IS  S.STATUS > 0
```

Suppose an user issues:

```
User:     REPLACE S (STATUS=S.STATUS-10)
           WHERE S.CITY= "London"
```



Automatically modified by system

```
System:   REPLACE S (STATUS=S.STATUS-10)
           WHERE  S.CITY= "London"
           AND  (S.STATUS-10) > 0
```

■ Note

- Destroy an integrity constraint: <e.g.> DESTROY INTEGRITY S 18
- The modification is silent too.
- The integrity constraints are kept in catalog.
- Advantages and disadvantages are similar to security constraint.

10.5 Security in Statistical Databases

Ref. p.10-11

Statistical Database

Def: Statistical Database is a database, such as a census database, that

(a) contains a large number of individually sensitive records .

(b) is intended to supply **only** statistical summary information to its users, not information to some specific individual.

- only queries that apply some **statistical function**.
 - E.g: **count**, **sum**, or **average**
- Problem:
 - “Deduction of confidential information by inference is possible”

Statistical Database: An Example

- e.g.1 Consider the **STATS** database

Name	Sex	Dependence	Occupation	Salary	Tax	Audits
Able	M	3	programmer	25k	5k	3
Baker	F	2	physician	65k	5k	0
Clark	F	0	programmer	28k	9k	1
Downs	F	2	builder	30k	6k	1
East	M	2	clerk	22k	2k	0
Ford	F	1	homemaker	51k	0k	0
Green	M	0	lawyer	95k	0k	0
Hall	M	3	homemaker	22k	1k	0
Lves	F	4	programmer	32k	5k	1
Jones	F	1	programmer	30k	10k	1

Fig. The **STATS** database

- Suppose some user **U** is intent on discovering **Able's** salary and tax payment.
- Suppose **U** knows that **Able** is a programmer and is male

Statistical Database: Case 1

危害, 信用傷害

- The security of the database has been compromised, even though U has issued only legitimate statistical queries (**count**, **sum**, or **average**.)

- **Case 1:** Q1 : SELECT COUNT(*)
FROM STATS
WHERE SEX = 'M'
AND OCCUPATION = 'Programmer'

Response :

1

- Q2 : SELECT SUM (SALARY), SUM (TAX)
FROM STATS
WHERE SEX = M
AND OCCUPATION = 'Programmer'

5k

Solution ↓

Response: 25k,

The **system** should refuse to response to a query for which the cardinality of the identified subset of records < lower bound **b**
e.g. **b** = 2 i.e., **b** ≤ **c** (result set cardinality)

Statistical Database: Case 2

Case 2: Consider the sequence of queries Q3-Q6 below

Q3 : SELECT COUNT(*)
FROM STATS

Response : 10

Q4 : SELECT COUNT(*)
FROM STATS
WHERE NOT
(SEX = 'M' AND
OCCUPATION = 'Programmer')

Response: 9
Subtract
(Q3 - Q4): 1

Q5 : SELECT SUM(SALARY), SUM(TAX)
FROM STATS

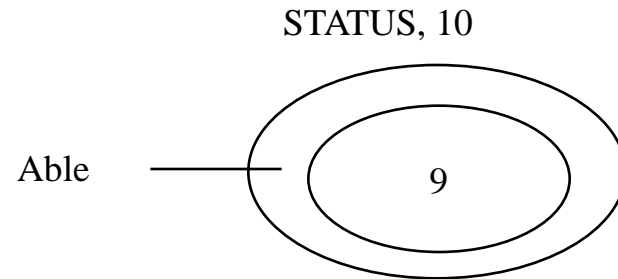
Response : 364k, 43k

Q6 : SELECT SUM(SALARY), SUM(TAX)
FROM STATS
WHERE NOT
(SEX = 'M' AND
OCCUPATION = 'Programmer')

Response 339k, 38k
Subtract
(Q5 - Q6): 25k, 5k

Statistical Database: Case 2 (cont.)

- Why ?



