

CHƯƠNG 5

MÃ HÓA

CƠ SỞ DỮ LIỆU



Bộ môn: Tin học quản lý
Khoa Thống kê – Tin học
Đại học Kinh Tế - Đại học Đà Nẵng



NỘI DUNG CHƯƠNG 5

1. Khái quát về mã hóa dữ liệu
2. Các mức độ mã hóa
3. Các mô hình mã hóa CSDL
4. Nhận xét về các giải pháp mã hóa
5. Một số vấn đề liên quan đến giải pháp mã hóa
6. Mô hình lưu trữ dữ liệu mã hóa
7. Hiện thực trên DMBS cụ thể (SQL Server)

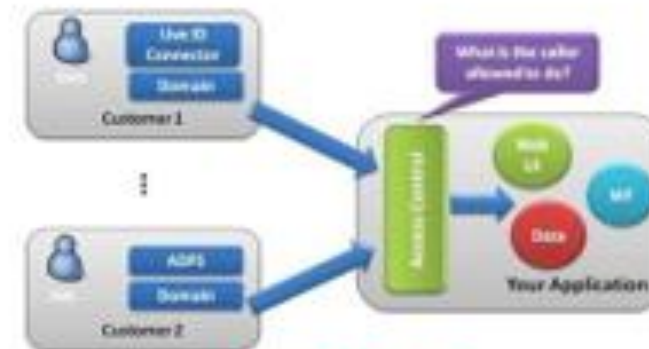
1. Khái quát về mã hóa dữ liệu



Identification
Định danh người dùng



Authentication
Xác thực người dùng



Access Control
Điều khiển truy cập



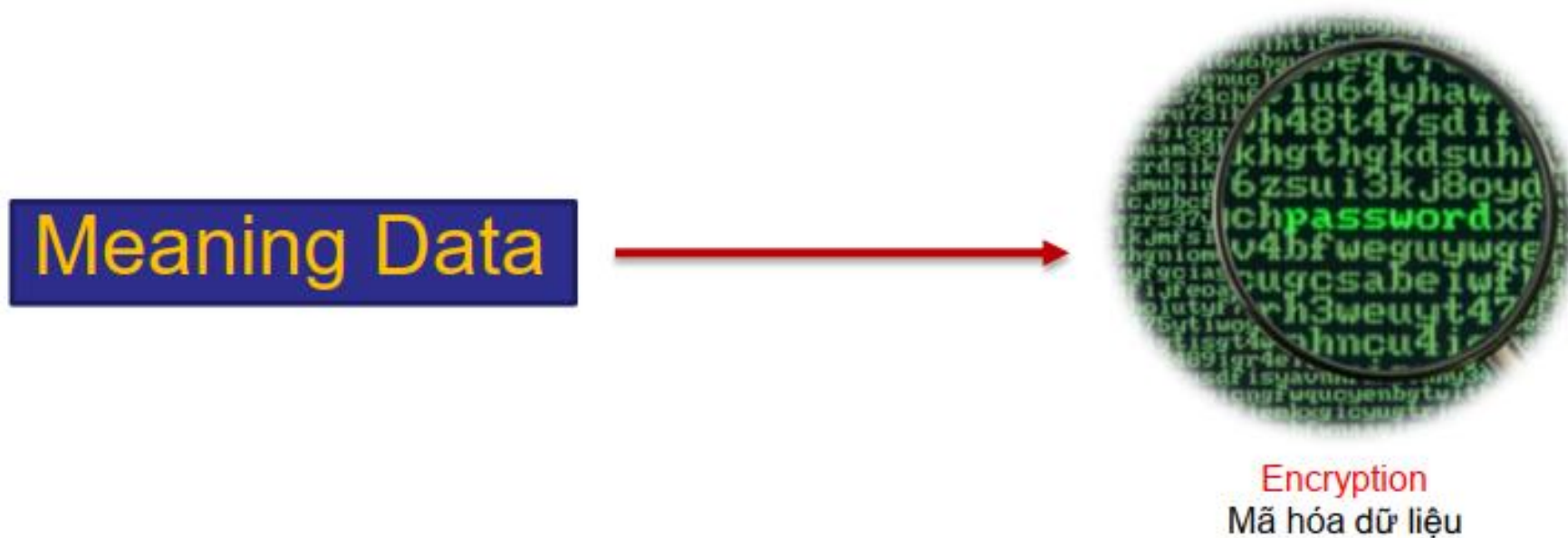
Encryption
Mã hóa dữ liệu



Auditing
Giám sát hoạt động

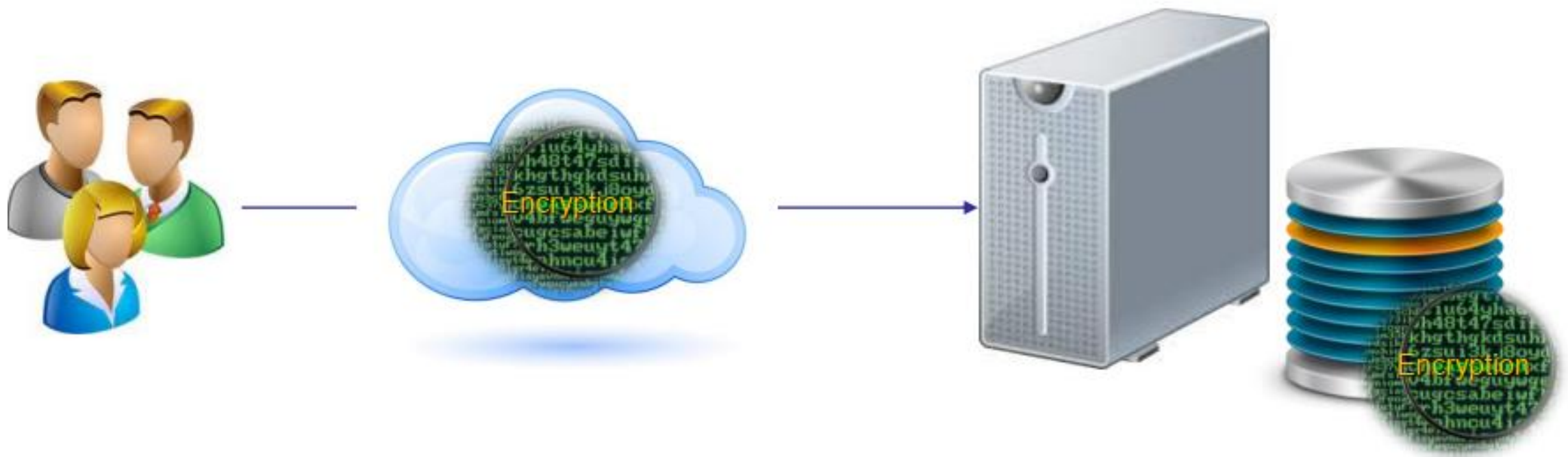
1. Khái quát về mã hóa dữ liệu

- ❖ Mã hóa là phương pháp che giấu dữ liệu, biến dữ liệu sang dạng mã không có ý nghĩa với kẻ tấn công.
- ❖ Đây là rào cản cuối cùng, khi mà kẻ tấn công vượt qua được các cơ chế bảo vệ dữ liệu khác.



1. Khái quát về mã hóa dữ liệu

❖ Mã hóa dữ liệu có thể thực hiện ở 2 thời điểm.

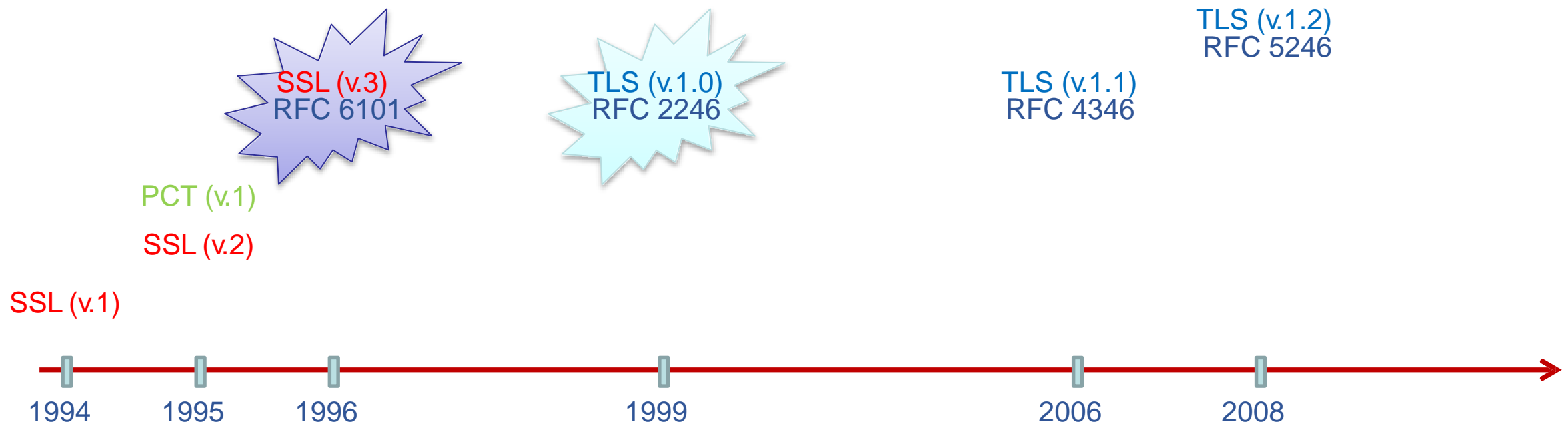


1. Khái quát về mã hóa dữ liệu

❖ Mã hóa dữ liệu trên đường truyền

- SSL (Secure Socket Layer) → Netscape
- PCT (Private Communication Technology) → Microsoft
- TLS (Transport Layer Security) → IETF (Internet Engineering Task Force)

Security) - IETF (Internet Engineering Task Force)



1. Khái quát về mã hóa dữ liệu

❖ Các phương pháp mã hóa

- Mã hóa đối xứng
- Mã hóa bất đối xứng
- Mã hóa lai

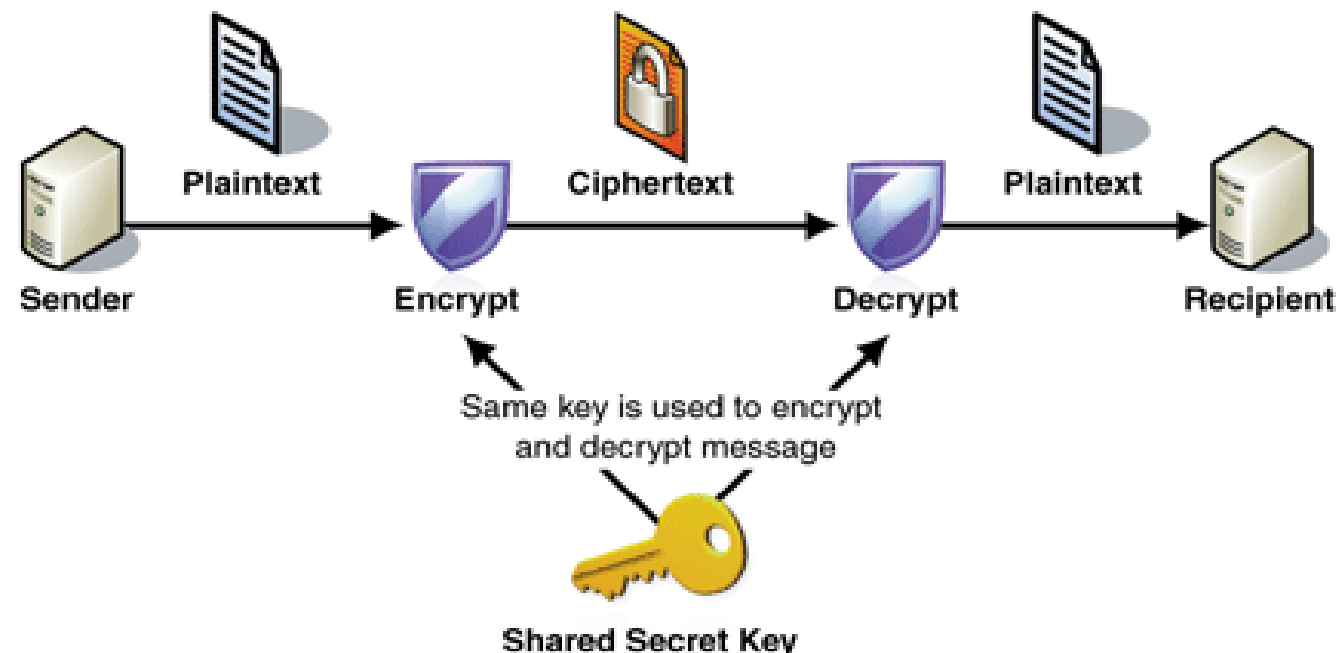
Meaning Data



Encryption
Mã hóa dữ liệu

Phương pháp mã hóa đối xứng

- ❖ Chỉ sử dụng một khóa (Shared Secret Key) để mã hóa và giải mã dữ liệu.
- ❖ Thuật toán đơn giản, độ dài khóa ngắn → tốc độ xử lý nhanh, phù hợp cho bảo mật lượng lớn dữ liệu.
- ❖ Hạn chế
 - Phân phối khóa → cần có hệ thống quản lý khóa
 - Không cung cấp khả năng chống lại sự thoái thác trách nhiệm.



Phương pháp mã hóa đối xứng

❖ Một số thuật toán phổ biến

▪ Block Cipher

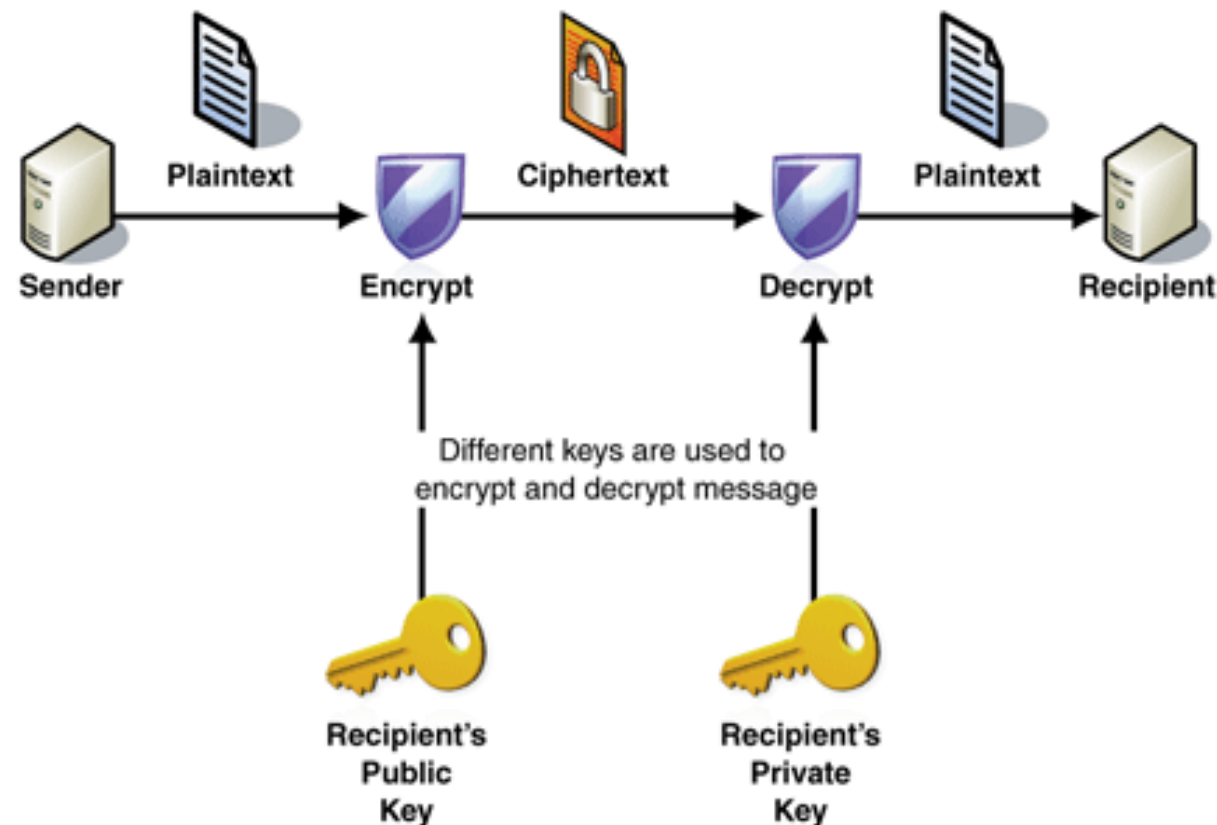
- DES (Data Encryption Standard)
- 3DES (Triple Data Encryption Standard)
- AES (Advanced Encryption Standard)
- SlowFish, TowFish, Serpent

▪ Stream Cipher

- RC4

Phương pháp mã hóa bất đối xứng

- ❖ Sử dụng một cặp khóa private key và public key.
- ❖ Giải quyết được vấn đề trao đổi khóa
- ❖ Thuật toán phức tạp → an toàn, nhưng tốc độ xử lý chậm.
- ❖ Phù hợp cho mã hóa lượng dữ liệu ít.

