Oracle Direct Seminar

ORACLE®

実践!! セキュリティ対策 ～暗号化編～

日本オラクル株式会社

Oracle Direct

Agenda

- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

セキュリティを取り巻く現在の状況
- 米国最大規模の情報漏えい事件 -

- 報告（発表）日時
  - 2007年1月
- 漏えいデータ
  - 個人情報 約4,750万件
- 小売業者のシステムに何者かが侵入、支払用に使われたクレジットカードデータなどの個人情報が大量に盗難

被害は甚大、顧客の損害、信用問題、信頼関係の崩壊・・・

止まらない、情報漏えい…

- ✓ システム管理者による不正アクセス、情報の持ち出し
- ✓ 元社員による個人情報の持ち出し・漏えい
- ✓ 外部からの不正アクセスにより個人情報が流出
- ✓ 顧客情報を保存していた会社のノートパソコンが盗難された
- ✓ 顧客情報を大量に記録したハードディスクを紛失

ORACLE

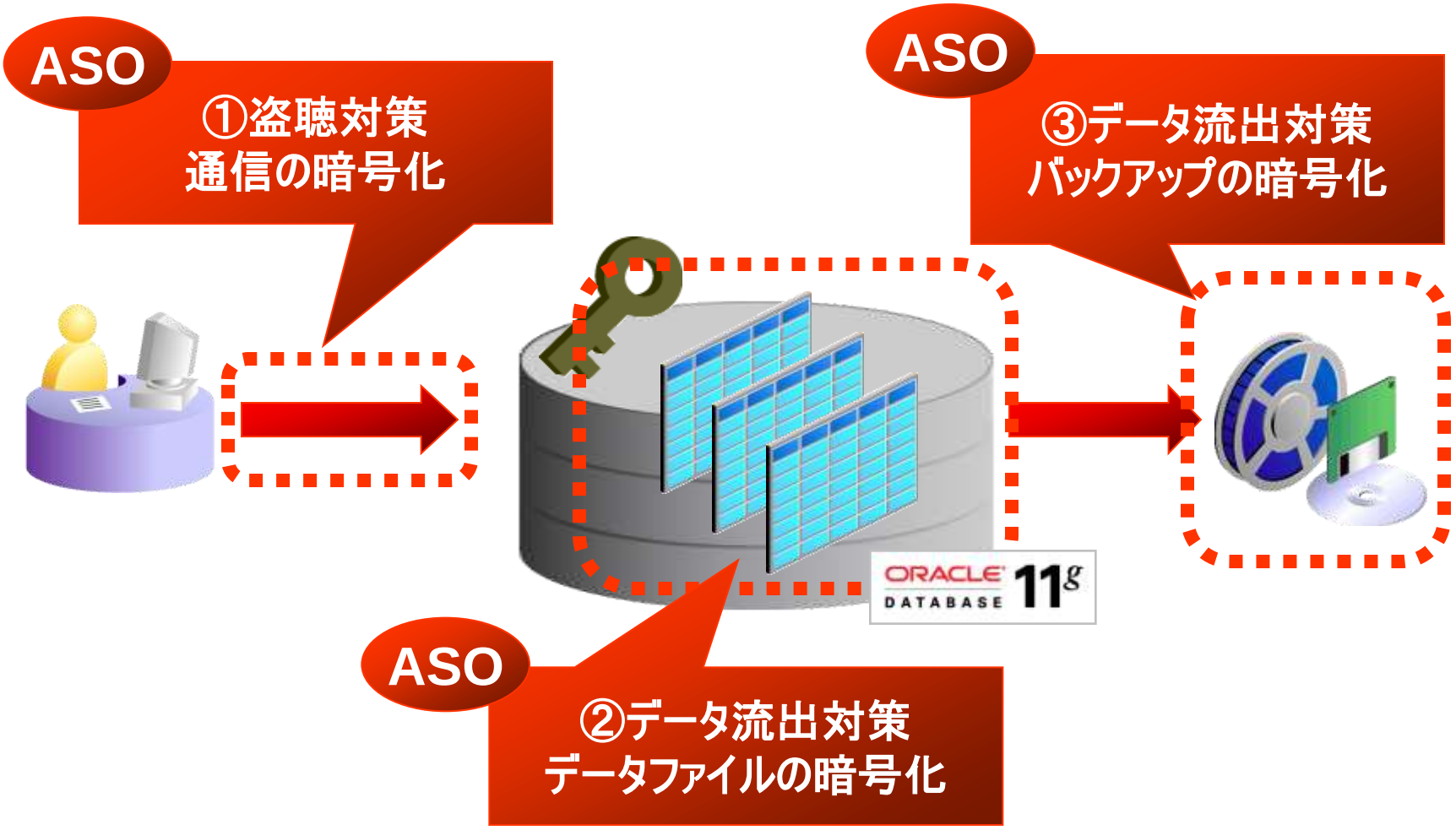
データ暗号化の必要性

- **なぜ重要か？「データ暗号化」**
  - 外部漏出しても、最悪の事態は回避可能
  - 漏出の危険性はシステム構成の中いたるところに潜む
    - クライアントとDBサーバー、アプリケーションサーバーとDBサーバーの通信
- **なぜ促進されなかったのか？「データ暗号化」**
  - 過去の暗号化には、2つの課題
    1. **実装の問題**
       - **手間のかかる実装作業**
       - **アプリケーションの大幅な書き換えが必要**
    2. **パフォーマンスの劣化**