- Solution

Let $b \leq c \leq n-b$ eg. $2 \leq c \leq 8$

Now predicate:

NOT (Sex = 'M' and Occupation = 'programmer')

is thus not admissible.

Statistical Database: Case 3

Case 3: Set C in the range $b \leq c \leq n-b$ eg. $2 \leq c \leq 8$ is inadequate to avoid compromise, in general.

Consider the following sequence (Q7-Q10):

```
Q7 : SELECT COUNT (*)  
      FROM STATS  
      WHERE SEX = 'M'
```

Response: 4

```
Q8 : SELECT COUNT (*)  
      FROM STATS  
      WHERE SEX = 'M'  
      AND NOT ( OCCUPATION = 'Programmer')
```

Response: 3

```
Q9 : SELECT SUM(SALARY), SUM(TAX)  
      FROM STATS  
      WHERE SEX = 'M'
```

Response: 164k, 8k

```
Q10 : SELECT SUM(SALARY), SUM(TAX)  
        FROM STATS  
        WHERE SEX = M AND NOT  
              (OCCUPATION = 'Programmer')
```

Response: 139k, 3k

Subtract: (Q9 - Q10): 25K, 5K

Statistical Database: Case 3 (cont.)

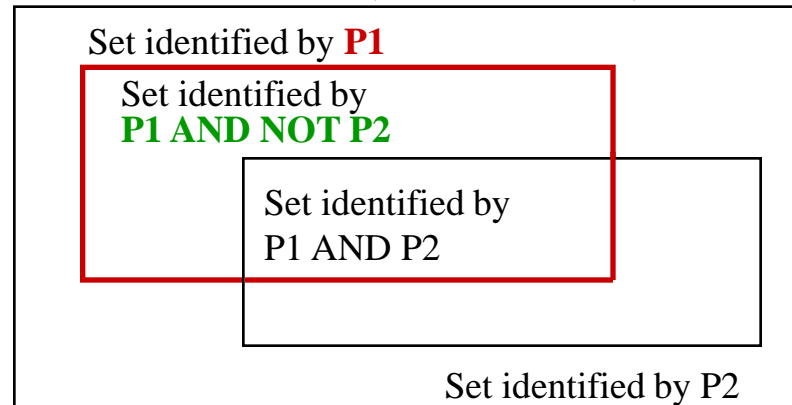
- Why ?

P: SEX = M and OCC.= 'Programmer'

P1: SEX = M

P2: OCC. = 'Programmer'

Total set of records (entire database)



- Solution ???

Statistical Database: Tracker

- Individual tracker

- for some specific inadmissible predicate
- individual tracker, the predicate

$SEX = 'M'$ and NOT (OCCUPATION = 'Programmer')

is called an **individual tracker** for Able, because it enables the user to track down information concerning Able.

- General tracker

- is a predicate that can be used to find the answer to any inadmissible query.
- **Note** : Any predicate with result set cardinality c in the range

$$2b \leq c \leq n-2b$$

where $b < n/4$ (typically case), is a general tracker.

Statistical Database: Case 4

Case 4:

- Assume
 1. $b=2$ ie. $4 \leq c \leq 6$
 2. **U** knows that **Able** is a male programmer.

Predicate P is **SEX = 'M' and OCCUPATION = 'programmer'**
 3. **U** wishes to discover **Able's** salary.
- Compromise steps:
 1. Make a guess at a predicate T that will serve as a general tracker,
T: Audits = 0
 2. Find total number of individuals in the **STATS**, using T and Not T.
 3. Find the number by using P or T; P or NOT T
 4. Repeat Q11 - Q14, but using SUM instead of COUNT
 5. Able's salary: 389k- 364k = 25k

Statistical Database: An Example

- e.g.1 Consider the **STATS** database

Name	Sex	Dependence	Occupation	Salary	Tax	Audits
Able	M	3	programmer	25k	5k	3
Baker	F	2	physician	65k	5k	0
Clark	F	0	programmer	28k	9k	1
Downs	F	2	builder	30k	6k	1
East	M	2	clerk	22k	2k	0
Ford	F	1	homemaker	51k	0k	0
Green	M	0	lawyer	95k	0k	0
Hall	M	3	homemaker	22k	1k	0
Lves	F	4	programmer	32k	5k	1
Jones	F	1	programmer	30k	10k	1

Fig. The **STATS** database

- Suppose some user **U** is intent on discovering **Able**'s salary and tax payment.