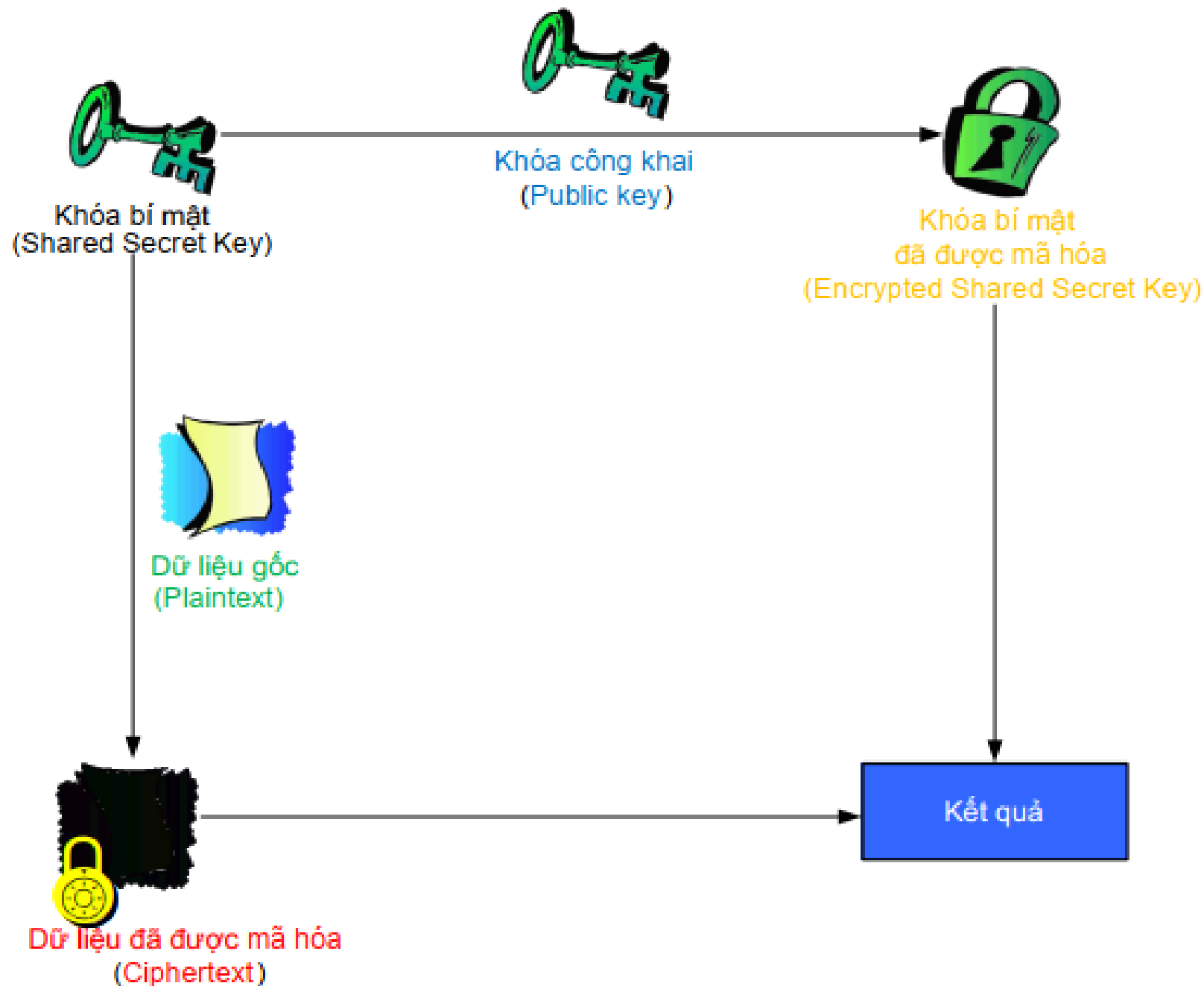


Phương pháp mã hóa bất đối xứng

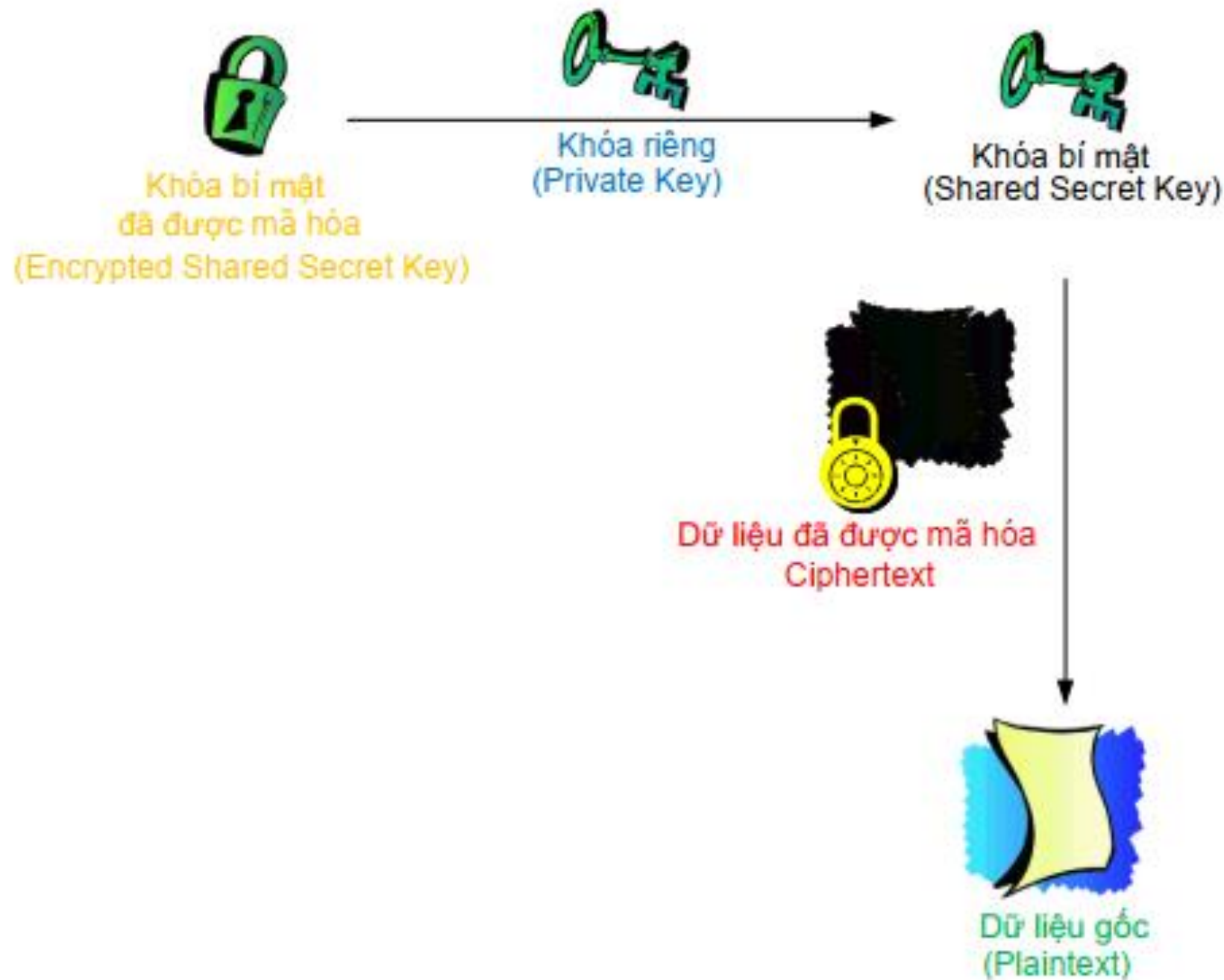
❖ Một số thuật toán phổ biến

- Diffie-Hellman Key Exchange
- Rivest-Shamir-Adleman (RSA)
- Digital Signature Algorithms (DSA)
- ElGamal
- Elliptic Curve Cryptography
- Paillier Cryptosystem

Phương pháp mã hóa lai



Phương pháp mã hóa lai



Phương pháp mã hóa lai

- ❖ Kết hợp phương pháp mã hóa đối xứng và phương pháp mã hóa bất đối xứng.
- ❖ Tận dụng được:
 - Ưu điểm về tốc độ của phương pháp mã hóa đối xứng.
 - Tính an toàn của phương pháp mã hóa bất đối xứng.

2. Các mức độ mã hóa dữ liệu

1. Mã hóa mức ứng dụng (Application Level)
2. Mã hóa mức lưu trữ (Storage Level)
3. Mã hóa mức CSDL (Database Level)

2.1. Mã hóa mức ứng dụng (application level)

- ❖ Việc mã hóa/giải mã dữ liệu được thực hiện ngay trong mã lệnh chương trình ở mức ứng dụng (application), liên quan đến các thao tác xử lý trên dữ liệu cần được bảo vệ, chọn lựa đơn vị dữ liệu mã hóa.
- ❖ Phù hợp với các ứng dụng thực hiện các công việc xử lý, cấp quyền, thao tác trên dữ liệu bí mật ở mức ứng dụng.
- ❖ Tập dụng được thư viện hỗ trợ mã hóa JCE (Java-based application) hoặc MS-CAPI (Microsoft-based application).

2.1. Mã hóa mức ứng dụng

❖ Bảo vệ dữ liệu khỏi nguy cơ

- Thiết bị lưu trữ bị đánh cắp
- Chống được tấn công dữ liệu mức lưu trữ
- Truy cập dữ liệu bí mật từ người quản trị dữ liệu.

❖ Hạn chế

- CSDL không dùng được cho các ứng dụng khác.
- Phải sử dụng mô hình mã hóa/giải mã dữ liệu tương thích hoặc thay đổi mã chương trình khi chia sẻ dữ liệu.

2.2. Mã hóa mức lưu trữ (Storage level)

- ❖ Mã hóa/giải mã tập tin lưu trữ toàn bộ dữ liệu, CSDL với một mã khóa duy nhất.
- ❖ Được thực hiện ở cấp hệ điều hành.
- ❖ Phù hợp cho việc bảo vệ sao lưu, dữ liệu offline.
- ❖ Bảo vệ được dữ liệu khi thiết bị lưu trữ bị đánh cắp hoặc bị tấn công ở mức lưu trữ.
- ❖ Thực tế đã có nhiều nhà cung cấp xây dựng các phần mềm đáp ứng nhu cầu này.

2.2. Mã hóa mức lưu trữ (Storage level)

❑ Hạn chế

- ❖ Không lựa chọn được dữ liệu cần bảo vệ.
- ❖ Không thể phân quyền trên đơn vị dữ liệu nhỏ hơn như bảng, dòng, cột.
- ❖ Không bảo vệ được dữ liệu khỏi những tấn công mức ứng dụng hoặc mức CSDL.
- ❖ Không ngăn chặn được việc quản trị hệ thống truy cập đến tập tin.
- ❖ Không ngăn chặn được việc truy cập đến tập tin dữ liệu đã được mã hóa khi mất quyền quản trị hệ thống.
- ❖ Vấn đề về hiệu năng khi đọc và ghi dữ liệu từ CSDL.

2.3. Mã hóa mức CSDL (Database level)

- ❖ Việc mã hóa/giải mã dữ liệu được thực hiện ở cấp HQT CSDL.
- ❖ Được đảm nhận thông qua việc dùng thủ tục hoặc trigger.
- ❖ Đơn vị dữ liệu được chọn để mã hóa có thể là: từng giá trị tại từng thuộc tính, từng dòng, từng cột, từng bảng...
- ❖ Dễ dàng chia sẻ dữ liệu mã hóa giữa các chương trình ứng dụng khác nhau.
- ❖ Chống được các kiểu tấn công như: đánh cắp thiết bị lưu trữ, tấn công mức CSDL (vd SQL injection, người quản trị truy cập dữ liệu bất hợp pháp).

2.3. Mã hóa mức CSDL (Database level)

□ Các cấp độ mã hóa mức CSDL

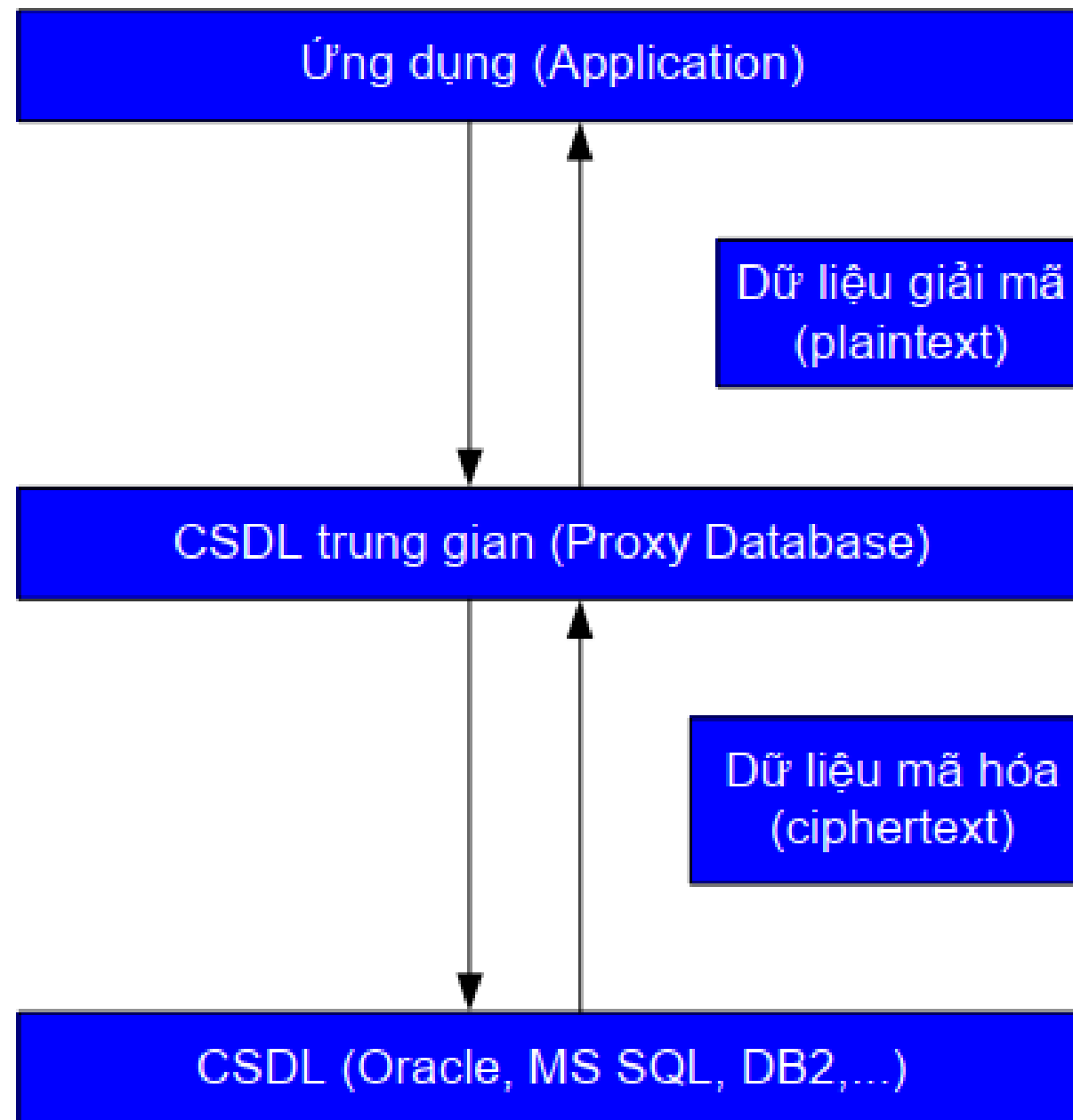
- ❖ **Attribute value** (giá trị thuộc tính): từng giá trị thuộc tính của bộ dữ liệu được mã hóa riêng biệt.
- ❖ **Record/Row level** (bộ/dòng): từng dòng trong bảng được mã hóa riêng lẻ → mã hóa luôn những thuộc tính không cần thiết phải che giấu.
- ❖ **Column/attribute level** (cột/thuộc tính): chỉ mã hóa những thuộc tính nhạy cảm.
- ❖ **Page/block level** (trang/khối): toàn bộ các dòng dữ liệu trong một trang được mã hóa một lần. Số lượng bộ dữ liệu trong trang phụ thuộc vào kích thước trang và kích thước bộ dữ liệu.

