CS 3002 Information Security

Fall 2023

1. Explain key concepts of information security such as design principles, cryptography, risk management,(1)

2. Discuss legal, ethical, and professional issues in information security (6)
3. Analyze real world scenarios, model them using security measures, and apply various security and risk management tools for achieving information security and privacy (2)
4. Identify appropriate techniques to tackle and solve problems of real life in the discipline of information security (3)
5. Understand issues related to ethics in the field of information security(8)

ISO/IEC 27001: 2013

Week # 11

Dr. Nadeem Kafi Khan

5.6 INFERENCE

- Inference, as it relates to database security, is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- The inference problem arises when the combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of higher sensitivity.
- Figure 5.7 illustrates the process. The attacker may make use of non sensitive data as well as metadata. The information transfer path by which unauthorized data is obtained is referred to as an **inference channel**.
- Two inference techniques can be used to derive additional information:
  - Analyzing functional dependencies between attributes within a table or across tables, and
  - Merging views with the same constraints.

Metadata refers to knowledge about correlations or dependencies among data items that can be used to deduce information not otherwise available to a particular user.

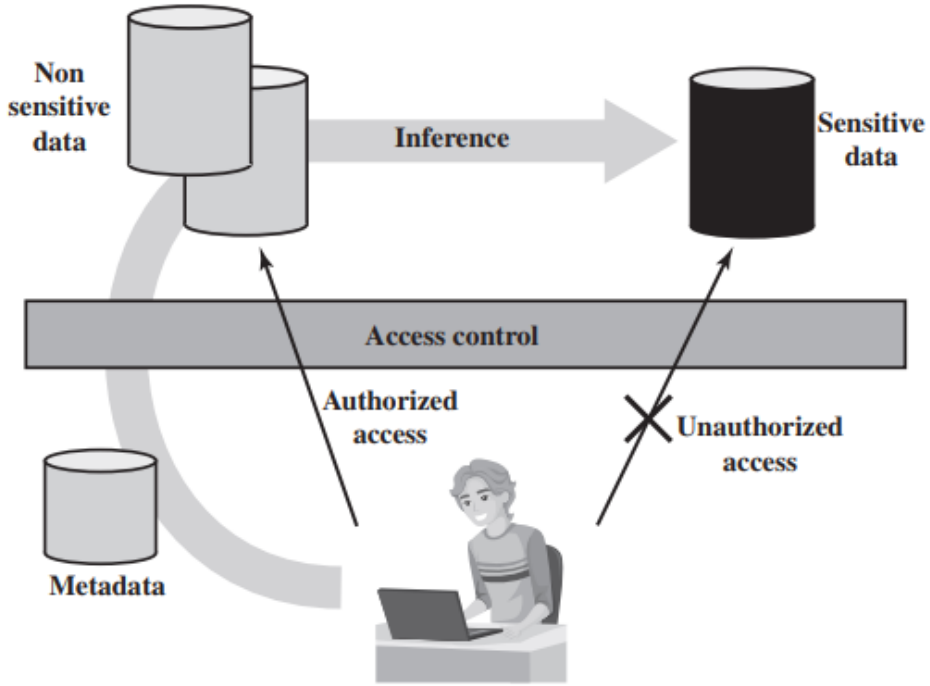


Figure 5.7 **Indirect Information Access via Inference Channel**

5.6 INFERENCE

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

```
CREATE view V1 AS
SELECT Availability, Cost
FROM Inventory
WHERE Department = "hardware"
```

```
CREATE view V2 AS
SELECT Item, Department
FROM Inventory
WHERE Department = "hardware"
```

Availability	Cost ($)
in-store/online	7.99
online only	5.49
in-store/online	104.99

Item	Department
Shelf support	hardware
Lid support	hardware
Decorative chain	hardware

(b) Two views

A user who knows the structure of the Inventory table and who knows that the view tables maintain the same row order as the Inventory table is then able to merge the two views to construct the table shown in Figure 5.8c. This violates the access control policy that the relationship of attributes Item and Cost must not be disclosed.

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

Access to this data is a violation.

(c) Table derived from combining query answers

Figure 5.8 Inference Example

5.6 INFERENCE

How to dead with the threat of disclosure by inference?