Agenda

- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

Oracle Databaseの暗号化ソリューション
Oracle Advanced Security

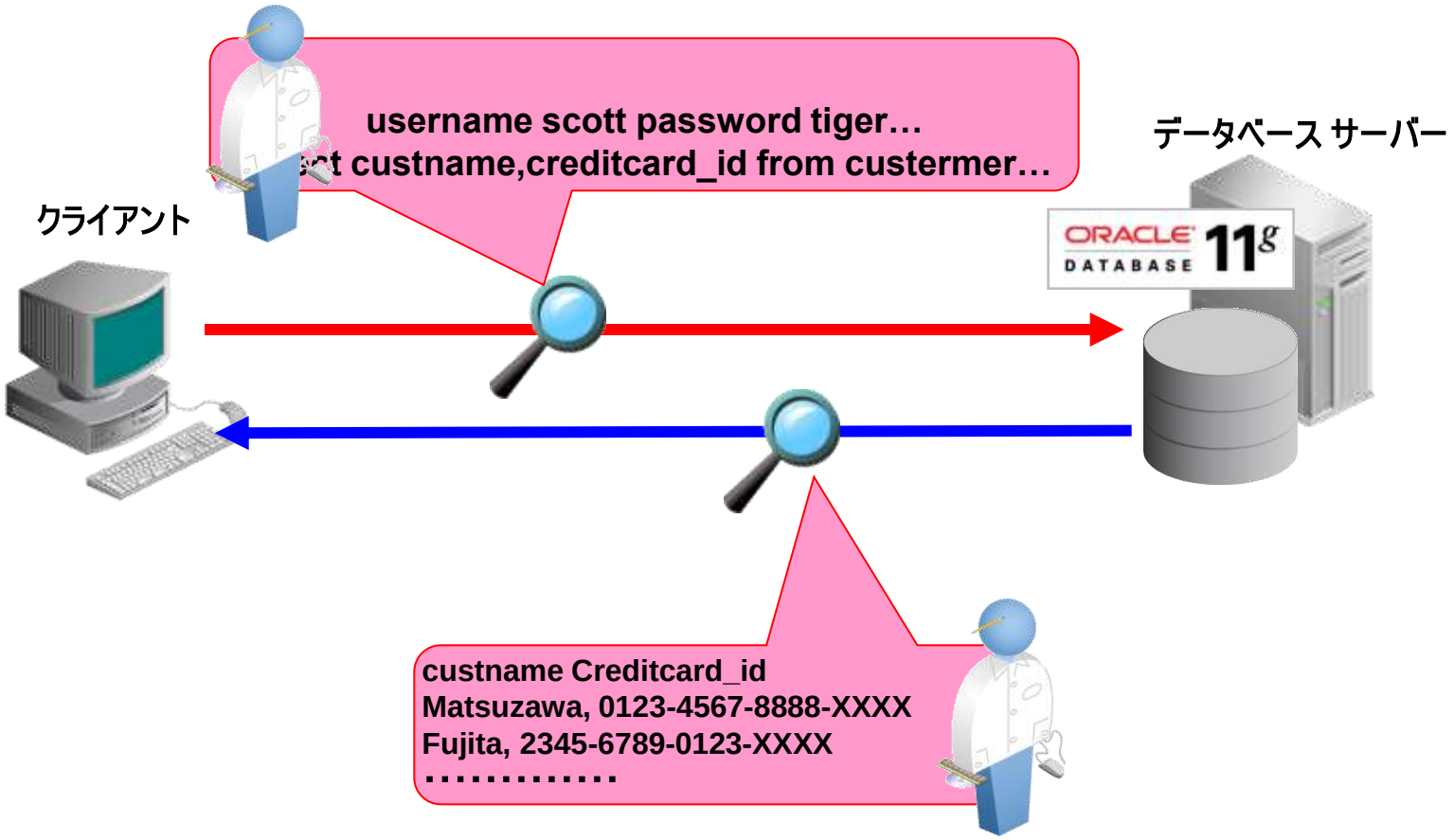