2.3. Mã hóa mức CSDL (Database level)

❑ Hạn chế

- ❖ Khi thay đổi **kiểu hay kích thước** của trường dữ liệu liên quan → tiến hành thay đổi thủ tục/trigger mã hóa/giải mã dữ liệu.
- ❖ Làm **chậm hệ thống** đáng kể → chỉ nên mã hóa dữ liệu nhạy cảm.
- ❖ Không an toàn với tấn công ở mức ứng dụng.

3. Các mô hình mã hóa CSDL

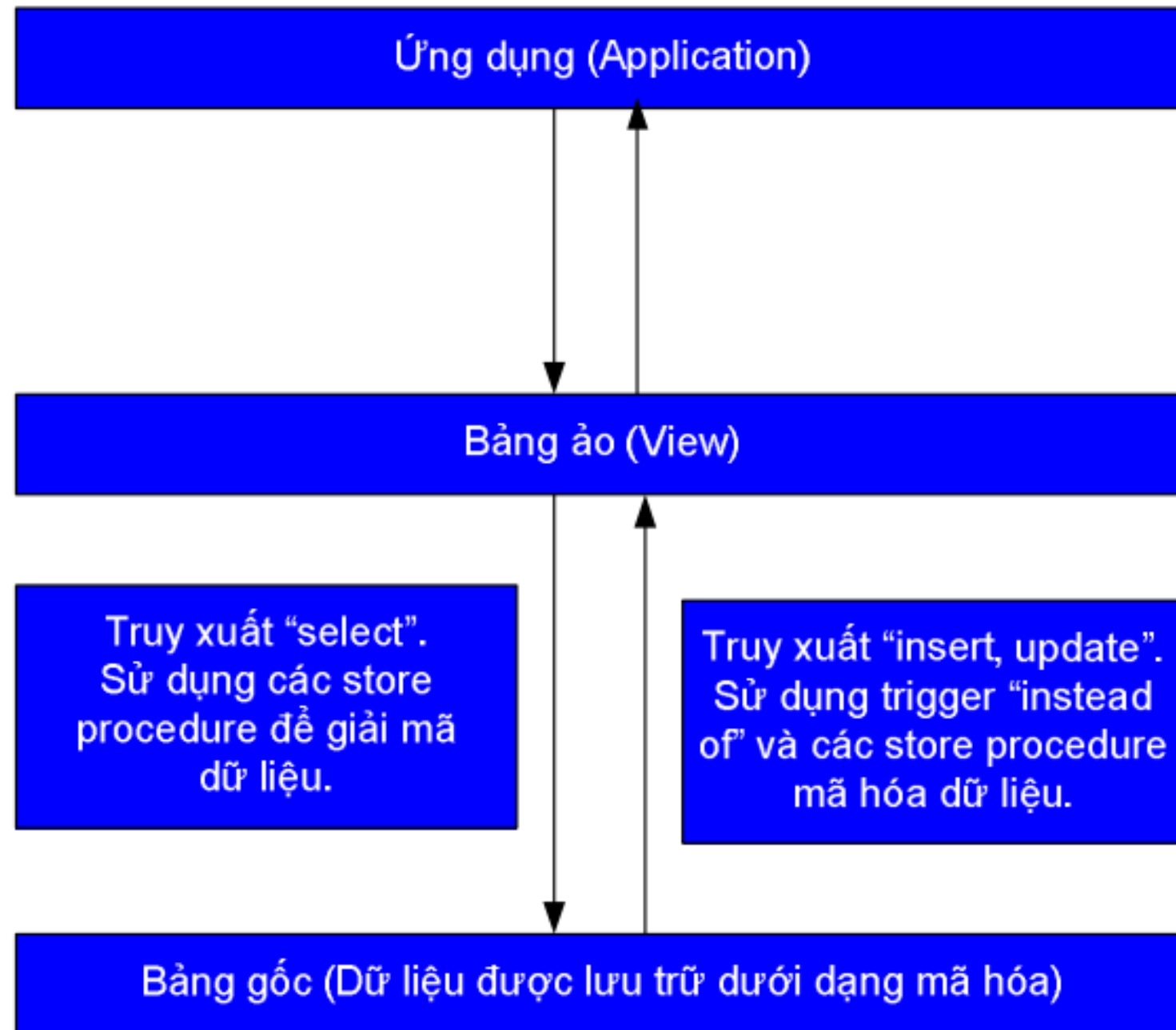


Mô hình CSDL trung gian

Mô hình CSDL trung gian

- ❖ Một CSDL trung gian (proxy) được xây dựng giữa ứng dụng và CSDL gốc, có vai trò:
 - Mã hóa dữ liệu trước khi cập nhật và CSDL gốc.
 - Giải mã dữ liệu trước khi cung cấp cho ứng dụng.
 - Cung cấp thêm chức năng quản lý khóa, xác thực người dùng và cấp phép truy cập.
- ❖ Sản phẩm mã hóa CSDL Secure.Data của công ty Protegrity (www.protegrity.com) sử dụng mô hình xây dựng tầng CSDL trung gian như trên.

3. Các mô hình mã hóa CSDL



Mô hình bảng ảo (view)

Mô hình bảng ảo

- ❖ Sử dụng cơ chế sẵn có trong CSDL.
- ❖ Giải quyết vấn đề mã hóa cột dựa trên cơ chế:
 - Các **Stored Procedure** trong CSDL cho chức năng mã hóa và giải mã.
 - Tạo **bảng ảo (view)** thay thế các bảng thật trong CSDL.
 - Sử dụng **“instead of” trigger** nhằm tự động hóa quá trình mã hóa từ các bảng ảo đến bảng gốc.
- ❖ Sản phẩm mã hóa CSDL DBEncrypt (www.appsecinc.com) và nCypher (www.ncypher.com) được phát triển theo mô hình này.

4. Nhận xét về các giải pháp mã hóa CSDL

□ Ưu điểm

- ❖ Mã hóa CSDL có thể giúp che giấu dữ liệu khỏi những kẻ xâm nhập, thậm chí cả DBA nếu họ không được phép truy cập dữ liệu.
- ❖ Mã hóa CSDL là phương pháp bảo vệ dữ liệu rất hiệu quả đối với những tấn công mức lưu trữ. Những kẻ tấn công có được dữ liệu nhưng không thể hiểu được dữ liệu.

4. Nhận xét về các giải pháp mã hóa CSDL

❑ Khuyết điểm

- ❖ Mã hóa CSDL làm **tăng lượng xử lý khi truy cập dữ liệu**, tăng dung lượng lưu trữ dữ liệu.
- ❖ Mã hóa CSDL làm HQT CSDL không thể thực thi các phương pháp truy cập dữ liệu cơ bản.
- ❖ Mã hóa đòi hỏi phải có chính sách **quản lý khóa** thích hợp.
- ❖ Mã khóa là thành phần quan trọng nhất
 - Mất khóa → bị lộ dữ liệu
 - Mất khóa → không giải mã được dữ liệu

4. Nhận xét về các giải pháp mã hóa CSDL

- ❖ Mặc dù mã hóa là cần thiết, nhưng mã hóa không phải là giải pháp hoàn toàn tốt
 - Mã hóa không thể đảm nhận công việc điều khiển truy cập mà chỉ nhằm che giấu nội dung dữ liệu.
 - Việc mã hóa không được làm ảnh hưởng đến kết quả của việc điều khiển truy cập. Ví dụ: A có quyền SELECT trên bảng NHANVIEN thì khi mã hóa xong A không bị ngăn cản quyền này.
 - Điều không mong muốn: DBA có quyền truy cập đến toàn bộ dữ liệu → Mã hóa CSDL.
 - **Mã hóa toàn bộ CSDL không phải là giải pháp tốt!**

5. Một số vấn đề liên quan đến giải pháp mã hóa

1. Khóa chính, khóa ngoại và ràng buộc toàn vẹn.
2. Chỉ mục trên dữ liệu mã hóa
3. Tìm kiếm trên dữ liệu mã hóa
4. Quản lý khóa

5.1. Khóa chính, khóa ngoại và ràng buộc toàn vẹn

- ❖ Nếu khóa chính chứa dữ liệu nhạy cảm → cần mã hóa
- ❖ **Giải pháp mã hóa:** Mã hóa dữ liệu ở tất cả các dòng tại các cột tham gia làm khóa chính của bảng dữ liệu.
 - Sử dụng cũng 1 mã khóa + cùng 1 vector khởi tạo (IV)
 - Sử dụng mỗi dòng một mã khóa khác nhau
 - Sử dụng cũng 1 mã khóa + khác vector khởi tạo
- ❖ **Vấn đề nào cần lưu ý khi mã hóa dữ liệu trên khóa chính?**

5.1. Khóa chính, khóa ngoại và ràng buộc toàn vẹn

- ❖ **Vi phạm RB khóa chính** (khi mã khóa khác nhau hoặc cùng mã khóa nhưng IV khác nhau)
 - Hủy RB khóa chính + tự cài đặt thủ tục kiểm tra
- ❖ **Vi phạm RB khóa ngoại**
 - Mã hóa cùng mã khóa và IV với giá trị tham chiếu ở khóa chính đã được mã hóa.
- ❖ **Vi phạm RBTV khác** trên khóa chính, khóa ngoại (nếu hệ thống có sẵn dữ liệu)
 - Hủy tất cả RB, tiến hành mã hóa và tạo lại RB.
- ❖ Không thực hiện được RBTV hiện có (do đặc tính của dữ liệu)
 - **Tự cài đặt lại** bằng hàm/thủ tục/trigger.

5.2. Vấn đề chỉ mục trên dữ liệu mã hóa

- ❖ Mục tiêu của việc lập chỉ mục trong CSDL
 - Tăng tốc độ tìm kiếm
- ❖ Nếu cần mã hóa trên dữ liệu nhạy cảm có chỉ mục → cần giải quyết 2 trường hợp sau:
 - Lập chỉ mục cho các cột dữ liệu đã được mã hóa
 - Lập chỉ mục trước khi mã hóa dữ liệu
- ❖ Các HQT CSDL không khuyến khích lập chỉ mục trên dữ liệu mã hóa, vì trong nhiều trường hợp việc tìm kiếm trên cột dữ liệu đã mã hóa yêu cầu HQT CSDL phải duyệt qua toàn bộ bảng để xác định phần tử cần tìm. Khi đó, **việc lập chỉ mục trở nên vô nghĩa.**

5.3. Vấn đề tìm kiếm trên dữ liệu mã hóa

❖ Tìm kiếm chính xác

→ Sử dụng cùng một khóa và IV khi mã hóa tất cả giá trị trên cột dữ liệu cần tìm kiếm

❖ Tìm kiếm gần đúng (like, >, <, ...)

→ Thông thường phải duyệt toàn bộ bảng nếu như không có cơ chế hỗ trợ tìm kiếm nhanh dùng chỉ mục.

→ Giải pháp chung: Áp dụng hàm băm mật mã trên một phần của dữ liệu nhạy cảm và lưu cùng dòng nhưng trên một cột khác.

5.3. Vấn đề tìm kiếm trên dữ liệu mã hóa

- ❖ **Ví dụ:** Có bảng dữ liệu lưu thông tin của khách hàng. Trong đó có trường **Email đã được mã hóa** vì là thông tin nhạy cảm.
- ❖ Để có thể tìm kiếm trên trường Email, sẽ tạo thêm một cột lưu lại giá trị băm của 4 ký tự đầu của địa chỉ Email đó.
- ❖ Cách giải quyết này cũng có thể dùng cho việc tìm kiếm chính xác với điều kiện là **biết trước điều kiện tìm kiếm** thường được thành lập trên những tiêu chí nào (4 ký tự đầu, 5 ký tự cuối...)
- ❖ Với cách tiếp cận này, luôn phải tạo ra thêm 1 trường mới để phục vụ cho mỗi một nhu cầu tìm kiếm.

5.4. Vấn đề quản lý khóa

- ❖ **Mã khóa** sử dụng trong quá trình mã hóa CSDL cần phải:
 - Tạo ra và truyền khóa cho người dùng được phép
 - Lưu trữ các khóa cho lần truy cập sau
- ❖ Việc quản lý khóa mã hóa **phải đảm bảo**
 - Những người dùng không có quyền thì không được “thấy” dữ liệu nhạy cảm đang được bảo vệ.
 - Dữ liệu sẽ được mã hóa cho từng người nhận khác nhau, với các quyền hạn khác nhau.
 - Các mã khóa phải được đảm bảo an toàn

Quản lý khóa

1. Lưu khóa trong CSDL
2. Quản lý khóa bởi ứng dụng
3. Tính toán ra khóa
4. Quản lý khóa dùng phương pháp mã hóa hai

5.4.1. Lưu khóa trong CSDL

❖ Mã khóa được lưu trong CSDL nhằm thuận tiện cho việc sao lưu và phục hồi dữ liệu.

❖ Một số lưu ý

- Bảng lưu trữ khóa phải được che giấu và bảo vệ chặt chẽ.
- Sử dụng cơ chế điều khiển truy cập để hạn chế truy cập vào bảng lưu trữ khóa.
- Tên bảng và tên các thuộc tính của bảng lưu trữ khóa không nên đặt rõ ràng.

5.4.1. Lưu khóa trong CSDL

□ Một số lưu ý

- ❖ Không nên tạo ràng buộc giữa bảng lưu trữ khóa và các bảng khác để tránh sự suy diễn.
- ❖ Dữ liệu lưu trữ trong bảng lưu khóa cũng phải được mã hóa với các hàm mã hóa và giải mã tự xây dựng.
- ❖ Duy trì việc giám sát truy cập vào bảng này và kiểm tra định kỳ.
- ❖ Lưu ý về rủi ro đối với việc can thiệp và thay đổi khóa của DBA.zZZ

5.4.2. Quản lý khóa bởi ứng dụng

- ❖ Lưu trữ khóa trong tập tin của Application Server.
- ❖ Tập tin lưu khóa phải được mã hóa bằng một Master Key.
- ❖ Đảm bảo khóa chỉ được gửi đến các chương trình liên quan.
- ❖ Đảm bảo DBA cũng không lấy được khóa.
- ❖ Luôn đảm bảo các khóa này phải giải mã được dữ liệu
→ cần có chiến lược để đảm bảo các khóa quản lý bởi ứng dụng được lưu trữ và sao lưu an toàn.
- ❖ Lưu ý rủi ro mất khóa khi ứng dụng gặp sự cố.