- Suppose **U** knows that **Able** is a programmer and is male

Statistical Database: Case 4 (cont.)

2. Find total number of individuals in the db, using T and Not T.

```
Q11 : SELECT COUNT (*)
      FROM STATS
      WHERE AUDITS = 0
```

Response : 5

```
Q12 : SELECT COUNT (*)
      FROM STATS
      WHERE NOT
```

```
(AUDITS = 0)
```

Response : 5

Add: (Q11 + Q12) : 10

$$4 \leq C = 5 \leq 6$$

as a result, T is a general tracker

Entire database

T audits=0	NOT T
---------------	-------

5

5

Statistical Database: Case 4 (cont.)

3. Find the number by using P or T; P or NOT T :

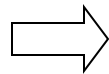
```
Q13 : SELECT COUNT (*)
      FROM STATS
      WHERE (SEX = 'M' AND
            OCCUPATION = 'Programmer')
      OR AUDITS = 0
```

Response : 6

```
Q14 : SELECT COUNT (*)
      FROM STATS
      WHERE (SEX = 'M' AND
            OCCUPATION = 'Programmer')
      OR NOT
      (AUDITS = 0)
```

Response : 5
Add
(Q13 + Q14): 11

T	P	NOT T
	0	1



from the results we have that the number of individuals satisfying P is one; i.e., P designates Able uniquely.

Statistical Database: Case 4 (cont.)

4. Repeat Q11 - Q14, but using SUM instead of COUNT.

T or not T	{	Q15 : SELECT SUM (SALARY) FROM STATS WHERE AUDITS = 0	Response : 219K
		Q16 : SELECT SUM (SALARY) FROM STATS WHERE NOT (AUDITS = 0)	Response : 145K Add (Q15 + Q16) : 364K
P or T	{	Q17 : SELECT SUM (SALARY) FROM STATS WHERE (SEX = 'M' AND OCCUPATION = 'Programmer') OR AUDITS = 0	Response : 244K
P or not T	{	Q18 : SELECT SUM (SALARY) FROM STATS WHERE (SEX = 'M' AND OCCUPATION = 'Programmer') OR NOT (AUDITS = 0)	Response : 145K Add (Q17 + Q18) : 389K

5. Able's salary: 389k- 364k = 25k

Statistical Database: General Tracker

- The general tracker T

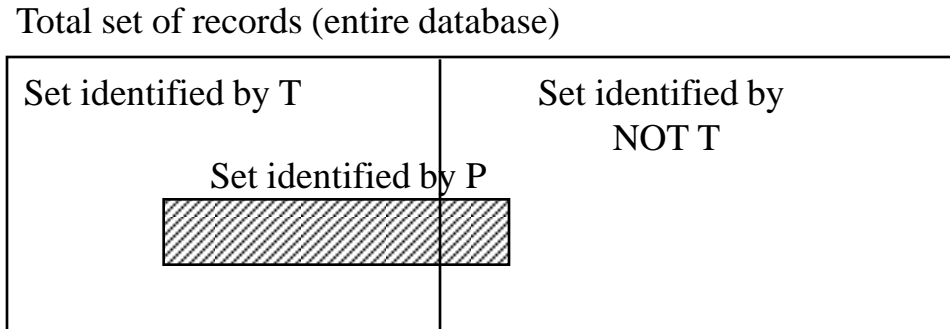


Fig. The general tracker T :

$$\text{SET}(P) = \text{SET}(P \text{ OR } T) + \text{SET}(P \text{ OR } \text{NOT } T) - \text{SET}(T \text{ OR } \text{NOT } T)$$

- Summary

- TODS, 4.1. D.E. Denning, et.al 79, “The Tracker: A threat to statistical database”
 - A general tracker *almost always* exists, and is usually both easy to find and easy to use.