ASO: Advanced Security Option

Agenda

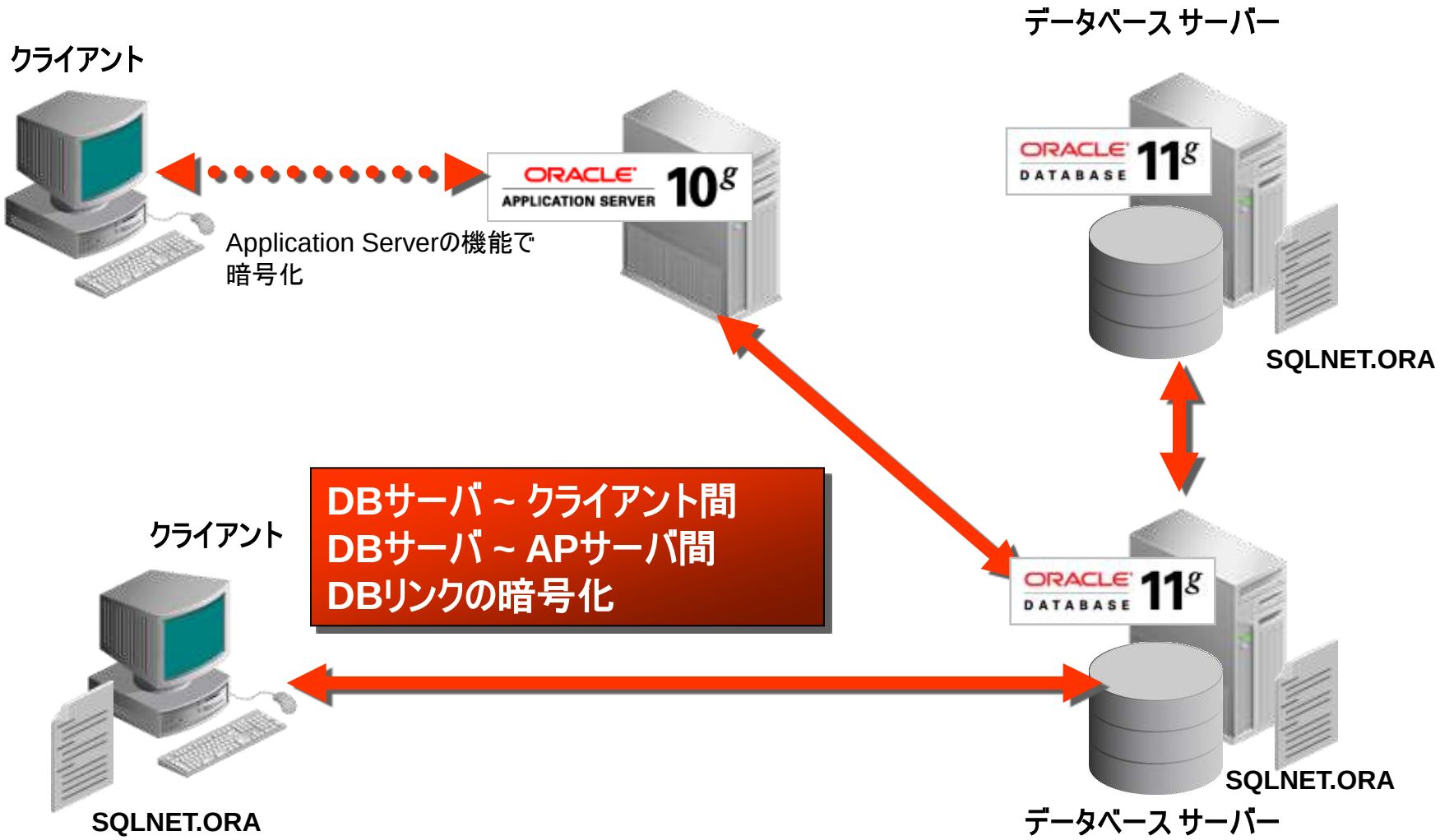
- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