5.4.3. Tính toán ra khóa

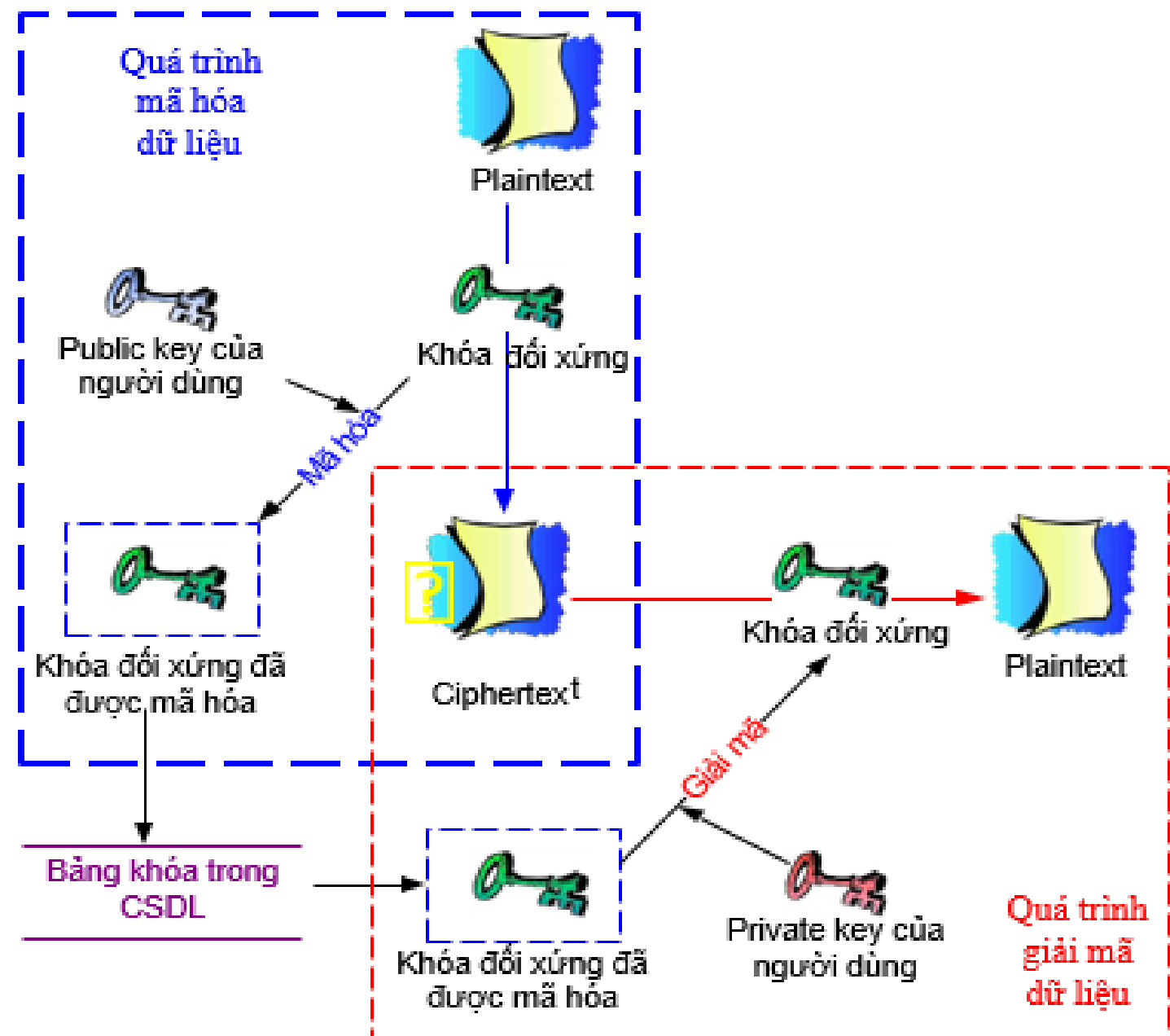
- ❖ Một cách hiệu quả để quản lý khóa là không lưu trữ khóa một cách thật sự. Các khóa có thể được tính toán ra một cách gián tiếp bởi một hàm dựa trên một giải thuật bảo mật.
- ❖ Các hàm, thủ tục trong CSDL nên được che dấu để ngăn chặn những người dùng có quyền thực thi tất cả thủ tục biết được thuật toán bảo mật.
- ❖ Sự an toàn của các khóa trong CSDL liên quan mật thiết đến sự an toàn của các thuật toán tạo khóa. Nếu mã chương trình bị phân tích → thuật toán bị lộ → các khóa sẽ bị lộ.

5.4.4. Quản lý khóa bằng PP mã hóa lai

- ❖ Sử dụng mã hóa đối xứng để mã/giải mã dữ liệu ở một cột dữ liệu nhạy cảm bằng một mã khóa bí mật.
- ❖ Sử dụng mã khóa khác nhau để mã/giải mã dữ liệu ở các cột không có quan hệ với nhau.
- ❖ Mỗi người dùng có một cặp khóa bất đối xứng (**public/private**). **Private key** được bảo vệ bằng mật khẩu của người dùng (xem như là passphrase).
- ❖ Khi người dùng được cấp quyền truy cập dữ liệu mã hóa → Mã khóa bí mật được mã bằng Public key của người dùng và được lưu trữ công khai trong CSDL.
- ❖ Mã khóa bí mật bị mã này chỉ được giải mã bằng Private key của người dùng đã được cấp quyền truy cập dữ liệu → có được mã khóa bí mật để giải mã dữ liệu.

5.4.4. Quản lý khóa bằng PP mã hóa lai

Khi người chủ của private key quên passphrase thì coi như dữ liệu trở nên vô nghĩa, không có cách nào giải mã dữ liệu.



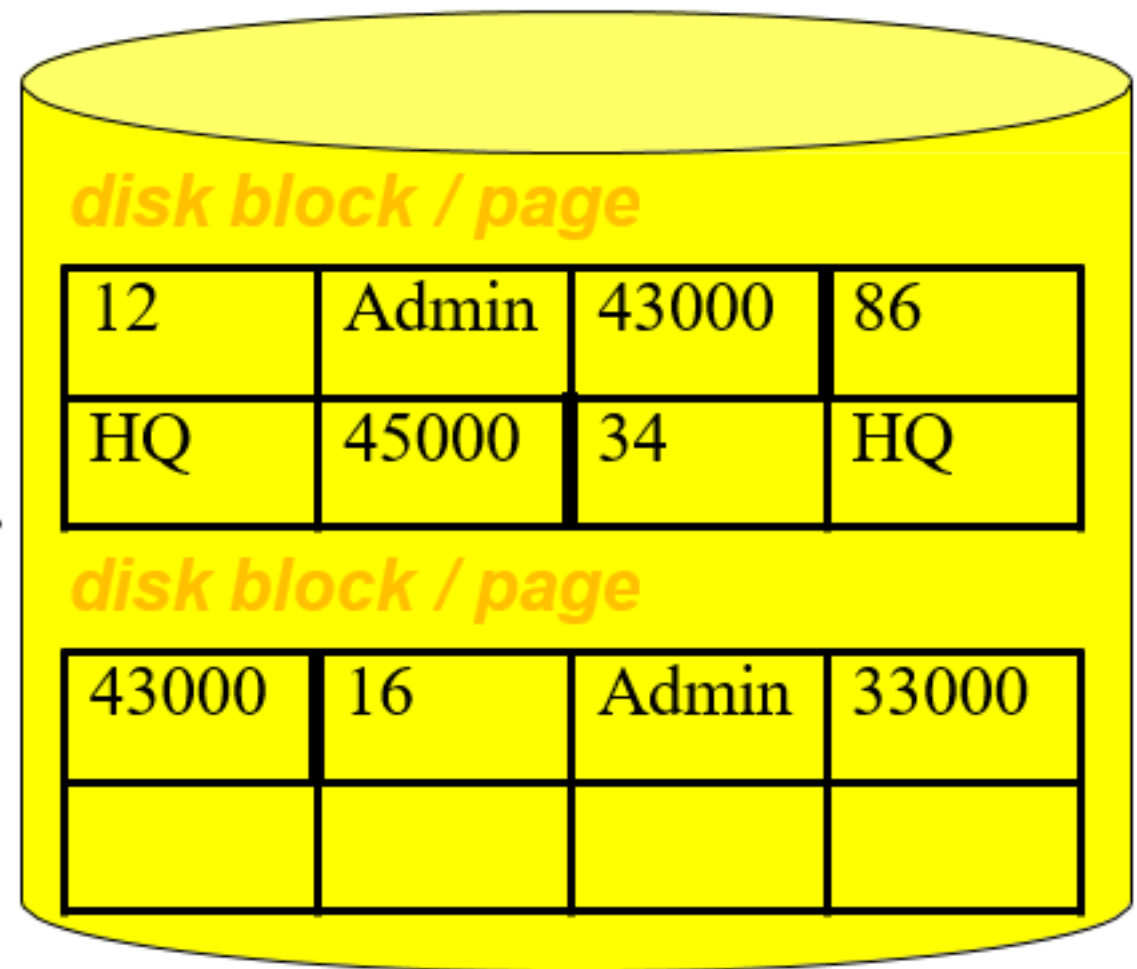
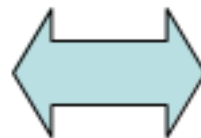
6. Mô hình lưu trữ dữ liệu mã hóa

1. N-ary Storage Mode (NSM)
2. Optimize NSM
3. Partition Plaintext Ciphertext Model (PPC)

6.1. N-ary Storage Mode (NSM)

- ❖ Các thuộc tính/bộ dữ liệu được lưu trữ tuần tự, hết thuộc tính/bộ dữ liệu này đến thuộc tính/bộ dữ liệu khác.

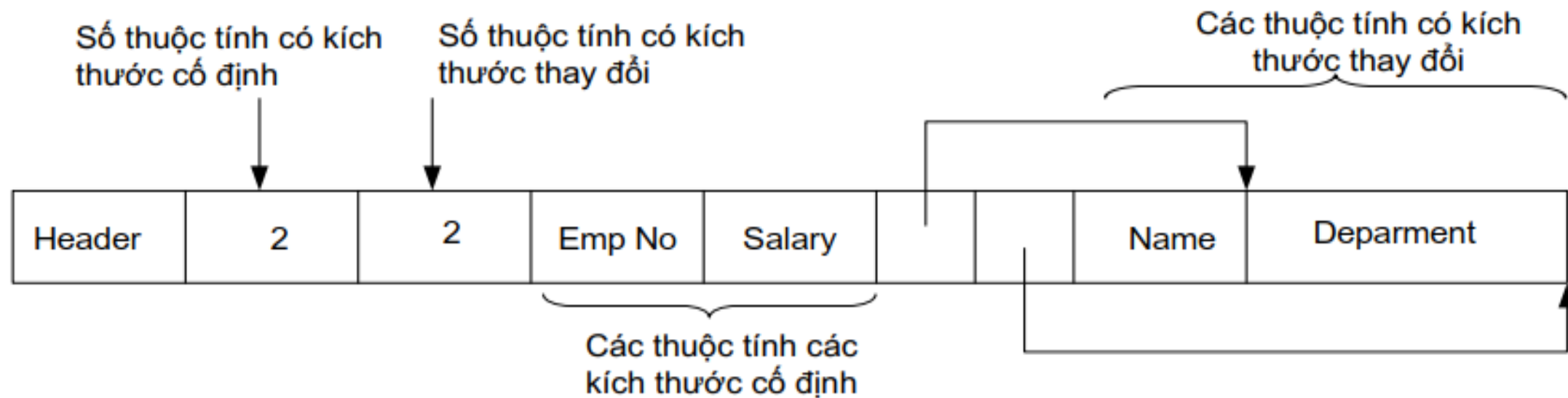
ID	DEPT	SALARY
12	Admin	43000
86	HQ	45000
34	HQ	43000
16	Admin	33000



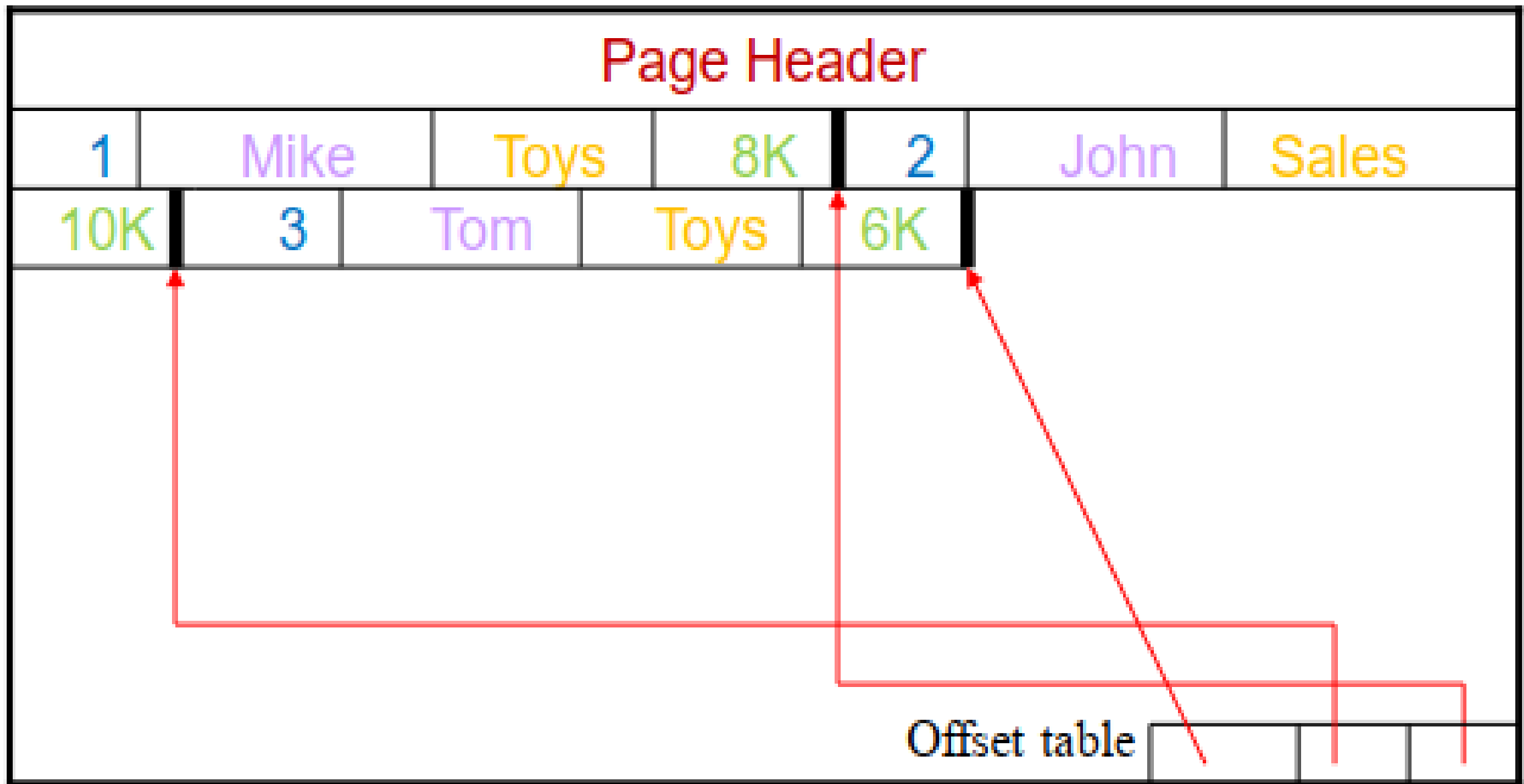
6.1. N-ary Storage Mode (NSM)

❖ Ví dụ: quan hệ **Employee** có 4 thuộc tính: **EmpNo**, **Name**, **Department**, **Salary**.

→ Cấu trúc thông tin lưu trữ dữ liệu trong mô hình NSM như sau:



6.1. N-ary Storage Mode (NSM)



6.1. N-ary Storage Mode (NSM)

- NSM hoạt động tốt với việc đọc ghi trong bộ nhớ phụ và hiệu quả khi truy vấn liên quan đến hầu hết các thuộc tính của bộ dữ liệu.
- Kiến trúc NSM không thích hợp khi lưu trữ dữ liệu mã hóa. Đặc biệt khi một bộ dữ liệu chứa cả dữ liệu nhạy cảm và dữ liệu không nhạy cảm, vấn đề tính toán và lưu trữ sẽ gia tăng đáng kể.

6.1. N-ary Storage Mode (NSM)

- ❖ Xét quan hệ **Employee** (EmpNo, Name, Department, Salary), **Cần bảo vệ Name và Salary** → mã hóa
 - ❖ Chọn mã hóa cấp độ thuộc tính (**attribute level**)
 - Tránh mã hóa các dữ liệu không cần thiết
 - Mỗi thuộc tính của 1 dòng cần 1 thao tác mã hóa
 - ❖ Chọn thuật toán mã hóa bằng **khóa đối xứng** theo kiểu **block cipher**
 - Dữ liệu mã hóa cần chuyển thành các khối có kích thước quy ước tùy theo thuật toán, VD thuật toán **AES** quy ước block size = **16** bytes.
 - Nếu **Salary** (2 bytes) → cần đệm thêm 14 bytes.
- Gia tăng kích thước lưu trữ khi mã hóa
- Tăng thao tác mã hóa/giải mã

6.2. Optimize NSM

- ❖ Cải tiến mô hình NSM để giảm chi phí tính toán và lưu trữ.
- ❖ Trong mỗi bộ dữ liệu, các thuộc tính được sắp xếp thành 2 phần:
 - **Plaintext**: chứa các thuộc tính không mã hóa
 - **Ciphertext**: chứa các thuộc tính cần mã hóa → chỉ thực hiện 1 lần thao tác mã hóa trên khối dữ liệu này.
- ❖ Mỗi bộ dữ liệu sau đó vẫn được lưu trữ riêng lẻ, tuần tự.
- ❖ Giảm kích thước dữ liệu đệm, giảm thao tác mã hóa/giải mã.
- ❖ Cần 1 thao tác mã hóa/giải mã cho mỗi bộ dữ liệu, dữ liệu đệm dùng riêng cho từng bộ.