 - Can be found by simple queries.
- Security in a statistical database is a **REAL PROBLEM!!**

10.6 Data Encryption

Data Encryption: Basic Idea

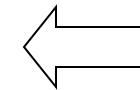
■ Infiltrator: 滲透者

1. Using the normal system facilities for accessing the database

- deduction information (statistical database)
- authorization matrix

2. Bypass the system

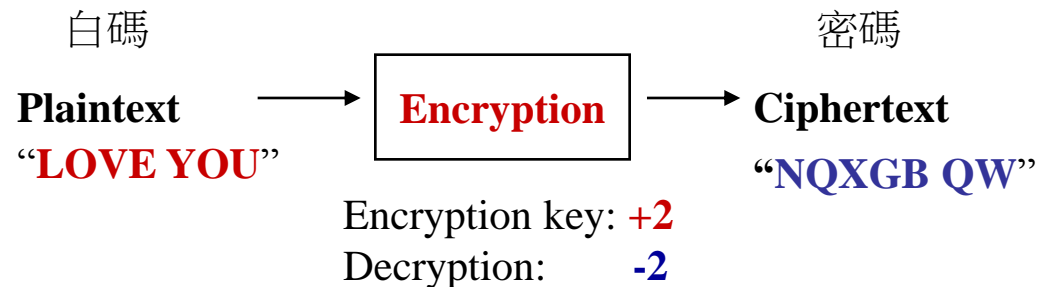
- stealing a disk park
- tapping into a communication line
- breaks through O.S.



Data Encryption

$$f(x) = 2x + 1$$

$$231 \rightarrow 463$$



Data Encryption: Basic Idea (cont.)

- e.g.2

WE NEED MORE SNOW
key = +2
→ ZG PGGF OQTG UPZY