①盗聴対策 – 通信の暗号化
- 盗聴による危険性への認識 -



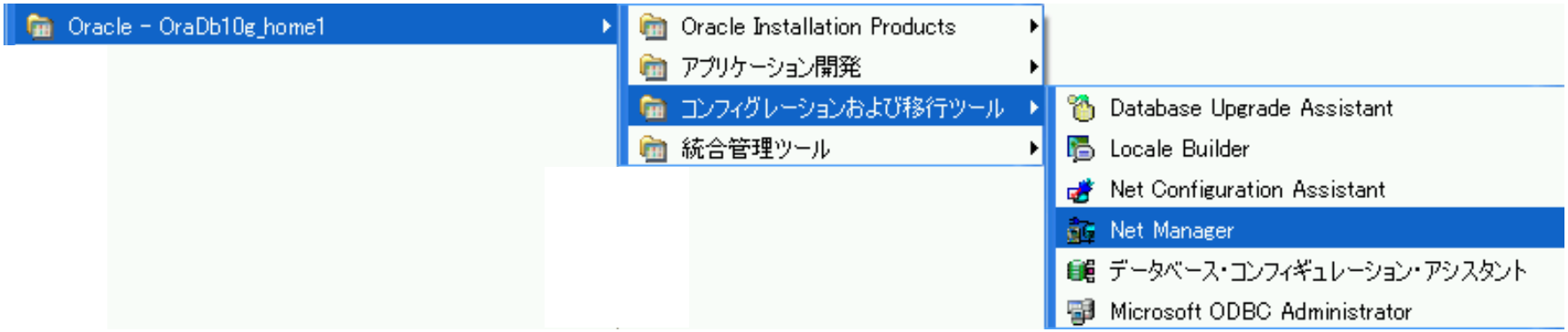
ツール利用は容易、盗聴も比較的簡単

通信の暗号化

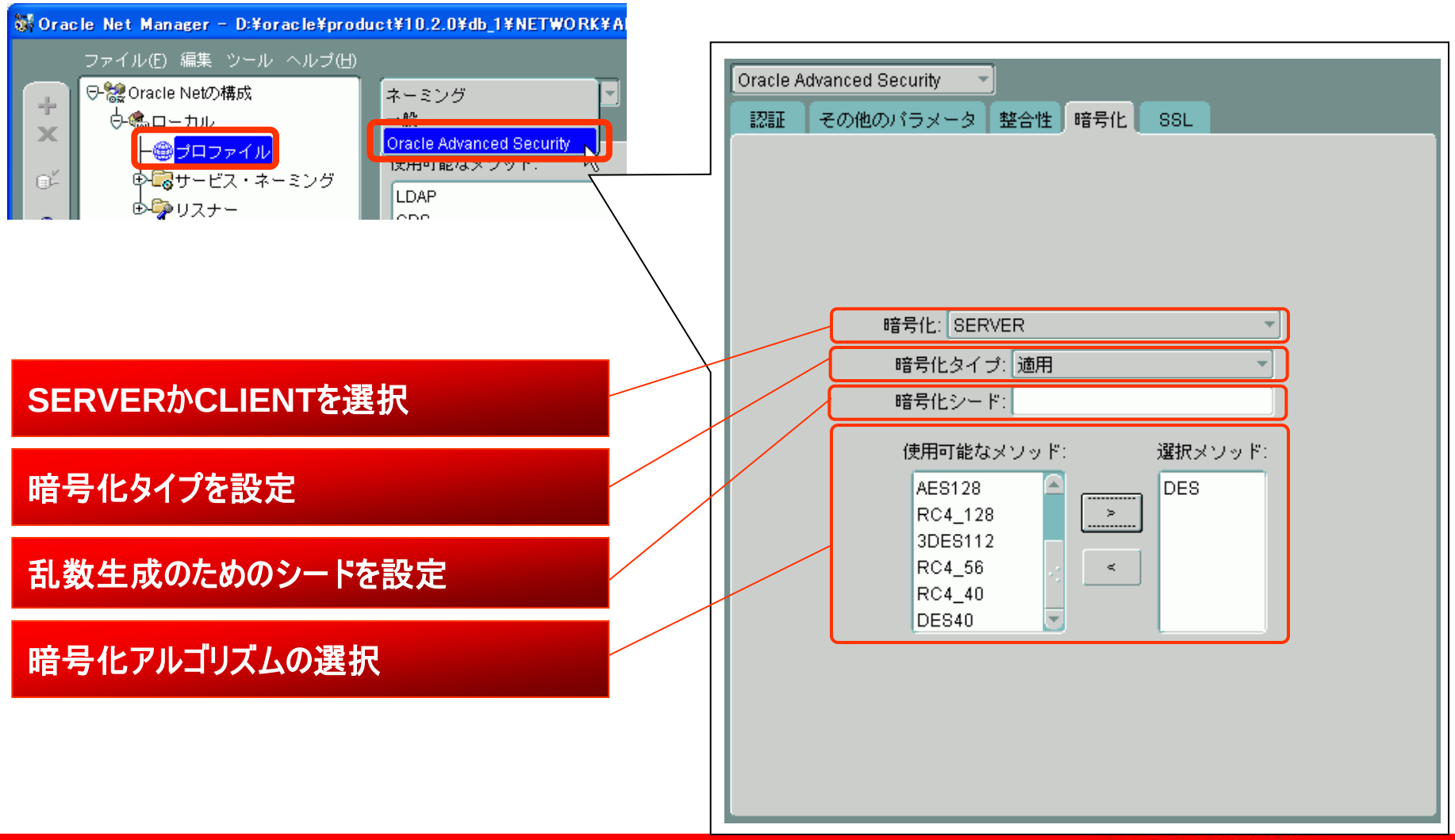


通信の暗号化の設定

- サーバとクライアントそれぞれのSQLNET.ORAを編集
- 編集にはOracle Net Managerを使用



Oracle Net Managerによる設定



- SERVERかCLIENTを選択
- 暗号化タイプを設定
- 乱数生成のためのシードを設定
- 暗号化アルゴリズムの選択

暗号化タイプについて

- 適用(accepted)　　暗号化通信が可能(デフォルト)
- 拒否(rejected)　　暗号化通信は不可