6.3. Partition Plaintext Ciphertext Model

- ❖ PPC mã hóa cấp độ trang/khối (page/block)
- ❖ Mỗi bộ dữ liệu được chia thành 2 bộ dữ liệu con
 - Một bộ chứa plaintext
 - Một bộ chứa ciphertext
- ❖ PPC lưu các dữ liệu trong 1 trang thành 2 trang con
 - Phần đầu của trang (plaintext mini-page): lưu tất cả các bộ dữ liệu con chứa plaintext
 - Phần còn lại của trang (ciphertext mini-page): lưu trữ các bộ dữ liệu con chứa ciphertext.
- ❖ Phần Page Header: bổ sung thông tin trỏ đến địa chỉ lưu trữ dữ liệu của 2 trang con.

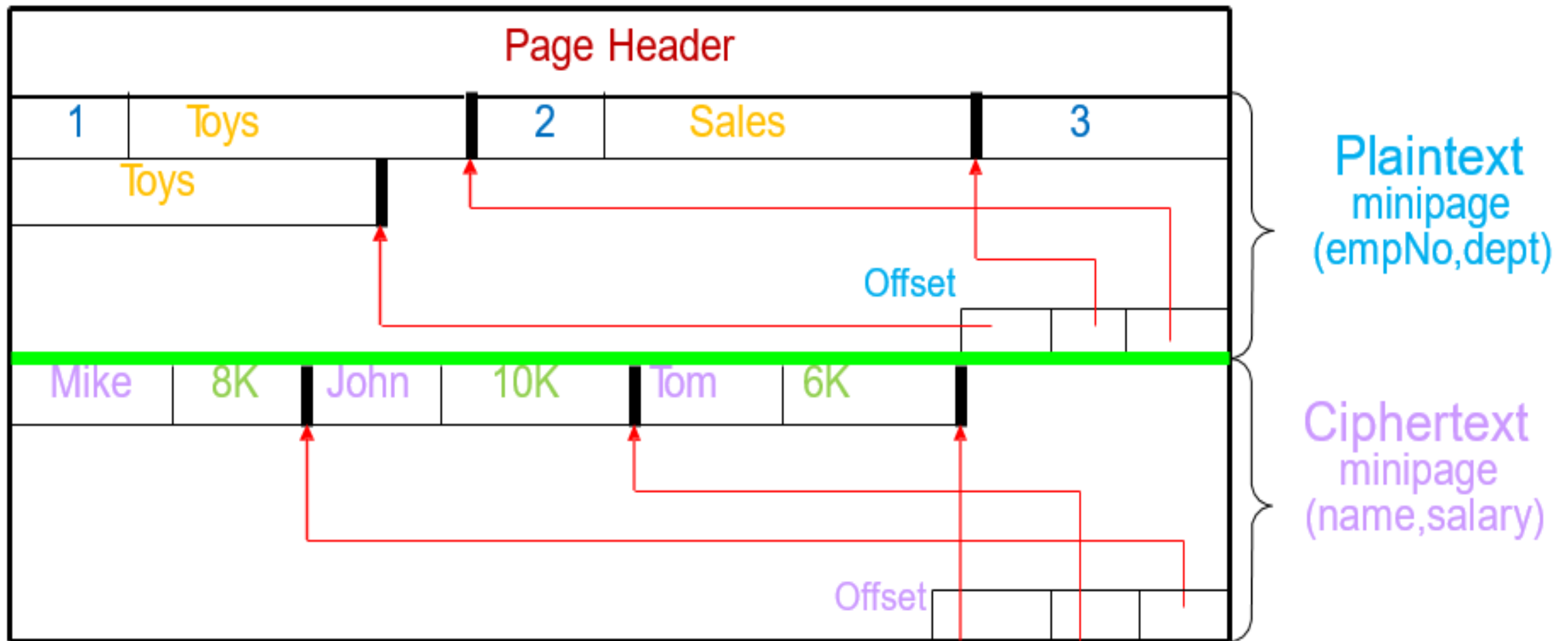
Source: Bala Iyer, Sharad Mehrotra, Einar Mykletun, Gene Tsudik², and Yonghua Wu, A Framework for Efficient Storage Security in RDBMS, EDBT 2004, LNCS 2992, pp. 147–164, 2004

6.3. Partition Plaintext Ciphertext Model

- ❖ Cấu trúc một trang con trong PPC giống như NSM.
- ❖ Các bộ dữ liệu trong 2 trang con được lưu theo thứ tự có liên quan.
- ❖ Cuối mỗi trang con sẽ có một bảng chỉ mục (**offset table**) chỉ đến điểm kết thúc của mỗi bộ dữ liệu con.
- ❖ **Mỗi trang** chỉ cần thực hiện **1 thao tác mã hóa/giải mã** cho 1 truy vấn.
- ❖ Mô hình PPC **giảm được chi phí mã hóa** (tính toán và lưu trữ) nhưng vẫn giữ được lược đồ của NSM.
- ❖ Số lượng bộ dữ liệu trên một trang trong mô hình PPC và NSM là như nhau.

Source: Bala Iyer, Sharad Mehrotra, Einar Mykletun, Gene Tsudik², and Yonghua Wu, *A Framework for Efficient Storage Security in RDBMS*, EDBT 2004, LNCS 2992, pp. 147–164, 2004

6.3. Partition Plaintext Ciphertext Model



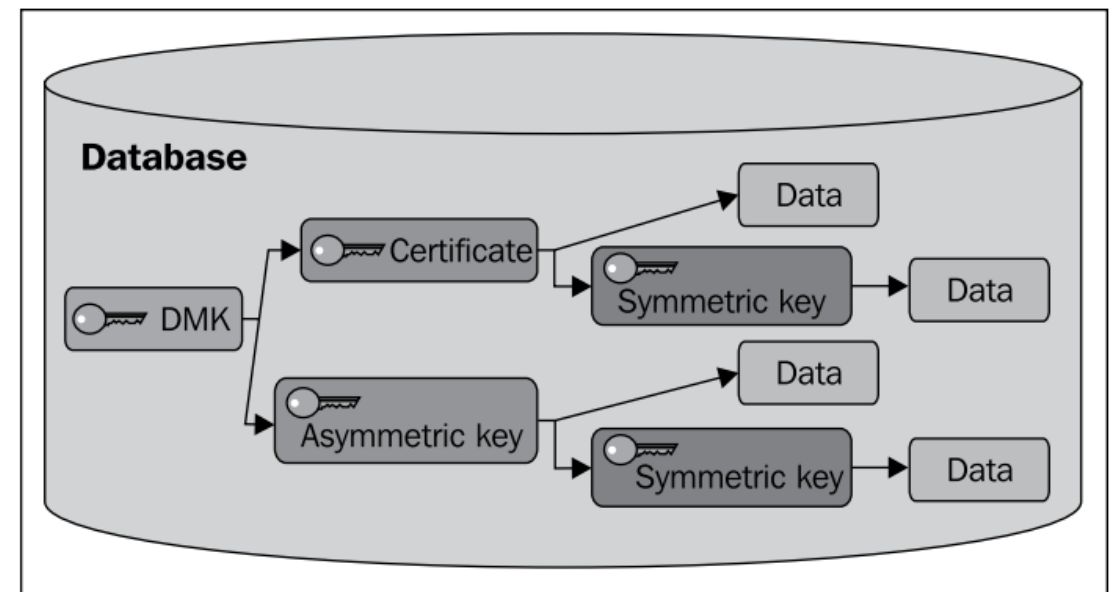
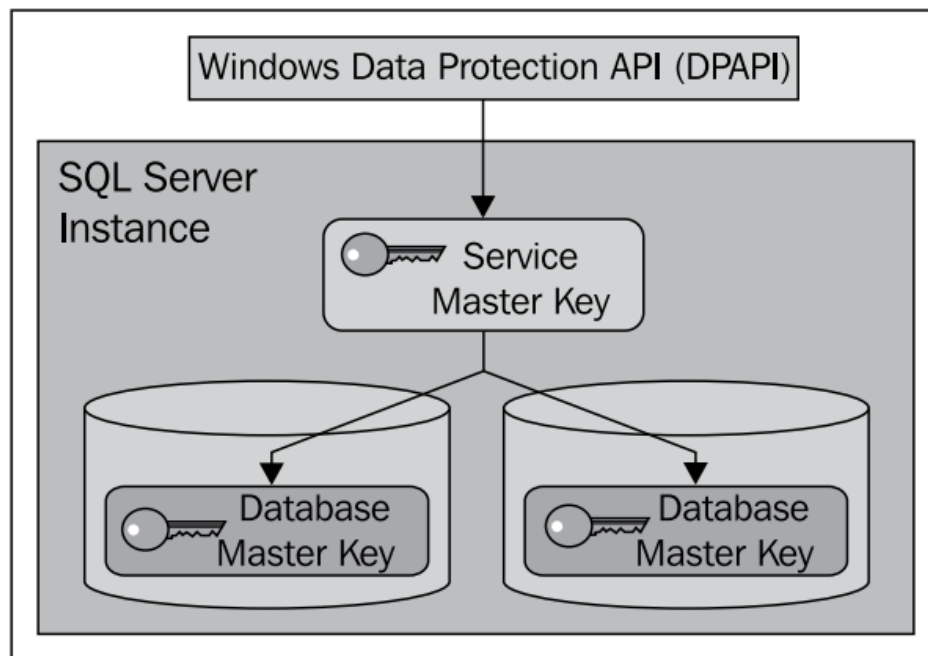
Source: Bala Iyer, Sharad Mehrotra, Einar Mykletun, Gene Tsudik², and Yonghua Wu, A Framework for Efficient Storage Security in RDBMS, EDBT 2004, LNCS 2992, pp. 147–164, 2004

Lưu trữ dữ liệu mã hóa

Cấp độ mã hóa	Số lượng thao tác mã / giải mã	
	NSM Model	PPC Model
Cấp độ thuộc tính	M*N (Chỉ mã hóa dữ liệu ở các thuộc tính nhạy cảm)	1 (Chỉ mã hóa những dữ liệu nhạy cảm)
Cấp độ dòng	M (Mã hóa toàn bộ dữ liệu, bao gồm dữ liệu không nhạy cảm)	1 (Chỉ mã hóa những dữ liệu nhạy cảm)
Cấp độ trang	1 (Mã hóa toàn bộ dữ liệu, bao gồm dữ liệu không nhạy cảm)	1 (Chỉ mã hóa những dữ liệu nhạy cảm)

7. Mã hóa CSDL trong SQL Server

- ❖ Mã hóa một chiều (Hash)
- ❖ Mã hóa đối xứng
- ❖ Mã hóa bất đối xứng



Một số khái niệm về khóa trong SQL Server

❑ Service Master Key (SMK)

- ❖ Là gốc của hệ thống phân cấp mã khóa, được tạo tự động khi SQL Server thực hiện mã hóa các khóa khác lần đầu tiên.
- ❖ Dùng để bảo vệ Database Master Key.
- ❖ Được bảo vệ bởi Windows Data Protect API (DPAPI), sử dụng thuật toán AES.
- ❖ Xem thông tin SMK trong view `SYS.SYMMETRIC_KEYS`: SMK là một khóa đối xứng, luôn có ID = 102 và có tên là `##MS-ServiceMasterKey##`.

Một số khái niệm về khóa trong SQL Server

❑ Database Master Key (DMK)

- Là gốc của hệ thống phân cấp mã hóa trong một CSDL, được tạo trong CSDL hiện hành và một bản sao khác được lưu trong CSDL master.
- Là một khóa đối xứng được dùng để bảo vệ các chứng chỉ và các khóa bất đối xứng.
- Được bảo vệ bởi mật khẩu và SMK, sử dụng thuật toán AES và được mở tự động.

Database Master Key

- ❖ Tạo DMK cho CSDL QLNV được bảo vệ bằng mật khẩu

USER QLNV

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'p@sswOrdF0rDMK'

- ❖ Mở khóa DMK bằng mật khẩu

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'p@sswOrdF0rDMK'

- ❖ Sao lưu DMK vào file, mã hóa bằng mật khẩu

BACKUP MASTER KEY TO FILE = 'E:\encryption_keys\dmk.key'
ENCRYPTION BY PASSWORD = 'MyB@ckupP@sswOrdF0rDMK'

Database Master Key

❖ Đóng DMK

```
CLOSE MASTER KEY
```

❖ Khôi phục DMK từ file

```
RESTORE MASTER KEY FROM FILE = 'E:\encryption_keys\dmk.key'  
DECRYPTION BY PASSWORD = 'MyB@ckupP@sswOrdF0rDMK'  
ENCRYPTION BY PASSWORD = 'p@sswOrdF0rDMK'
```

❖ Tạo bản sao DMK được bảo vệ bằng SMK

```
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY
```

❖ Xóa DMK

```
DROP MASTER KEY
```

Một số khái niệm về khóa trong SQL Server

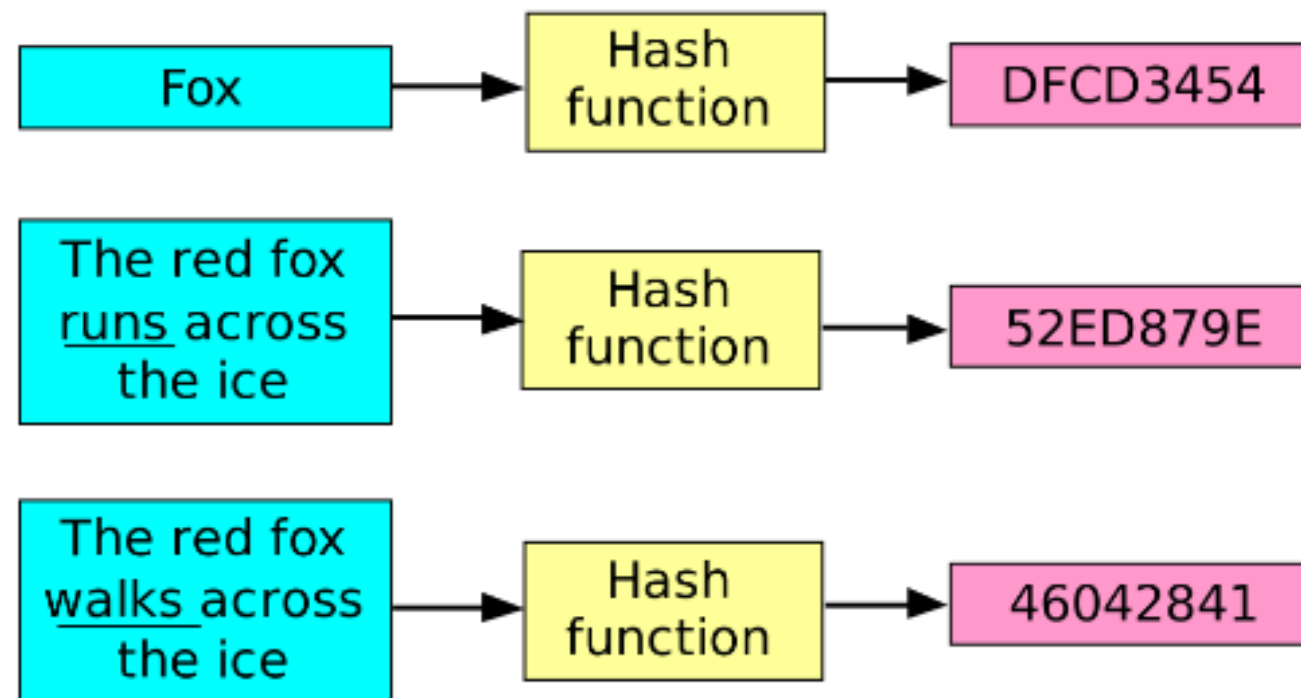
- **Khóa đối xứng** (Khóa bí mật)
 - Sử dụng cùng một khóa để mã hóa và giải mã
 - Có thể được bảo vệ bằng một khóa bất đối xứng hoặc một chứng chỉ (Certificate).
- **Khóa bất đối xứng**: sử dụng 2 khóa khác nhau, một dùng để mã hóa, một dùng để giải mã.
- **Chứng chỉ số** (Certificate): là một khóa bất đối xứng với public key được đính kèm thông tin định danh của người sở hữu private key.

Các kỹ thuật mã hóa trong SQL Server

- Mã hóa một chiều (Hash)
- Mã hóa đối xứng
- Mã hóa bất đối xứng

Mã hóa một chiều (Hash)

- Không sử dụng khóa mà chỉ dựa trên thuật toán
- Chỉ có thể mã hóa mà không thể giải mã ra dữ liệu gốc
- Sử dụng hàm băm với một thuật toán để biến một chuỗi plain text thành một chuỗi hash có độ dài nhất định.