Problem: How difficult is it for a would-be infiltrator to determine the key without prior knowledge ?

Answer: Fairly obviously, "not very" ; but equally obviously.

- e.g.3

WE NEED MORE SNOW
+ 23 1579 2315 7923
—
YH OJLM PRSJ ZWQZ

key = 231579

最早有關密碼的書 (1920)



World War II

Enigma (德國)



Big machine (美國)



Three-Rotor Machine

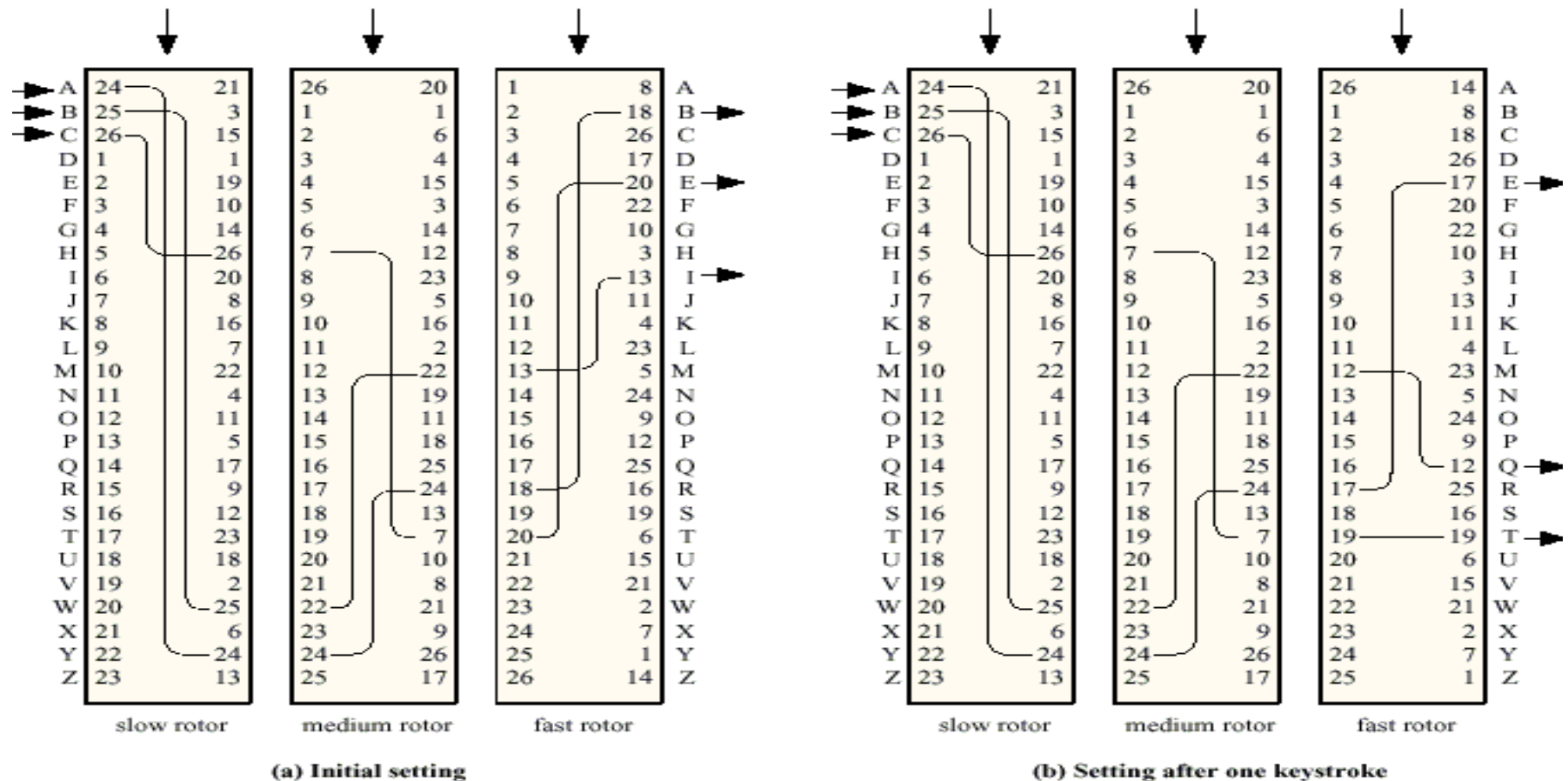


Figure 2.8 Three-Rotor Machine With Wiring Represented by Numbered Contacts

新時代

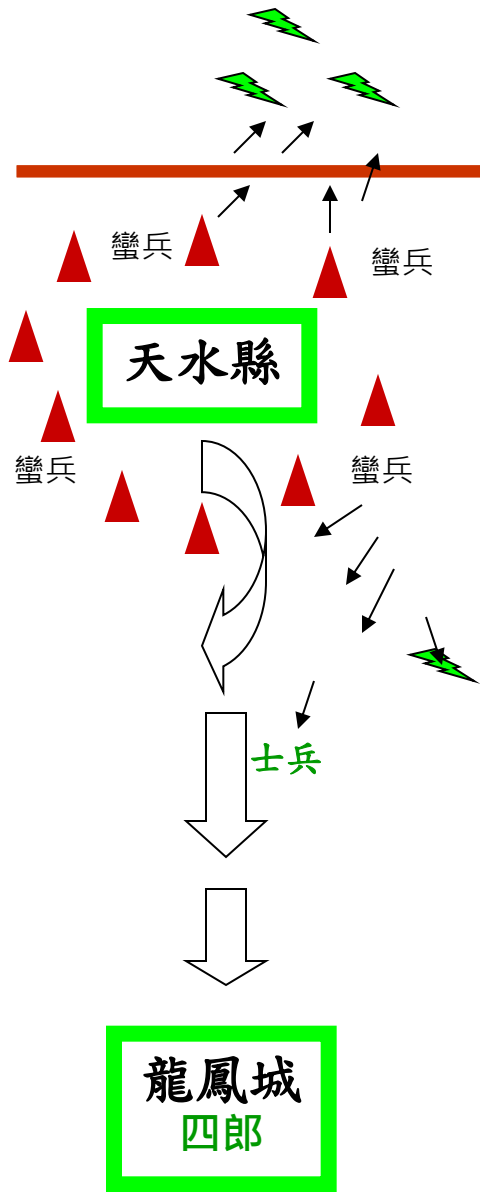
特殊IC



Supercomputer: Cray-XMP



諸葛四郎大門雙假面



四郎

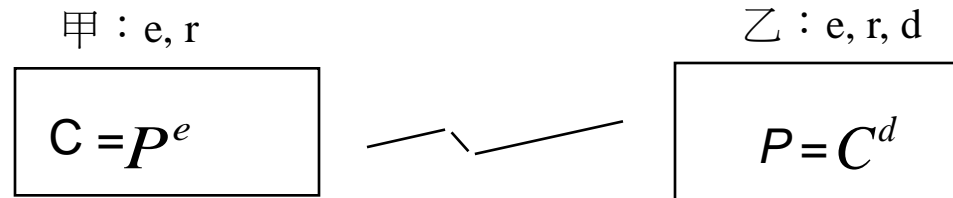
Public Encryption: RSA 公開金鑰

- e.g.