In general terms, there are two approaches to dealing with the threat of disclosure by inference:

- **Inference detection during database design:** This approach removes an inference channel by altering the database structure or by changing the access control regime to prevent inference. Examples include removing data dependencies by splitting a table into multiple tables or using more fine-grained access control roles in an RBAC scheme. Techniques in this category often result in unnecessarily stricter access controls that reduce availability.
- **Inference detection at query time:** This approach seeks to eliminate an inference channel violation during a query or series of queries. If an inference channel is detected, the query is denied or altered.

Consider a database containing personnel information, including names, addresses, and salaries of employees. Individually, the name, address, and salary information is available to a subordinate role, such as Clerk, but the association of names and salaries is restricted to a superior role, such as Administrator.

Create a small database schema to hold the above information. How we safeguard against inference?

Figure 5.8. One solution to this problem is to construct three tables, which include the following information:

Clerk sees these tables

Employees (Emp#, Name, Address)

Salaries (S#, Salary)

Emp-Salary (Emp#, S#)

where each line consists of the table name followed by a list of column names for that table. In this case, each employee is assigned a unique employee number (Emp#) and a unique salary number (S#). The Employees table and the Salaries table are accessible to the Clerk role, but the Emp-Salary table is only available to the Administrator role. In this structure, the sensitive relationship between employees and salaries is protected from users assigned the Clerk role.

Consider a database containing personnel information, including names, addresses, and salaries of employees. Individually, the name, address, and salary information is available to a subordinate role, such as Clerk, but the association of names and salaries is restricted to a superior role, such as Administrator.

In which table a new attribute, employee start date could be added if it is not sensitive?

Now, suppose we want to add a new attribute, employee start date, which is not sensitive. This could be added to the Salaries table as follows:

Employees (Emp#, Name, Address)

Salaries (S#, Salary, Start-Date)

Emp-Salary (Emp#, S#)

should be added in Employees table.

However, an employee's start date is an easily observable or discoverable attribute of an employee. Thus, a user in the Clerk role should be able to infer (or partially infer) the employee's name. This would compromise the relationship between employee and salary. A straightforward way to remove the inference channel is to add the start-date column to the Employees table rather than to the Salaries table.

- Database Encryption (Section 5.7)
  - Actors in a Database Encryption Scheme
- Whole Database Encryption Scheme (Figure 5.9)
- Row Encryption Scheme (Figure 5.10, Table 5.3)
  - How row-wise encryption scheme works?
  - Performance and other improvements

5.7 DATABASE ENCRYPTION

Encryption becomes the last line of defense in database security.

There are two disadvantages to database encryption:

- **Key management:** Authorized users must have access to the decryption key for the data for which they have access. Because a database is typically accessible to a wide range of users and a number of applications, providing secure keys to selected parts of the database to authorized users and applications is a complex task.
- **Inflexibility:** When part or all of the database is encrypted, it becomes more difficult to perform record searching.

Encryption can be applied to the (i) entire database, (ii) at the record level (encrypt selected records), (iii) at the attribute level (encrypt selected columns), or at the level of the individual field.

Actors in a Database Encryption Scheme

- **Data owner:** An organization that produces data to be made available for controlled release, either within the organization or to external users.
- **User:** Human entity that presents requests (queries) to the system. The user could be an employee of the organization who is granted access to the database via the server, or a user external to the organization who, after authentication, is granted access.
- **Client:** Front end that transforms user queries into queries on the encrypted data stored on the server.
- **Server:** An organization that receives the encrypted data from a data owner and makes them available for distribution to clients. The server could in fact be owned by the data owner but, more typically, is a facility owned and maintained by an external provider.