Mã hóa một chiều (Hash)

HASHBYTES ('thuật toán', <plain text>)

- **Các thuật toán áp dụng để băm:**
 - ✓ Trả về 128 bits (16 bytes): MD2, MD4, MD5
 - ✓ Trả về 160 bits (20 bytes): SHA, SHA1
 - ✓ Trả về 256 bits (32 bytes): SHA2_256
 - ✓ Trả về 512 bits (64 bytes): SHA2_512
- Plain text: kiểu chuỗi hoặc nhị phân
- Kiểu dữ liệu trả về: VARBINARY (tối đa 8000 bytes)

Mã hóa một chiều (Hash)

```
SELECT HASHBYTES('MD5', 'Hello') AS ENCRYPTED_STRING_1  
SELECT HASHBYTES('MD5', 'Hello') AS ENCRYPTED_STRING_2
```

ENCRYPTED_STRING_1	
1	0x8B1A9953C4611296A827ABF8C47804D7
ENCRYPTED_STRING_2	
1	0x8B1A9953C4611296A827ABF8C47804D7

→ Chuỗi plain text giống nhau về nội dung và kiểu dữ liệu thì có kết quả mã hóa giống nhau.

Mã hóa một chiều (Hash)

```
SELECT HASHBYTES('SHA1', CAST(150 AS CHAR)) AS ENCRYPTED_NUMBER_1  
SELECT HASHBYTES('SHA1', CAST(150 AS VARCHAR)) AS ENCRYPTED_NUMBER_2  
SELECT HASHBYTES('SHA1', CAST(150 AS NVARCHAR)) AS ENCRYPTED_NUMBER_3
```

ENCRYPTED_NUMBER_1	
1	0x8660D08D4F4CDDF2CC10E3B4B1FA7645DF546B41
ENCRYPTED_NUMBER_2	
1	0x13682AC418603AA0966369D46BBF282F562ACF47
ENCRYPTED_NUMBER_3	
1	0x5E696457ACA34F9A438A4F107A537E40DDCDD335

→ Cùng plaintext nhưng kiểu dữ liệu khác nhau thì cho kết quả mã hóa khác nhau.

Mã hóa một chiều (Hash)

➤ Mã hóa trên cột trong table

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhau	Luong
001	Trần Lan Anh	1995-10-12 00:0...	anhti	lananh123	8000000
002	Nguyễn Gia Linh	1990-08-05 00:0...	linhng	linhgia247	6500000
003	Lê Minh Trung	1985-11-10 00:0...	trunglm	trungminh199	11000000

```
UPDATE NHANVIEN SET MatKhau=CONVERT(varchar(100),  
                                     HASHBYTES('SHA2_256',MatKhau),1)
```

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhau	Luong
001	Trần Lan Anh	1995-10-12 00:0...	anhti	0xB7C87E447958DEAFED816FC2C869E5C55BD98725A799884ADB99...	8000000
002	Nguyễn Gia Linh	1990-08-05 00:0...	linhng	0x348EED2D739167C2099AD89397D8738EB31DA32449892BF99E7A3...	6500000
003	Lê Minh Trung	1985-11-10 00:0...	trunglm	0x5863F3EA70015156838AC43929A7E7095A57C115405764D465F157...	11000000

Mã hóa một chiều (Hash)

➤ Tạo trigger tự động mã hóa khi insert dữ liệu

```
CREATE TRIGGER trig_Insert_MK ON NHANVIEN
FOR INSERT AS
BEGIN
    --Cập nhật mật khẩu của nhân viên thêm vào là chuỗi được mã hóa
    UPDATE NHANVIEN
    SET MatKhau = ( SELECT CONVERT(VARCHAR(100),
                                HASHBYTES('SHA2_256', i.MatKhau), 1)
                    FROM inserted i)
    WHERE MaNV = ( SELECT i.MaNV
                   FROM inserted i)
END
```

Mã hóa một chiều (Hash)

➤ Khi **insert** dữ liệu vào table

```
INSERT INTO NHANVIEN VALUES ('004', N'Trần Thanh Tâm', '1985-06-07',  
                                'tamtt', 'thanhtran2000', 5500000)
```

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhau	Luong
001	Trần Lan Anh	1995-10-12 00:0...	anhti	0xB7C87E44795...	8000000
002	Nguyễn Gia Linh	1990-08-05 00:0...	linhng	0x348EED2D739...	6500000
003	Lê Minh Trung	1985-11-10 00:0...	trunglm	0x5863F3EA700...	11000000
004	Trần Thanh Tâm	1985-06-07 00:0...	tamtt	0x102CF9C8ED6...	5500000

Mã hóa một chiều (Hash)

- Tạo trigger tự động mã hóa khi update mật khẩu

```
CREATE TRIGGER trig_Update_MK ON NHANVIEN
FOR UPDATE AS
BEGIN
    IF UPDATE (MatKhau) --chỉ thực hiện khi MatKhau được update
    BEGIN
        --Cập nhật mật khẩu là chuỗi được mã hóa 1 chiều
        UPDATE NHANVIEN
        SET MatKhau = ( SELECT CONVERT(VARCHAR(100),
                                    HASHBYTES('SHA2_256', i.MatKhau), 1)
                        FROM inserted i)
        WHERE MaNV = ( SELECT i.MaNV
                        FROM inserted i)
    END
END
```

Mã hóa một chiều (Hash)

➤ **Tìm kiếm** dữ liệu mã hóa một chiều

→ Cần khớp plaintext và thuật toán mã hóa

```
SELECT * FROM NHANVIEN  
WHERE TenDangNhap = 'tamtt'  
AND MatKhai = CONVERT(VARCHAR(100),  
HASHBYTES('SHA2_256', 'thanhtran2000'), 1)
```

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhai	Luong
004	Trần Thanh Tâm	1985-06-07 00:00:00	tamtt	0x102CF9C8ED6579714A4BBA5E1BEF58D40A22B0...	5500000

Mã hóa một chiều với salt

- **Salt** là một chuỗi ngẫu nhiên được thêm vào chuỗi plaintext trước khi mã hóa nhằm tăng độ phức tạp cho giá trị của hàm băm, gây khó khăn khi suy ngược.
- Một trong những cách để tạo một chuỗi ngẫu nhiên là sử dụng hàm CRYPT GEN RANDOM.

CRYPT_GEN_RANDOM (<độ dài (byte)>)

→ Kiểu dữ liệu trả về: VARBINARY (tối đa 8000 bytes).

Mã hóa một chiều với salt

➤ Mã hóa trên chuỗi

```
DECLARE @MySecret VARCHAR(20) = 'my secret'  
DECLARE @MySalt VARCHAR(100) = CONVERT(VARCHAR(100), CRYPT_GEN_RANDOM(8), 1)  
SELECT @MySecret + @MySalt, HASHBYTES('SHA1', @MySecret + @MySalt)
```

Results Messages		
	(No column name)	(No column name)
1	my secret0xD595B92FFB4094A1	0x2070F83D930EF269AD27E90C78E1FD05388396DD

→ Khó suy ngược ra chuỗi ban đầu.

Mã hóa một chiều với salt

➤ Mã hóa **trên cột của table**

– Tạo thêm cột mới để lưu **salt**

```
ALTER TABLE NHANVIEN ADD Salt VARCHAR(100)
```

– Phát sinh **salt**

```
UPDATE NHANVIEN SET Salt = CONVERT(VARCHAR(100), CRYPT_GEN_RANDOM(8), 1)
```

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhau	Luong	Salt
001	Trần Lan Anh	1995-10-12 00:0...	anhtl	0x0EBD43AD0E...	8000000	0x965E3DDACD7DA137
002	Nguyễn Gia Linh	1990-08-05 00:0...	linhng	0xBB4CE56B260...	6500000	0xC4520F186FC3EAC5
003	Lê Minh Trung	1985-11-10 00:0...	trunglm	0x1D2E7BE3B3E...	11000000	0x381614434E5C2D5C
004	Trần Thanh Tâm	1985-06-07 00:0...	tamtt	0x10E9051B6D4...	5500000	0xD1F879F4B3EC8B67

Mã hóa một chiều với salt

➤ Mã hóa **trên cột của table**

- Nối mật khẩu với salt và mã hóa

```
UPDATE NHANVIEN SET MatKhau = CONVERT(VARCHAR(100),  
                                         HASHBYTES('SHA2_256', MatKhau + Salt), 1)
```

MaNV	HoTen	NgaySinh	TenDangNhap	MatKhau	Luong	Salt
001	Trần Lan Anh	1995-10-12 00:00:00	anhti	0x86C16AFAD5...	8000000	0x965E3DDACD...
002	Nguyễn Gia Linh	1990-08-05 00:00:00	linhng	0x3EEA594E1B7...	6500000	0xC4520F186FC...
003	Lê Minh Trung	1985-11-10 00:00:00	trunglm	0x772B9190813...	11000000	0x381614434E5...
004	Trần Thanh Tâm	1985-06-07 00:00:00	tamtt	0x85FDE81C8E8...	5500000	0xD1F879F4B3E...

Mã hóa đối xứng

- Mã hóa bằng **passphrase**
- Mã hóa bằng **khóa đối xứng** (Symmetric key)

Mã hóa đối xứng bằng passphrase

- Truyền vào **Passphrase** (xem như một chuỗi mật khẩu dài), sử dụng thuật toán **TRIPLE DES** với khóa có độ dài 128 bit.
 - Passphrase có thể chứa khoảng trắng và thường được đặt là **1 chuỗi có nghĩa** nên dễ ghi nhớ hơn một chuỗi mật khẩu.
- **Cú pháp** mã hóa bằng passphrase

EncryptByPassPhrase ('chuỗi passphrase', <plain text>)

- ✓ Passphrase và plain text: kiểu chuỗi hoặc nhị phân
- ✓ Kiểu dữ liệu trả về: **VARBINARY** (tối đa 8000 bytes)

Mã hóa đối xứng bằng passphrase

➤ **Cú pháp** giải mã bằng passphrase

`DecryptByPassPhrase` ('chuỗi passphrase', <cipher text>)

- ✓ Cipher text: kiểu nhị phân
- ✓ Kiểu dữ liệu trả về: `VARBINARY` (tối đa 8000 bytes) hoặc `NULL` (nếu sai passphrase)

Lưu ý: muốn trả về đúng plain text cần chuyển đổi kết quả hàm giải mã sang đúng kiểu dữ liệu ban đầu của plain text.

Mã hóa đối xứng bằng passphrase

➤ Ví dụ:

```
Declare @encrypted VARBINARY(128)
```

```
Set @encrypted = EncryptByPassphrase('This is my passphrase', 'My secret')
```

```
Select @encrypted AS encrypted, CONVERT(VARCHAR,  
    DecryptByPassphrase('This is my passphrase', @encrypted)) AS decrypted
```

encrypted	decrypted
0x01000000AF9D15FF99272D1A02FFDC5243382CBC509520B00186918E3025F79C3A4F9605	My secret

Mã hóa đối xứng bằng passphrase

➤ **Ví dụ:** passphrase kiểu nhị phân (ở dạng số hex)

```
Declare @encrypted VARBINARY(128)
```

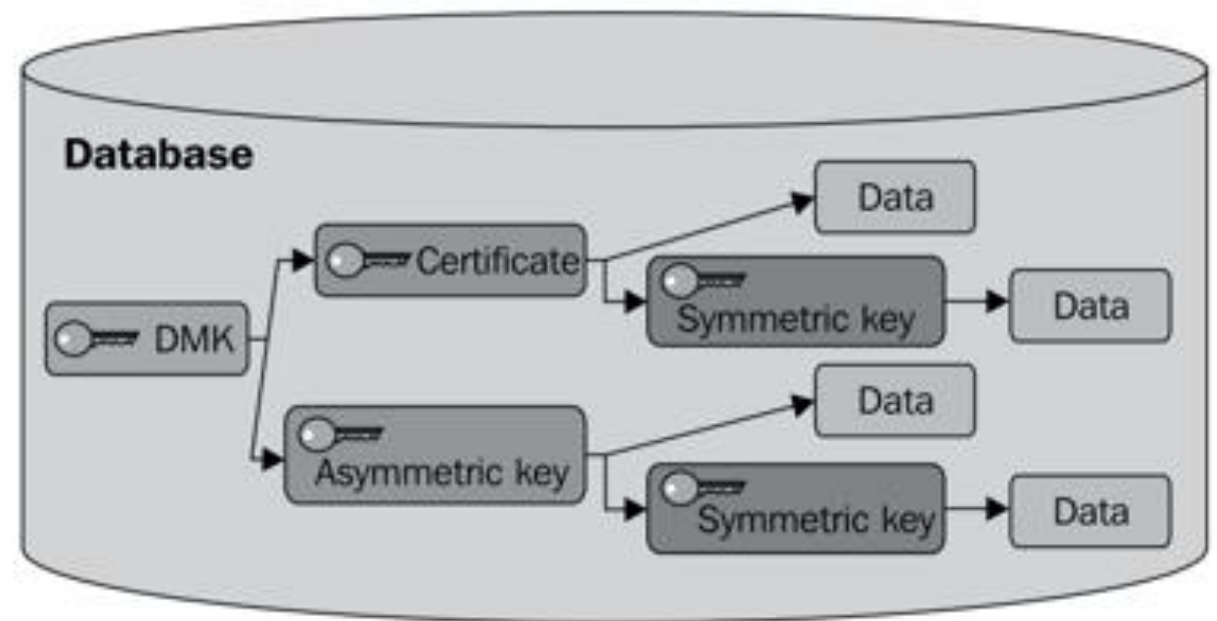
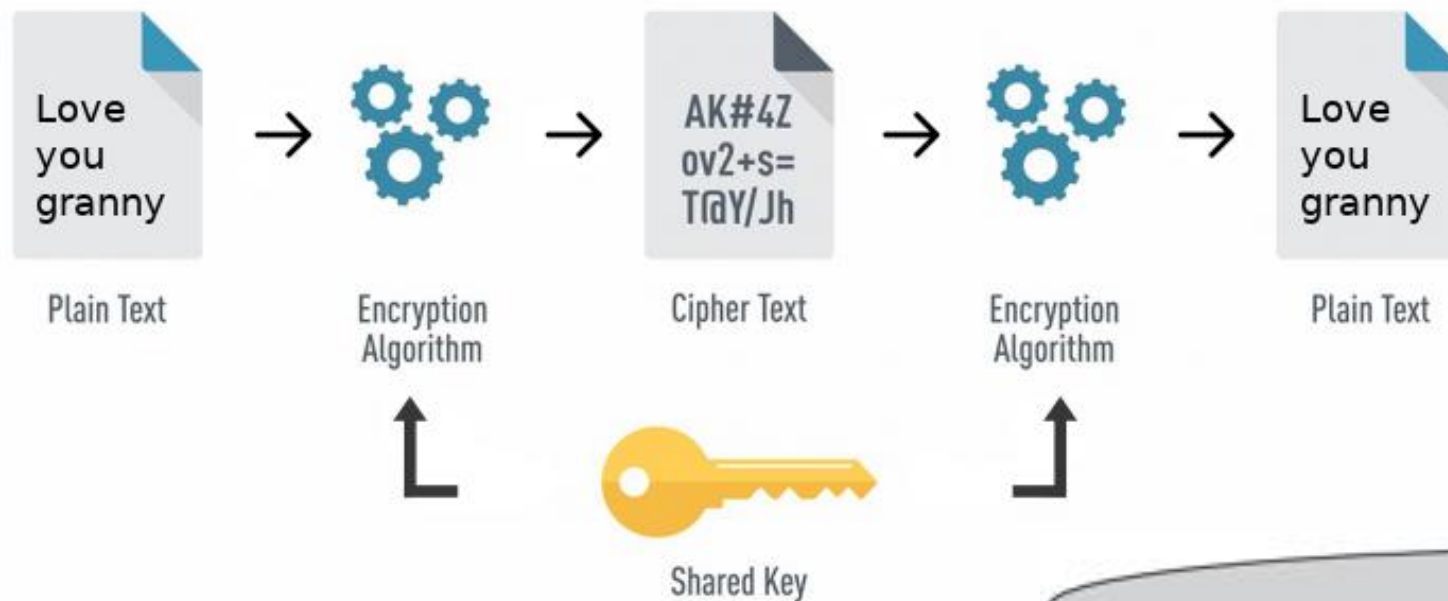
```
Set @encrypted = EncryptByPassphrase(0x123456789, 'My secret')
```

```
Select @encrypted AS encrypted, CONVERT(VARCHAR,  
    DecryptByPassphrase(0x123456789, @encrypted)) AS decrypted
```

encrypted	decrypted
0x010000000D525774D133AB2D2F748EAE51992DEF8AEBEC367868B45E2CDBD732777C476A	My secret

Mã hóa bằng khóa đối xứng (Symmetric key)

Symmetric Encryption



Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Tạo khóa đối xứng

CREATE SYMMETRIC KEY <tên_khóa>	
AUTHORIZATION <chủ_sở_hữu> ,	User có toàn quyền trên khóa
WITH ALGORITHM = <thuật_toán>	AES_128, AES_192, AES_256
ENCRYPTION BY PASSWORD = 'mật khẩu'	Mã hóa bằng mật khẩu
ENCRYPTION BY CERTIFICATE <tên_chứng_chỉ>	Mã hóa bằng chứng chỉ
ENCRYPTION BY SYMMETRIC KEY <tên_khóa_đx>	Mã hóa bằng khóa đối xứng
ENCRYPTION BY ASYMMETRIC KEY <tên_khóa_bất_đx>	Mã hóa bằng khóa bất đối xứng

Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Mở khóa đối xứng

OPEN SYMMETRIC KEY <tên_khóa>	
DECRYPTION BY PASSWORD = 'mật khẩu'	Giải mã bằng mật khẩu
DECRYPTION BY CERTIFICATE <tên_chứng_chỉ>	Giải mã bằng chứng chỉ
DECRYPTION BY SYMMETRIC KEY <tên_khóa_đx>	Giải mã bằng khóa đối xứng
DECRYPTION BY ASYMMETRIC KEY <tên_khóa_bất_đx>	Giải mã bằng khóa bất đối xứng

➤ Đóng khóa đối xứng

CLOSE SYMMETRIC KEY <tên_khóa>

Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Thay đổi mật khẩu của khóa

```
ALTER SYMMETRIC KEY MySymKey  
ADD ENCRYPTION BY PASSWORD = '<new_password>'
```

```
ALTER SYMMETRIC KEY MySymKey  
DROP ENCRYPTION BY PASSWORD = '<old_password>'
```

➤ Cấp quyền trên khóa

Quyền sử dụng khóa	GRANT CONTROL ON SYMMETRIC KEY :: <tên khóa> TO user
Quyền thay đổi khóa	GRANT ALTER ON SYMMETRIC KEY :: <tên khóa> TO user

Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Mã hóa dữ liệu bằng khóa đối xứng

```
EncryptByKey(Key_GUID('tên/định danh khóa'),  
             <plaintext>)
```

```
EncryptByKey(Key_GUID('tên/định danh khóa'),  
             <plaintext>, 1, <authenticator>)
```

- ✓ Plaintext, authenticator: kiểu chuỗi hoặc nhị phân
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu chưa mở khóa, khóa không tồn tại hoặc text là NULL.

Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Giải mã dữ liệu bằng khóa đối xứng

```
DecryptByKey(<ciphertext>)
```

```
DecryptByKey(<ciphertext>, 1, <authenticator>)
```

- ✓ Ciphertext: kiểu nhị phân
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu chưa mở khóa, khóa không tồn tại hoặc text là NULL.

Mã hóa bằng khóa đối xứng (Symmetric key)

➤ Xóa khóa đối xứng

DROP SYMMETRIC KEY <tên khóa>

➤ Xem thông tin các khóa đối xứng có trong CSDL trên View

SYS.SYMMETRIC_KEYS

Results		Messages			
	name	symmetric_key_id	key_length	algorithm_desc	create_date
1	##MS_DatabaseMasterKey##	101	256	AES_256	2019-04-09 18:40:07.027
2	SymKey_001	283	128	AES_128	2019-05-23 10:23:40.690
3	SymKey_002	284	128	AES_128	2019-05-23 10:23:40.693
4	SymKey_003	285	128	AES_128	2019-05-23 10:23:40.697

Mã hóa bằng khóa đối xứng (Symmetric key)

--Tạo khóa đối xứng mã hóa bằng mật khẩu

```
CREATE SYMMETRIC KEY MySymKey
```

```
WITH ALGORITHM = AES_256
```

```
ENCRYPTION BY PASSWORD = 'p@sswOrdOr$ymKey'
```

--Mở khóa

```
OPEN SYMMETRIC KEY MySymKey
```

```
DECRYPTION BY PASSWORD = 'p@sswOrdOr$ymKey'
```

Mã hóa bằng khóa đối xứng (Symmetric key)

--Mã hóa

```
DECLARE @encrypted VARBINARY(8000)
DECLARE @string NVARCHAR(50) = N'An toàn và bảo mật'
SET @encrypted = EncryptByKey(Key_GUID('MySymKey'), @string)
```

--Giải mã

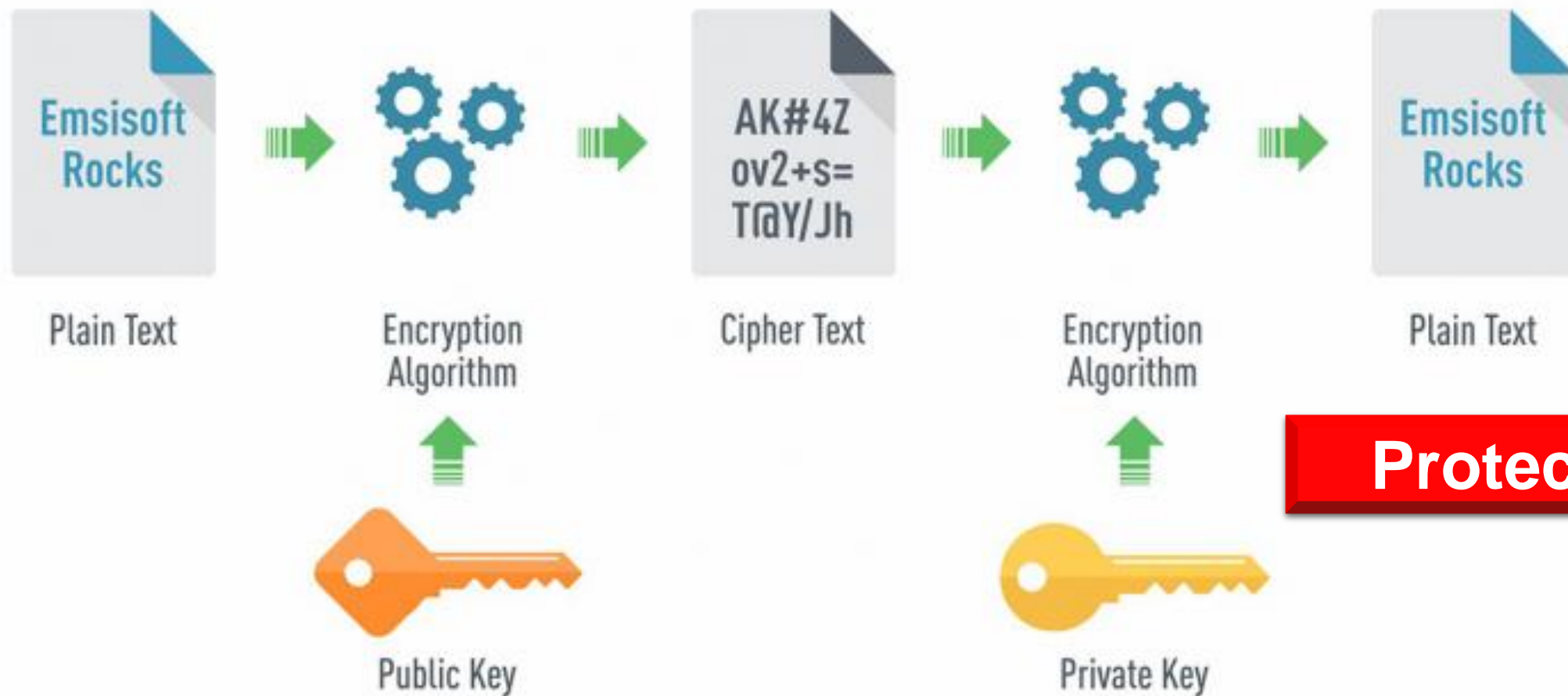
```
DECLARE @decrypted NVARCHAR(50)
SET @decrypted = CONVERT(NVARCHAR, DecryptByKey(@encrypted))
```

--Đóng khóa

```
CLOSE SYMMETRIC KEY MySymKey
```

Mã hóa bất đối xứng

Asymmetric Encryption



Protected

Mã hóa bất đối xứng

- Mã hóa bằng **khóa bất đối xứng** (Asymmetric key)
- Mã hóa bằng **chứng chỉ** (Certificate)

Mã hóa bất đối xứng bằng khóa

❑ Tạo khóa bất đối xứng

CREATE ASYMMETRIC KEY <tên_khóa>	
AUTHORIZATION <chủ_sở_hữu> ,	User có toàn quyền trên khóa
WITH ALGORITHM = <thuật_toán>	RSA_4096, RSA_3072, RSA_2048
ENCRYPTION BY PASSWORD = 'mật khẩu'	Mã hóa bằng mật khẩu*

Mã hóa bất đối xứng bằng khóa

❑ Thay đổi mật khẩu của private key

```
ALTER ASYMMETRIC KEY MyAKey WITH PRIVATE KEY (  
    DECRYPTION BY PASSWORD = '<old_password>',  
    ENCRYPTION BY PASSWORD = '<new_password>')
```

❑ Cấp quyền trên khóa bất đối xứng

Sử dụng khóa để mã hóa và giải mã	GRANT CONTROL ON ASYMMETRIC KEY :: <tên khóa> TO user
Chỉ được sử dụng khóa để mã hóa	GRANT VIEW DEFINITION ON ASYMMETRIC KEY :: <tên khóa> TO user
Thay đổi khóa	GRANT ALTER ON ASYMMETRIC KEY :: <tên khóa> TO user

Mã hóa bất đối xứng bằng khóa

❑ Mã hóa dữ liệu bằng khóa bất đối xứng

```
EncryptByASymKey(ASymKey_ID('tên/định danh  
khóa'), <plain text>)
```

- ✓ Plain text: kiểu chuỗi hoặc nhị phân
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu khóa không tồn tại hoặc text là NULL.

Mã hóa bất đối xứng bằng khóa

❑ Giải mã dữ liệu bằng khóa bất đối xứng

```
DecryptByASymKey(AsymKey_ID('tên/định danh  
khóa'), <cipher text>, <mật khẩu>)
```

- ✓ Ciphertext: kiểu nhị phân
- ✓ Mật khẩu: kiểu NVARCHAR
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu khóa không tồn tại hoặc text là NULL.

Mã hóa bất đối xứng bằng khóa

❑ **Xóa** khóa bất đối xứng

DROP ASYMMETRIC KEY <tên khóa>

➤ **Xem** thông tin các khóa bất đối xứng có trong CSDL trên View

SYS.ASYMMETRIC_KEYS

name	prin...	asymmetric_key_id	pvt_key_encryption_...	pvt_key_encryption_type_desc	thumbprint
MyKey	1	256	PW	ENCRYPTED_BY_PASSWORD	0x03A56128A039C486

algorit	algorithm_desc	key_length	sid
3R	RSA_2048	2048	0x010300000000000090200000003A56128A03

Mã hóa bất đối xứng bằng khóa

--Tạo khóa bất đối xứng mã hóa bằng mật khẩu

```
CREATE ASYMMETRIC KEY MyAKey  
WITH ALGORITHM = RSA_2048  
ENCRYPTION BY PASSWORD = 'myp@sswOrdOrA$ymKey'
```

--Mã hóa

```
DECLARE @encrypted VARBINARY(8000)  
DECLARE @string NVARCHAR(50) = N'An toàn và bảo mật'  
SELECT @encrypted=  
EncryptByASymKey(AsymKey_ID('MyAKey'), @string)
```

--Giải mã

```
DECLARE @decrypted NVARCHAR(50)  
SELECT @decrypted = CONVERT(NVARCHAR,  
DecryptByASymKey(AsymKey_ID('MyAKey'), @encrypted,  
N'myp@sswOrdOrA$ymKey'))
```

Mã hóa bất đối xứng bằng chứng chỉ

❑ **Xóa** khóa bất đối xứng

DROP ASYMMETRIC KEY <tên khóa>

➤ **Xem** thông tin các khóa bất đối xứng có trong CSDL trên View

SYS.ASYMMETRIC_KEYS

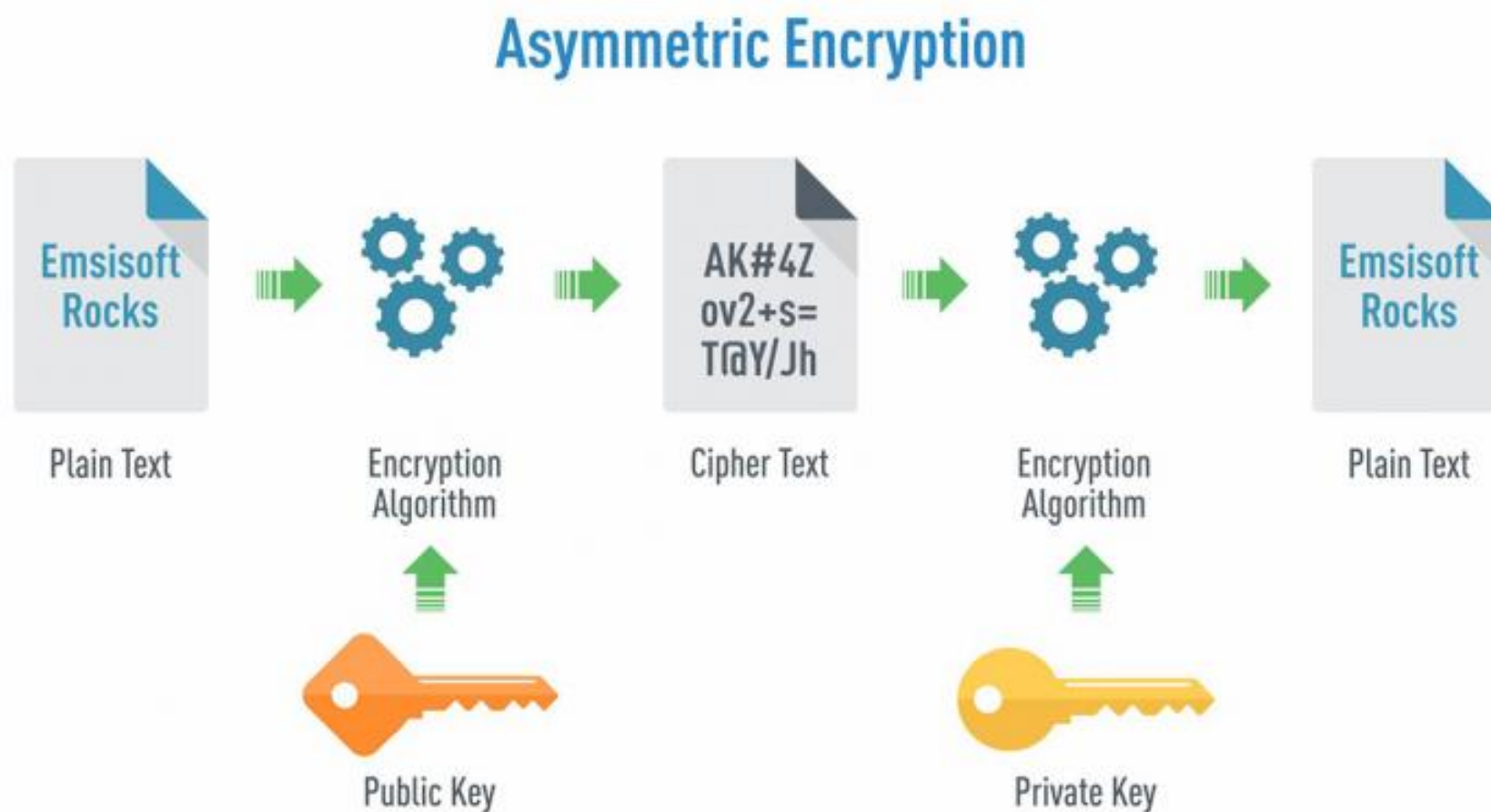
name	prin...	asymmetric_key_id	pvt_key_encryption_...	pvt_key_encryption_type_desc	thumbprint
MyKey	1	256	PW	ENCRYPTED_BY_PASSWORD	0x03A56128A039C486

algorit	algorithm_desc	key_length	sid
3R	RSA_2048	2048	0x010300000000000090200000003A56128A03

Mã hóa bất đối xứng bằng chứng chỉ

❑ Chứng chỉ (Certificate) là gì?