- 要求(requested)　　暗号化通信を要求
- 必要(required)　　暗号化通信が必須

通信が暗号化されるか否かは、サーバーとクライアントの暗号化タイプの設定の組み合わせで決まる

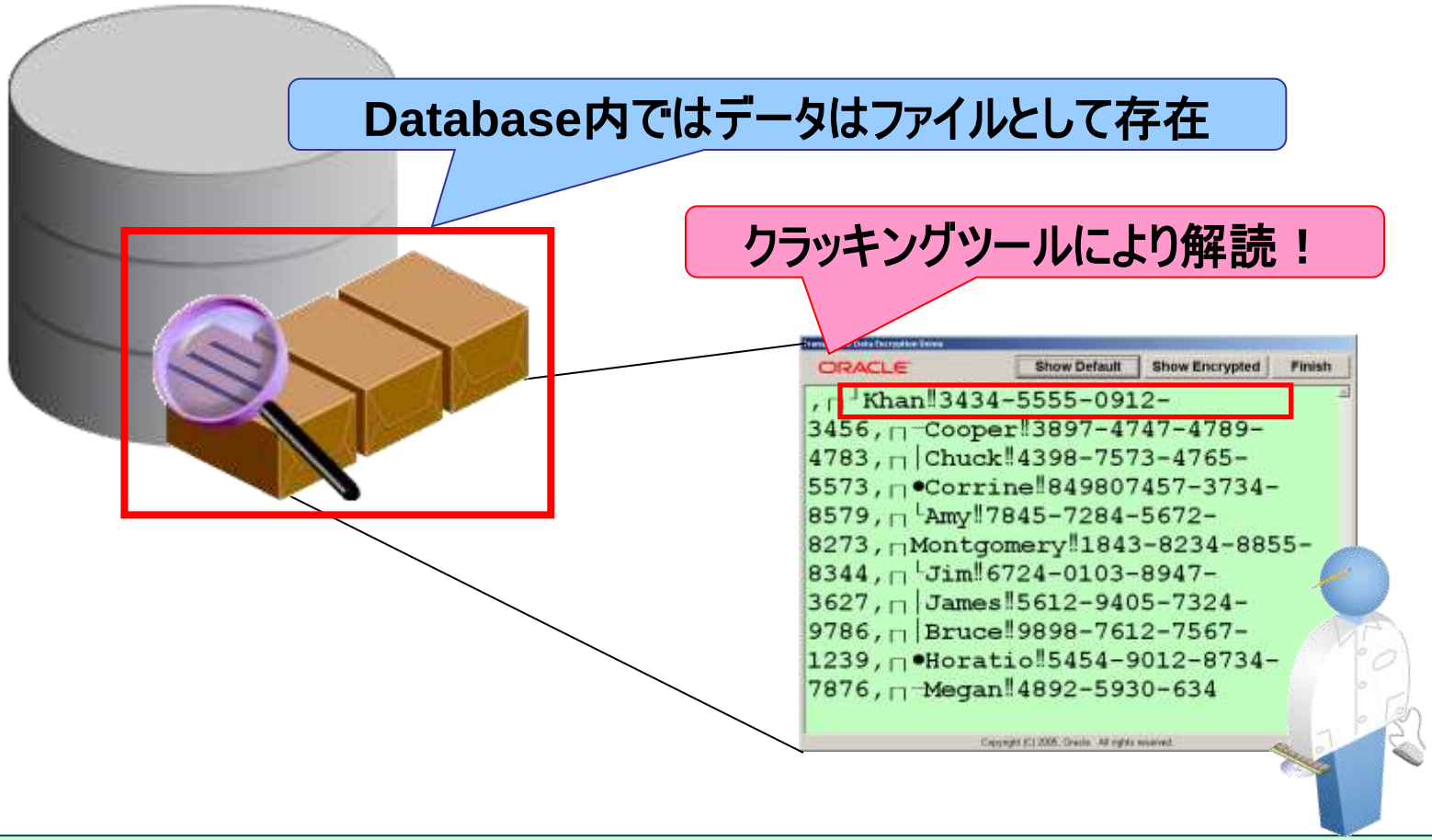
		クライアント			
		適用 (accepted)	拒否 (rejected)	要求 (requested)	必要 (required)
サーバー	適用(accepted)	暗号化なし	暗号化なし	暗号化あり	暗号化あり
	拒否(rejected)	暗号化なし	暗号化なし	暗号化なし	接続失敗
	要求(requested)	暗号化あり	暗号化なし	暗号化あり	暗号化あり
	必要(required)	暗号化あり	接続失敗	暗号化あり	暗号化あり