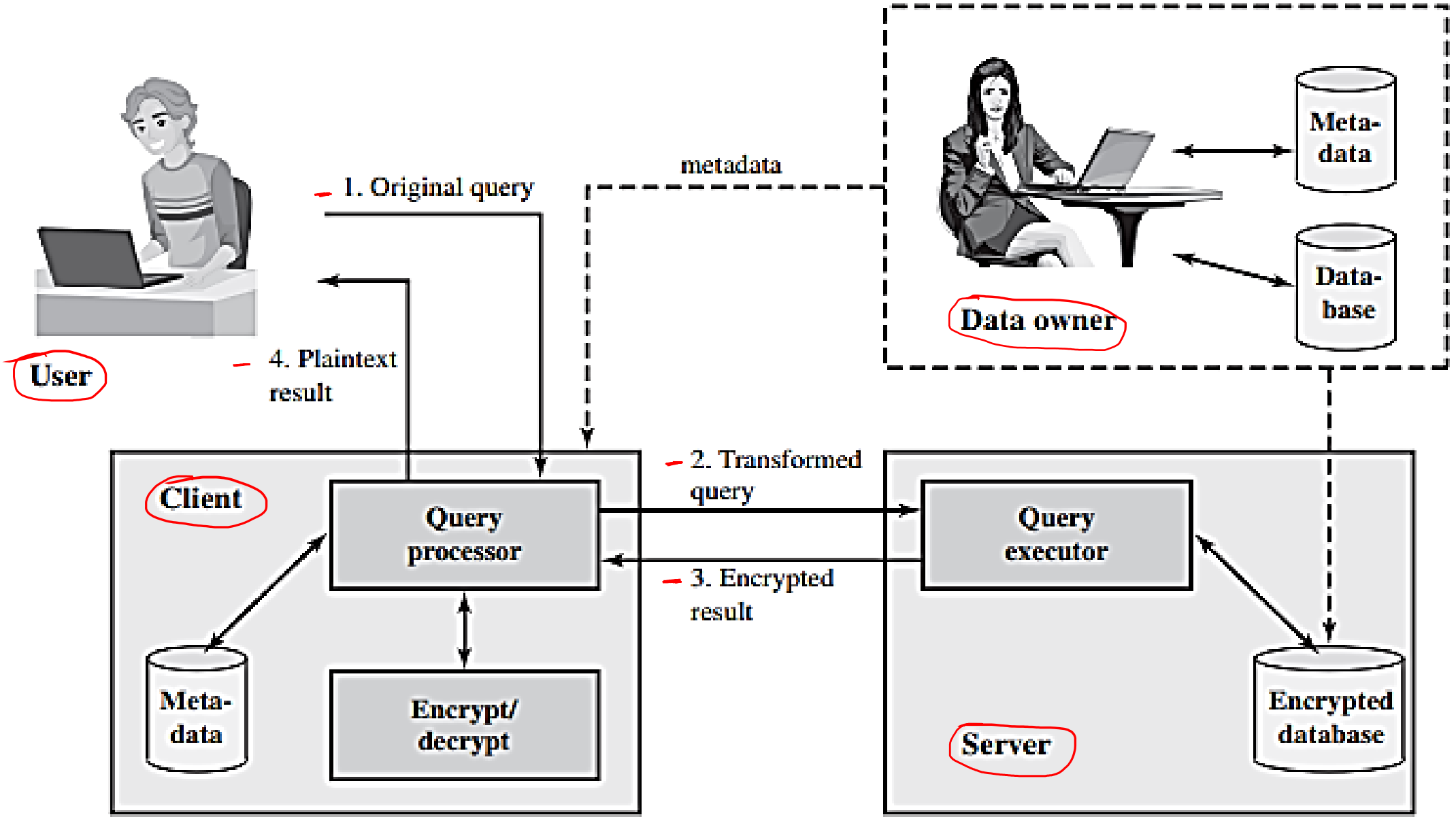


Figure 5.9 **A Database Encryption Scheme**

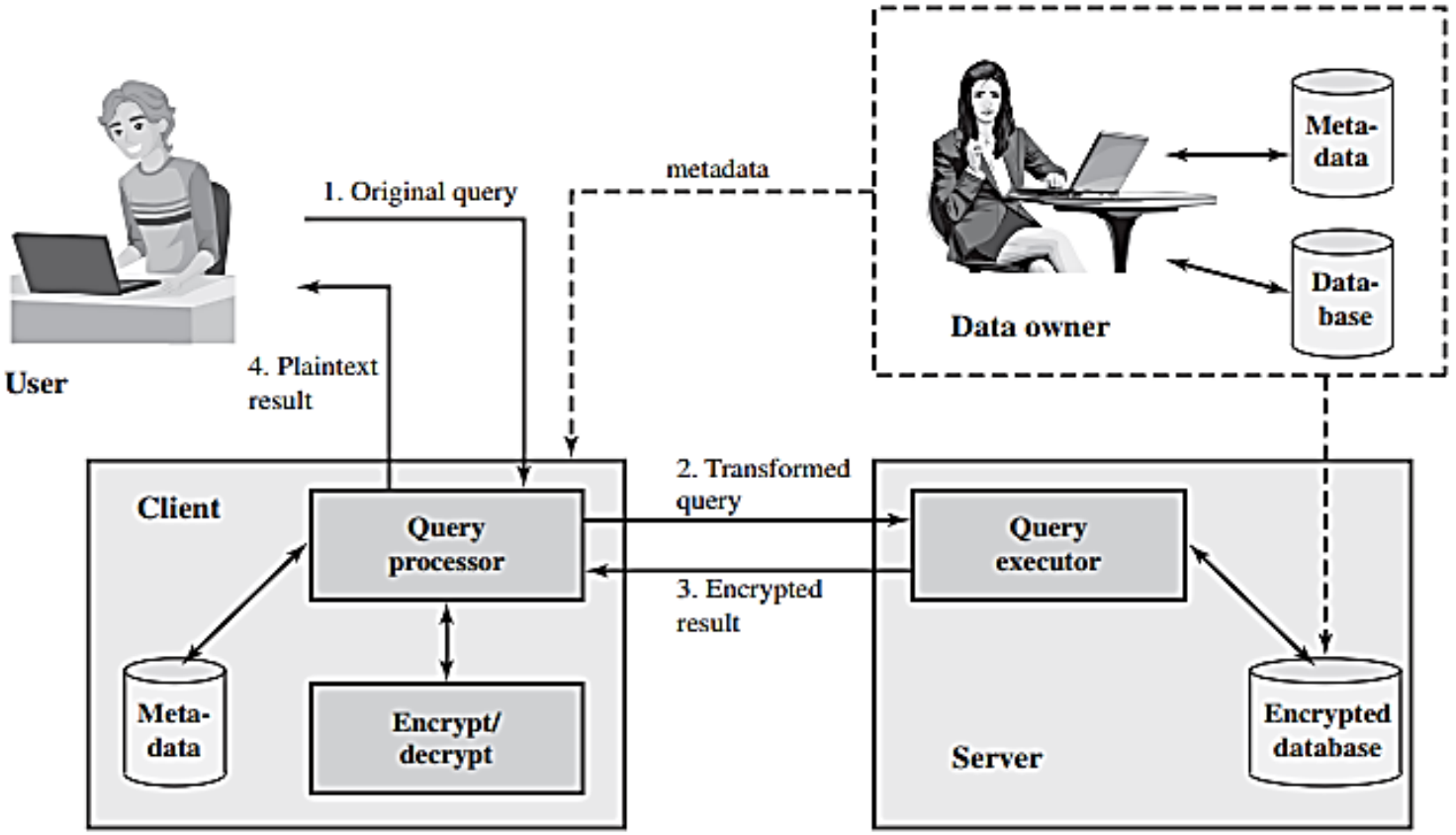


Figure 5.9 **A Database Encryption Scheme**

1. The user issues an SQL query for fields from one or more records with a specific value of the primary key.
2. The query processor at the client encrypts the primary key, modifies the SQL query accordingly, and transmits the query to the server.
3. The server processes the query using the encrypted value of the primary key and returns the appropriate record or records.
4. The query processor decrypts the data and returns the results.