Plaintext $P = 13$; public-key: $e=11, r=15$

$$\begin{aligned}\text{Ciphertext } C &= P^e \text{ modulo } r \\ &= 13^{11} \text{ modulo } 15 \\ &= 1792160394037 \text{ modulo } 15 \\ &= 7\end{aligned}$$

$$\begin{aligned}\text{Decryption } P &= C^d \text{ modulo } r \\ &= 7^3 \text{ modulo } 15 \\ &= 343 \text{ modulo } 15 \\ &= 13\end{aligned}$$



Public Encryption: RSA 公開金鑰 (cont.)

- The scheme of : [Rivest78]

1. Choose, randomly, two distinct large primes p and q , and compute the product

$$r = p * q \quad \text{e.g. } p=3, q=5, r=15$$

2. Choose, randomly, a large integer e , such that

$$\text{gcd}(e, (p-1)*(q-1)) = 1$$

Note: any prime number greater than both p and q will do.

$$(p-1)*(q-1) = 2*4 = 8, \quad e = 11$$

3. Take the decryption key, d , corresponding to e to be the unique "multiplicative inverse" of e , modulo $(p-1)*(q-1)$;

$$\text{i.e. } d*e = 1, \text{ modulo } (p-1)*(q-1)$$

Note: The algorithm for computing d is straight forward.

$$d*11=1, \text{ mod } 8, \implies d=3$$

Public Encryption: RSA 公開金鑰 (cont.)

- Exercise: Suppose we have $r = 2773$, $e = 17$, try to find $d = ?$

Answer:

1. $50 * 50 = 2500 \sim 2773$

2. $47 * 53 \neq 2773$

3. $47 * 59 = 2773$ so $p = 47$, $q = 59$

4. $(p-1)(q-1) = 2668$

5. $d * 17 = 1$, modulo $(p-1) * (q-1)$

$\Rightarrow d * 17 = 1$, modulo 2668

$d = 1: \quad 17 + 2667 \quad 2668 \quad x$

$d = 2: \quad 4 + 2667 \quad 2668 \quad x$

\vdots

$d = 157: \quad (157 * 17) / (2668) = 2$

r: 50 digits \Rightarrow 4 hrs
 75 digits \Rightarrow 100 days
 100 digits \Rightarrow 74 years
 500 digits \Rightarrow $4 * 10^{25}$ years

"Signed" Ciphertext

Algorithm ENCRYPT_FOR_A

- for encryption message to be sent to A

Algorithm DECRYPT_FOR_A

- inverse of ENCRYPT_FOR_A

Algorithm ENCRYPT_FOR_B

- for encrypting message to be sent to B

Algorithm DECRYPT_FOR_B

- ...

[A do] 1. $P' = \text{ENCRYPT_FOR_B}(\text{DECRYPT_FOR_A}(P))$

[sent] 2. Sent P' to B

[B do] 3. $\text{ENCRYPT_FOR_A}(\text{DECRYPT_FOR_B}(P')) \Rightarrow P$