Agenda

- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

②データの暗号化 – データファイルの暗号化

- データファイルの盗難の危険性の認識 -



悪意を持つユーザからは格好の対象となる

データファイルの暗号化

- 標準のPL/SQLパッケージ
  - DBMS_OBFUSCATION_TOOLKIT ..... 8i ～
  - DBMS_CRYPTO ..... 10g R1～
- Transparent Data Encryption(TDE) ... 10g R2～
- LOBの暗号化 ..... 11g R1～
- 表領域の暗号化 ..... 11g R1～

ORACLE

標準のPL/SQLパッケージの使用

予め、各パッケージのプロシージャを使用して、暗号化用/復号用のファンクションを作成しておきます。

暗号化してデータを挿入
```
SQL> INSERT INTO customers(cust_id)
VALUES (encrypt_function('xxxxxx'));
```

復号してデータを取得
```
SQL> SELECT decrypt_function(cust_id)
FROM customers;
```

USER_ID	NAME	ADDRESS	CARD_ID
001	KING	TOKYO	3351-xxxx-xx
002	SCOTT	FUKUOKA	3352-xxxx-xx
003	CLARK	SAPPORO	3353-xxxx-xx

復号されたデータ

USER_ID	NAME	ADDRESS	CARD_ID
001	KING	TOKYO	mCJs8Aakm
002	SCOTT	FUKUOKA	p$hv/WiMnhf
003	CLARK	SAPPORO	V%Jsa6aUm

暗号化されて格納

暗号化パッケージの機能と比較

	DBMS_CRYPTO	DBMS_OBFUSCATION_TOOLKIT
暗号化アルゴリズム	DES、Triple-DES、AES、RC4、Triple-DES_2KEY	DES、Triple-DES
パディング方式	PKCS5、0（ゼロ）	なし
ブロック暗号連鎖モード	CBC、CFB、ECB、OFB	CBC
ハッシュ・アルゴリズム	MD5、SHA-1、MD4	MD5
ハッシュ・アルゴリズム (キーを必要とする)	HMAC_MD5、HMAC_SH1	なし
暗号化対象データ型	BLOB/CLOB/RAW	VARCHAR2/RAW

(参考)DBMS_CRYPTOの使用例

```
DECLARE
   input_string       VARCHAR2 (200) :=  'Secret Message';
   output_string      VARCHAR2 (200);
   encrypted_raw      RAW (2000);              -- stores encrypted binary text
   decrypted_raw      RAW (2000);              -- stores decrypted binary text
   num_key_bytes      NUMBER := 256/8;         -- key length 256 bits (32 bytes)
   key_bytes_raw      RAW (32);                -- stores 256-bit encryption key
   encryption_type    PLS_INTEGER :=           -- total encryption type
                         DBMS_CRYPTO.ENCRYPT_AES256
                       + DBMS_CRYPTO.CHAIN_CBC
                       + DBMS_CRYPTO.PAD_PKCS5;
BEGIN
   DBMS_OUTPUT.PUT_LINE ( 'Original string: ' || input_string);
   key_bytes_raw := DBMS_CRYPTO.RANDOMBYTES (num_key_bytes);
   encrypted_raw := DBMS_CRYPTO.ENCRYPT (
         src => UTL_I18N.STRING_TO_RAW (input_string,  'AL32UTF8'),
         typ => encryption_type, key => key_bytes_raw
      );
   decrypted_raw := DBMS_CRYPTO.DECRYPT (
         src => encrypted_raw,
         typ => encryption_type, key => key_bytes_raw
      );
   output_string := UTL_I18N.RAW_TO_CHAR (decrypted_raw, 'AL32UTF8');
   DBMS_OUTPUT.PUT_LINE ('Decrypted string: ' || output_string);
END;
```