1.
```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

2. Assume the encryption key *k* is used and the encrypted value of the department id 15 is $E(k, 15) = 1000110111001110$.

3.
```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 1000110111001110
```

4. If the user wishes to retrieve all records for salaries less than $70K. There is no obvious way to do this, because the attribute value for salary in each record is encrypted. The set of encrypted values do not preserve the ordering of values in the original attribute.

Alternate Approach (1)

- Each row Ri is treated as a contiguous block forming a sequence of bits, and all of the attribute values for that row are concatenated together to form a single binary block.
- The entire row is encrypted.

$$E(k, B_i) = E(k, (x_{i1} \| x_{i2} \| \ldots \| x_{iM}))$$

- Shown as I_{i1}, I_{i2}, I_{i3}, … are indexes are associated with each attribute of the table.

$$(x_{i1}, x_{i2}, \ldots, x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]$$

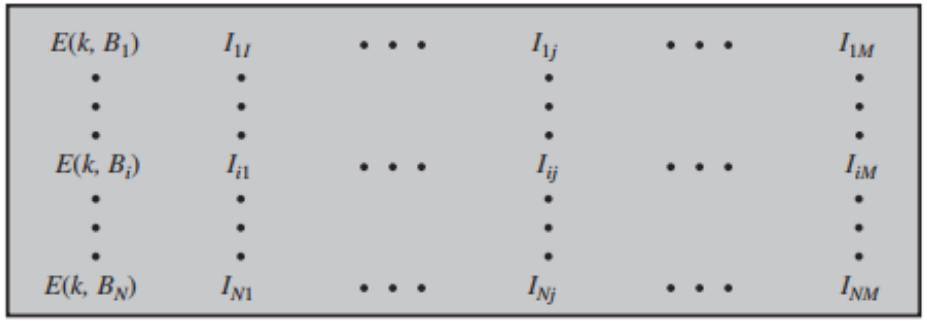


Figure 5.10 **Encryption Scheme for Database of Figure 5.3**

Table 5.3 **Encrypted Database Example**

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(k, B)	*I(eid)*	*I(ename)*	*I(salary)*	*I(addr)*	*I(did)*
1100110011001011 . . .	1	10	3	7	4
0111000111001010 . . .	5	7	2	7	8
1100010010001101 . . .	2	5	1	9	5
0011010011111101 . . .	5	5	2	4	9

Alternate Approach (2)

- Suppose employee ID (eid) values lie in the range [1, 1000].
- We can divide these values into five partitions: [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000]; then assign index values 1,2, 3, 4, and 5, respectively.
- 23 → partition 1
- 860 → partition 5
- 320 → partition 2
- 875 → partition 5

Table 5.3 **Encrypted Database Example**

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(k, B)	*I(eid)*	*I(ename)*	*I(salary)*	*I(addr)*	*I(did)*
1100110011001011 . . .	1	10	3	7	4
0111000111001010 . . .	5	7	2	7	8
1100010010001101 . . .	2	5	1	9	5
0011010011111101 . . .	5	5	2	4	9

Alternate Approach (3)

- This arrangement provides for more efficient data retrieval. Suppose, for example, a user requests records for all employees with eid < 300.
  - The query processor requests all records with I(eid) = 2.
  - These are returned by the server.
  - The query processor decrypts all rows returned, discards those that do not match the original query, and returns the requested unencrypted data to the user.

- The indexing scheme just described does provide a certain amount of information to an attacker, namely a rough relative ordering of rows by a given attribute. To obscure such information, the ordering of indexes can be randomized.