- Là một khóa bất đối xứng với public key được đính kèm thông tin định danh của người sở hữu private key
→ Tránh trường hợp mạo danh



Mã hóa bất đối xứng bằng chứng chỉ

- Tạo chứng chỉ

CREATE CERTIFICATE <tên_chứng_chỉ>	
AUTHORIZATION <chủ_sở_hữu>,	User có toàn quyền trên chứng chỉ
ENCRYPTION BY PASSWORD = 'mật khẩu'	Mã hóa bằng mật khẩu*
WITH SUBJECT = 'tên_chủ_đề',	Thông tin định danh
START_DATE = 'ngày',	Ngày bắt đầu có hiệu lực
EXPIRY_DAY = 'ngày'	Ngày hết hiệu lực

Mã hóa bất đối xứng bằng chứng chỉ

❑ Sao lưu chứng chỉ

BACKUP CERTIFICATE <tên_chứng_chỉ>	
TO FILE = 'đường_dẫn\tên_file.cer'	Chứng chỉ có đuôi là '.cer'
WITH PRIVATE KEY (FILE = 'đường_dẫn\tên_file.pvk', ENCRYPTION BY PASSWORD = 'mật_khẩu_sao_lưu', DECRYPTION BY PASSWORD = 'mật_khẩu_tạo')	Private key có đuôi là '.pvk' Mật khẩu bảo vệ file sao lưu Mật khẩu của chứng chỉ khi tạo

Mã hóa bất đối xứng bằng chứng chỉ

--Tạo chứng chỉ



```
CREATE CERTIFICATE MyCert1  
ENCRYPTION BY PASSWORD = 'p@sswOrdOrCert1'  
WITH SUBJECT = 'Certificate for demo',  
START_DATE = '2019-05-01', EXPIRY_DATE = '2020-01-01'
```

--Sao lưu chứng chỉ

```
BACKUP CERTIFICATE MyCert1 TO FILE = 'E:\Backup\MyCert1.cer'  
WITH PRIVATE KEY ( FILE = 'E:\Backup\MyCert1_PrivateKey.pvk',  
ENCRYPTION BY PASSWORD = 'p@sswOrdOrb@ckupCert1',  
DECRYPTION BY PASSWORD = 'p@sswOrdOrCert1')
```

> This PC > Local Disk (E:) > Backup

Search Backup

Name	Date modified	Type	Size
 MyCert.cer	24/05/2019 8:38 CH	Security Certificate	1 KB
 MyCert_PrivateKey.pvk	24/05/2019 8:38 CH	PVK File	2 KB

Mã hóa bất đối xứng bằng chứng chỉ

- Tạo chứng chỉ mới từ file sao lưu

CREATE CERTIFICATE <tên_chứng_chỉ>	
FROM FILE = 'đường_dẫn\tên_file.cer'	Đường dẫn tới file sao lưu chứng chỉ
WITH PRIVATE KEY (FILE = 'đường_dẫn\tên_file.pvk', DECRYPTION BY PASSWORD = 'mật_khẩu_sao_lưu', ENCRYPTION BY PASSWORD = 'mật_khẩu_mới')	Đường dẫn tới file sao lưu private key Mật khẩu khi sao lưu Mật khẩu để bảo vệ chứng chỉ mới

Mã hóa bất đối xứng bằng chứng chỉ

- Tạo chứng chỉ mới từ file sao lưu

```
CREATE CERTIFICATE MyCert2  
FROM FILE = 'E:\Backup\MyCert1.cer'  
WITH PRIVATE KEY ( FILE = 'E:\Backup\MyCert1_PrivateKey.pvk',  
    DECRYPTION BY PASSWORD = 'p@sswOrdFOrb@ckupCert1',  
    ENCRYPTION BY PASSWORD = 'p@sswOrdFOrCert2')
```

➤ Xem thông tin chứng chỉ trong CSDL trên View **SYS.CERTIFICATES**

Results		Messages					
	name	certificate_id	pvt_key_encryption_type_desc	subject	expiry_date	start_date	key_length
1	MyCert1	262	ENCRYPTED_BY_PASSWORD	Certificate for demo	2020-01-01 00:00:00.000	2019-05-01 00:00:00.000	2048
2	MyCert2	263	ENCRYPTED_BY_PASSWORD	Certificate for demo	2020-01-01 00:00:00.000	2019-05-01 00:00:00.000	2048

Mã hóa bất đối xứng bằng chứng chỉ

- **Thay đổi mật khẩu của private key**

ALTER CERTIFICATE MyCert1 WITH PRIVATE KEY (

DECRYPTION BY PASSWORD = '<old_password>' ,

ENCRYPTION BY PASSWORD = '<new_password>')

- **Cấp quyền trên chứng chỉ**

Sử dụng để mã hóa và giải mã	GRANT CONTROL ON CERTIFICATE :: <tên chứng chỉ> TO user
Chỉ được sử dụng để mã hóa	GRANT VIEW DEFINITION ON CERTIFICATE :: <tên chứng chỉ> TO user
Thay đổi	GRANT ALTER ON CERTIFICATE :: <tên chứng chỉ> TO user

Mã hóa bất đối xứng bằng chứng chỉ

- Mã hóa

```
EncryptByCert(Cert_ID('tên/định danh chứng chỉ'),  
              <plain text>)
```

- ✓ Plaintext: kiểu chuỗi hoặc nhị phân
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu chứng chỉ không tồn tại hoặc text là NULL.

Mã hóa bất đối xứng bằng chứng chỉ

- **Giải mã**

```
DecryptByCert(Cert_ID('tên/định danh khóa'),  
              <ciphertext>, <mật khẩu>)
```

- ✓ Ciphertext: kiểu nhị phân
- ✓ Mật khẩu: kiểu **NVARCHAR**
- ✓ Kiểu dữ liệu trả về:
 - VARBINARY (tối đa 8000 bytes)
 - NULL nếu chứng chỉ không tồn tại hoặc text là NULL.

Mã hóa bất đối xứng bằng chứng chỉ

--Tạo chứng chỉ

```
CREATE CERTIFICATE MyCert1  
ENCRYPTION BY PASSWORD = 'p@sswOrdOrCert1'  
WITH SUBJECT = 'Certificate for demo',  
START_DATE = '2019-05-01', EXPIRY_DATE = '2020-01-01'
```

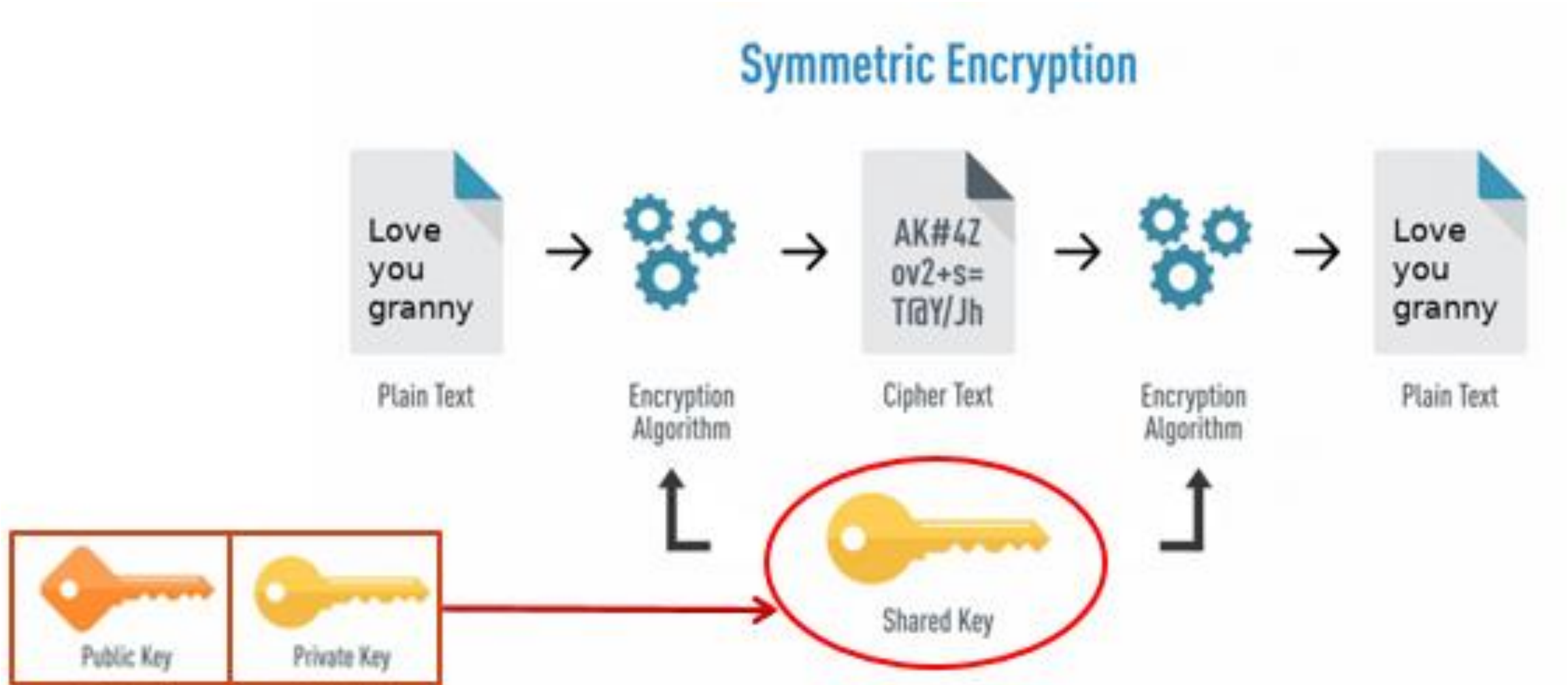
--Mã hóa

```
DECLARE @encrypted VARBINARY(8000)  
DECLARE @string NVARCHAR(50) = N'An toàn và bảo mật'  
SELECT @encrypted = EncryptByCert(Cert_ID('MyCert1'), @string)
```

--Giải mã

```
DECLARE @decrypted NVARCHAR(50)  
SELECT @decrypted = CONVERT(NVARCHAR,  
DecryptByCert(Cert_ID('MyCert1'), @encrypted, N'p@sswOrdOrCert1'))
```

Mã hóa lai



Mã hóa lại

- Chỉ trưởng đơn vị được xem lương của các thành viên trong đơn vị mình quản lý.

MaThanhVien	HoTen	GioiTinh	QueQuan	NamSinh	DonVi	ChiNhanh	CongTac	TamNghỉ	Luong
TV1	Đoàn Yến Phượng	Nữ	TP.HCM	1980	DV1	CN1	1	0	20000000
TV2	Huỳnh Hồng Thịnh	Nam	Hòa Bình	1983	DV1	CN2	0	0	25000000
TV3	Nghiêm Mạnh Chiến	Nam	Lâm Đồng	1982	DV1	CN3	0	0	30000000
TV4	Nguyễn An Bình	Nữ	Bình Định	1981	DV2	CN1	0	1	40000000
TV5	Nguyễn Minh Thảo	Nữ	Hải Dương	1977	DV2	CN2	0	0	15000000
TV6	Nguyễn Phú Hiệp	Nam	Kiên Giang	1975	DV2	CN3	0	0	28000000

Mã hóa lại

--DBA thực hiện:

--Tạo chứng chỉ

```
CREATE CERTIFICATE MyCert  
ENCRIPTION BY PASSWORD = 'Cert_P@$$wOrd'  
WITH SUBJECT = 'Certificate for THANHVIEN'
```

--Tạo 2 khóa đối xứng được mã hóa bằng chứng chỉ cho 2 trưởng đơn vị

```
CREATE SYMMETRIC KEY SymKey_DV01 WITH  
ALGORITHM = AES_192  
ENCRIPTION BY CERTIFICATE MyCert  
CREATE SYMMETRIC KEY SymKey_DV02 WITH  
ALGORITHM = AES_192  
ENCRIPTION BY CERTIFICATE MyCert
```

Mã hóa lại

--Cấp quyền cho trưởng đơn vị được sử dụng khóa và chứng chỉ

```
GRANT CONTROL ON SYMMETRIC KEY :: SymKey_DV01  
TO TV01
```

```
GRANT CONTROL ON SYMMETRIC KEY :: SymKey_DV02  
TO TV04
```

```
GRANT CONTROL ON CERTIFICATE :: MyCert TO  
Role_TruongDonVi
```

--Cấp quyền cho trưởng đơn vị được xem bảng *THANHVIEN*

```
GRANT SELECT ON THANHVIEN TO Role_TruongDonVi
```

Mã hóa lại

--Mở khóa

OPEN SYMMETRIC KEY SymKey_DV01 DECRYPTION BY
CERTIFICATE MyCert

WITH PASSWORD = 'Cert_P@\$\$wOrd'

OPEN SYMMETRIC KEY SymKey_DV02 DECRYPTION BY
CERTIFICATE MyCert

WITH PASSWORD = 'Cert_P@\$\$wOrd'

Mã hóa lại

--Mã hóa lương của mỗi đơn vị bằng khóa tương ứng

```
UPDATE THANHVIEN
```

```
SET ENCRYPT_LUONG = EncryptByKey(Key_GUID('SymKey_DV01'),  
CONVERT(VARCHAR(20), LUONG), 1, N'Doàn Yến Phượng')
```

```
WHERE DONVI = 'DV01'
```

```
UPDATE THANHVIEN
```

```
SET ENCRYPT_LUONG = EncryptByKey(Key_GUID('SymKey_DV02'),  
CONVERT(VARCHAR(20), LUONG), 1, N'Nguyễn An Bình')
```

```
WHERE DONVI = 'DV02'
```

--Đóng khóa

```
CLOSE SYMMETRIC KEY SymKey_DV01
```

```
CLOSE SYMMETRIC KEY SymKey_DV02
```

```
ALTER TABLE THANHVIEN DROP COLUMN LUONG --Bỏ cột lương
```

Mã hóa lại

```
EXECUTE AS USER = 'TV01'
```

```
OPEN SYMMETRIC KEY SymKey_DV01 DECRYPTION BY CERTIFICATE MyCert  
WITH PASSWORD = 'Cert_P@$wOrd'
```

```
SELECT MA_THANHVIEN, HOTEN, DONVI, ENCRYPT_LUONG,  
CONVERT(VARCHAR(20), DecryptByKey(ENCRYPT_LUONG, 1, N'Đoàn Yến  
Phượng')) AS DECRYPT_LUONG  
FROM THANHVIEN
```

```
CLOSE SYMMETRIC KEY SymKey_DV01  
REVERT
```

Results

Messages

	MA_THANHVIEN	HOTEN	DONVI	ENCRYPT_LUONG	DECRYPT_LUONG
1	TV01	Đoàn Yến Phượng	DV01	0x002A8DD436193B45A4D3970D5C976205020000008B...	20000000
2	TV02	Huỳnh Hồng Thịnh	DV01	0x002A8DD436193B45A4D3970D5C976205020000004B...	25000000
3	TV03	Nghiêm Mạnh Chiến	DV01	0x002A8DD436193B45A4D3970D5C97620502000000DF...	30000000
4	TV04	Nguyễn An Bình	DV02	0x00F20340DB44C94A85B8F1F3B1740858020000002E...	NULL
5	TV05	Nguyễn Minh Thảo	DV02	0x00F20340DB44C94A85B8F1F3B174085802000000697...	NULL
6	TV06	Nguyễn Phú Hiệp	DV02	0x00F20340DB44C94A85B8F1F3B1740858020000009E...	NULL

Mã hóa lại

```
EXECUTE AS USER = 'TV04'
```

```
OPEN SYMMETRIC KEY SymKey_DV02 DECRYPTION BY CERTIFICATE MyCert  
WITH PASSWORD = 'Cert_P@$$wOrd'
```

```
SELECT MA_THANHVIEN, HOTEN, DONVI, ENCRYPT_LUONG,  
CONVERT(VARCHAR(20), DecryptByKey(ENCRYPT_LUONG, 1, N'Nguyễn An  
Bình')) AS DECRYPT_LUONG  
FROM THANHVIEN
```

```
CLOSE SYMMETRIC KEY SymKey_DV02  
REVERT
```

Results Messages					
	MA_THANHVIEN	HOTEN	DONVI	ENCRYPT_LUONG	DECRYPT_LUONG
1	TV01	Đoàn Yến Phượng	DV01	0x002A8DD436193B45A4D3970D5C976205020000008B7...	NULL
2	TV02	Huỳnh Hồng Thịnh	DV01	0x002A8DD436193B45A4D3970D5C976205020000004B...	NULL
3	TV03	Nghiêm Mạnh Chiến	DV01	0x002A8DD436193B45A4D3970D5C97620502000000DF3...	NULL
4	TV04	Nguyễn An Bình	DV02	0x00F20340DB44C94A85B8F1F3B1740858020000002ED...	40000000
5	TV05	Nguyễn Minh Thảo	DV02	0x00F20340DB44C94A85B8F1F3B174085802000000697...	15000000
6	TV06	Nguyễn Phú Hiệp	DV02	0x00F20340DB44C94A85B8F1F3B1740858020000009EA...	28000000