暗号化

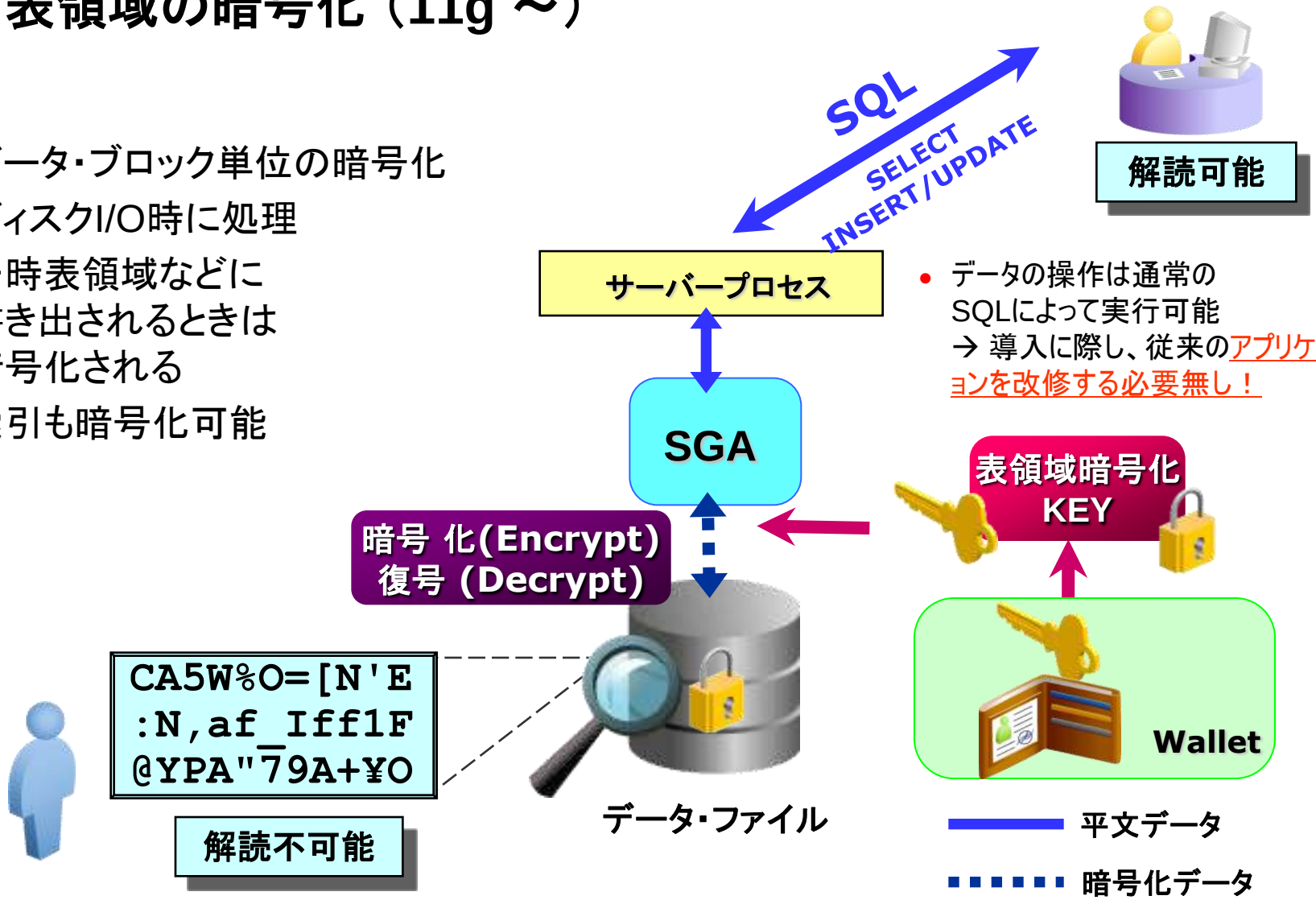
VARCHAR2→RAW

復号化

RAW→VARCHAR2

Transparent Data Encryption (TDE)
表領域の暗号化（11g ～）

- データ・ブロック単位の暗号化
- ディスクI/O時に処理
- 一時表領域などに書き出されるときは暗号化される
- 索引も暗号化可能



- データの操作は通常のSQLによって実行可能
→ 導入に際し、従来のアプリケーションを改修する必要無し！

表領域の暗号化の設定

- 暗号化された表領域の作成
  - CREATE TABLE文のSTORAGE属性にENCRYPT句を付加
  - ENCRYPTION属性で暗号化アルゴリズムを指定（デフォルト：AES128）