  - For example, the eid values could be partitioned by mapping [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000] into 2, 3, 5, 1, and 4, respectively. Because the metadata are not stored at the server, an attacker could not gain this information from the server.

Alternate Approach (4)

Retrieval Efficiency

- To increase the efficiency of accessing records by means of the primary key, the system could use the encrypted value of the primary key attribute values, or a hash value.
  - In either case, the row corresponding to the primary key value could be retrieved individually.

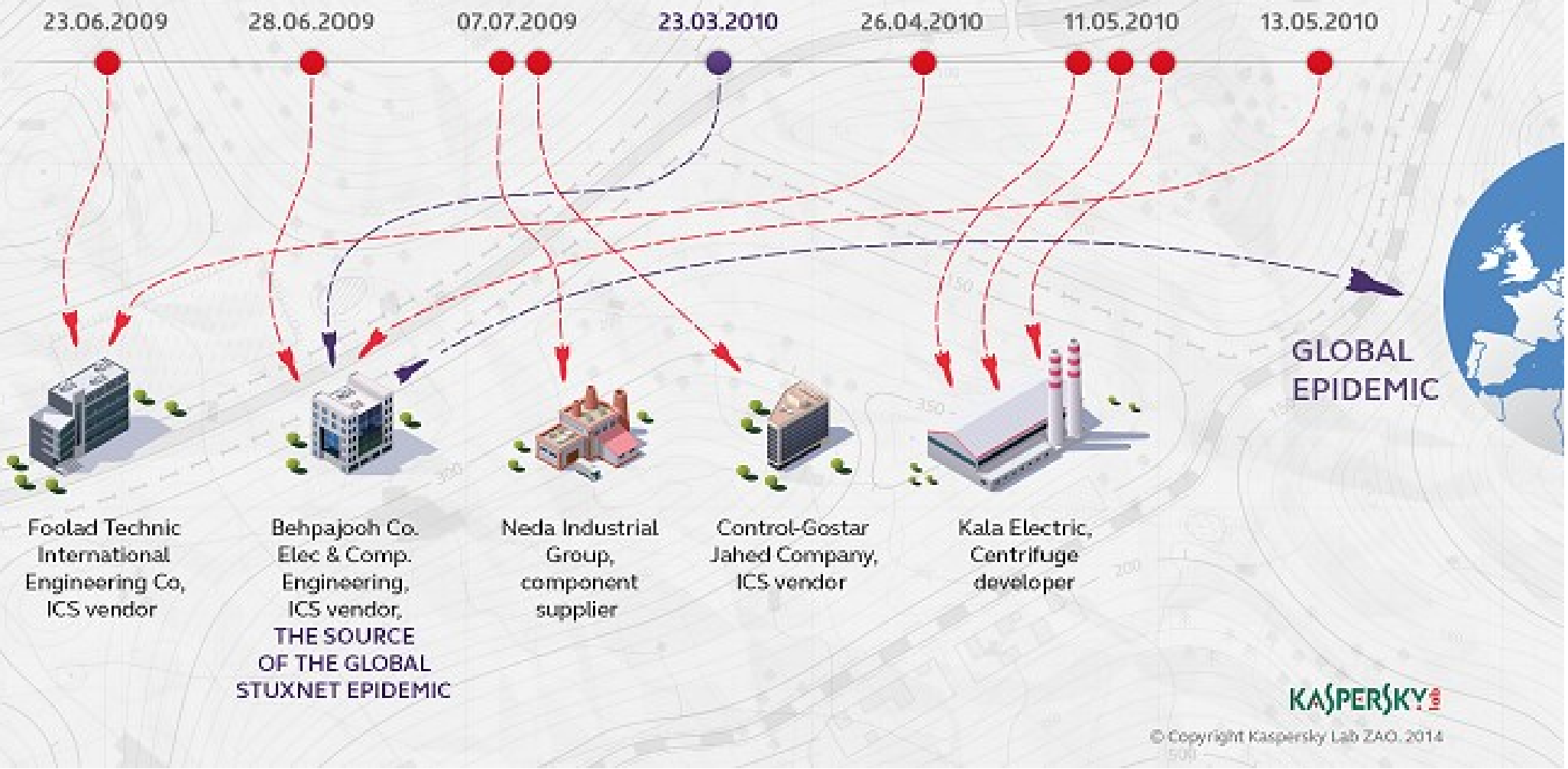
Increased Security

- Different portions of the database could be encrypted with different keys, so users would only have access to that portion of the database for which they had the decryption key.
  - This latter scheme could be incorporated into a role-based access control system.

Chapter #6

OUTBREAK: THE FIRST FIVE VICTIMS OF THE STUXNET WORM

The infamous Stuxnet worm was discovered in 2010, but had been active since at least 2009.
The attack started by infecting five carefully selected organizations



KASPERSKY

Advanced Persistent Threats (APTs)

Chapter #6

- Well-resourced, persistent application of a wide variety of intrusion technologies and malware to **selected targets** (usually business or political)
- Typically attributed to criminal enterprises
- Differ from other types of attack by their **careful target selection** and stealthy intrusion efforts over extended periods
- High profile attacks include Aurora, RSA, APT1, and **Stuxnet**

Characteristics of APT

Chapter #6

- Advanced
  - Used by the attackers of a wide variety of intrusion technologies and malware including the development of custom malware if required
  - The individual components may not necessarily be technically advanced but are carefully selected to suit the chosen target
- Persistent
  - Determined application of the attacks **over an extended period** against the chosen target in order to maximize the chance of success
  - A variety of attacks may be progressively applied until the target is compromised
- Threat
  - Threats to the selected targets as a result of the organized, capable, and well-funded attackers intent to compromise the specifically chosen targets
  - The active involvement of people in the process greatly raises the threat level from that due to automated attacks tools, and also the likelihood of successful attacks