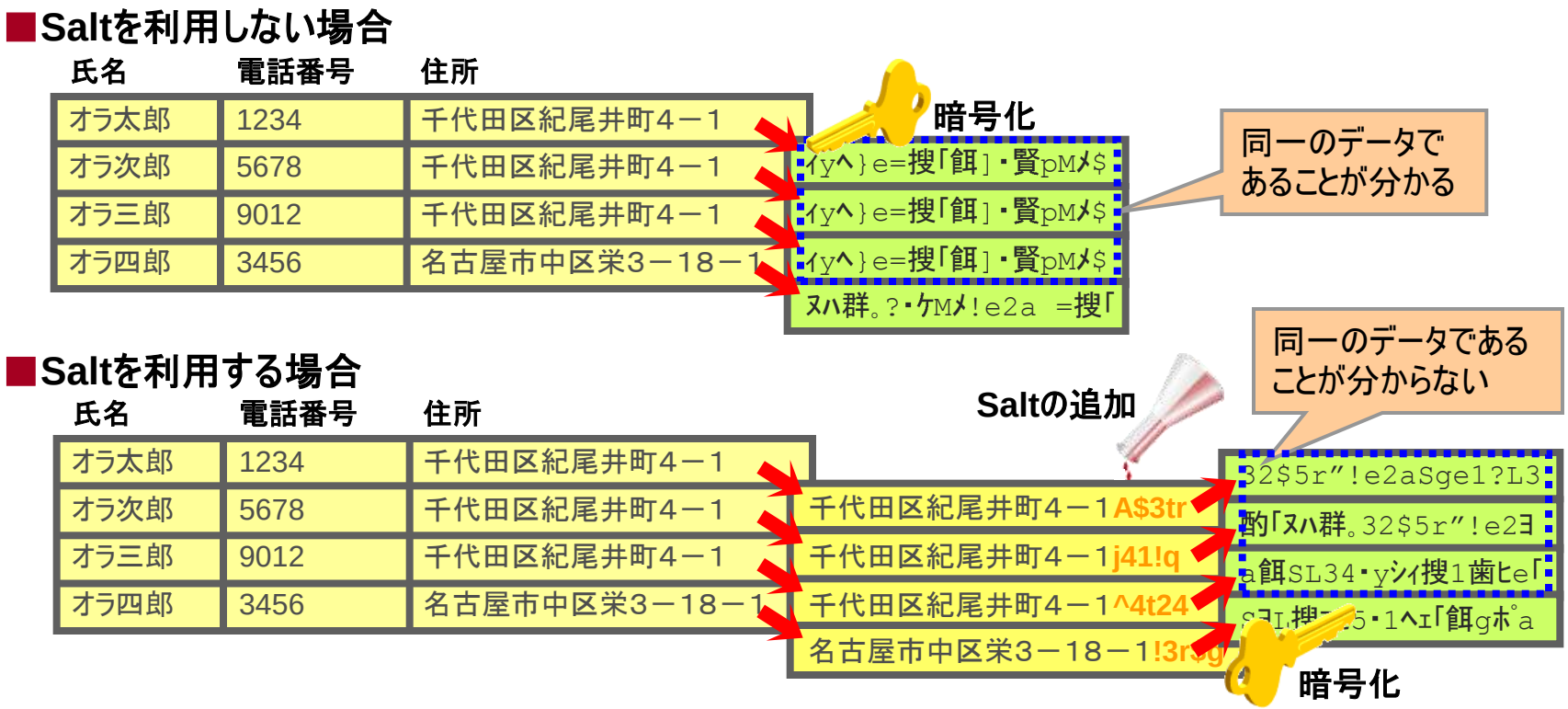
```
CREATE TABLESPACE securespace
    DATAFILE '/home/user/oradata/secure01.dbf'
    SIZE 150M
    ENCRYPTION (*1)
    DEFAULT STORAGE (ENCRYPT);
```

  - 既存の表領域は暗号化できません
  - 既存のデータを暗号化された表領域に移動することは可能です。
    - CREATE TABLE...AS SELECT...
    - ALTER TABLE...MOVE... 等

(*1)「NO SALT 」は指定できません
SALTについては後述

SALTとは

- 暗号化前のデータにランダムな文字列（Salt）を追加することで、同一データが同じ暗号文にならないようにし、セキュリティを高める



注意：索引の対象となる列では、Saltを利用することができません

LOBの暗号化（11g ～）

- TDEの機能を利用し、LOBの暗号化が可能
  - SecureFiles(11g～) の利用が必須
- 暗号化されたLOBを含む表の作成
  - CREATE TABLE文のLOB列もしくはLOB属性にENCRYPT句を付加

```
CREATE TABLE t1 (c1 CLOB ENCRYPT)
  LOB (c1) STORE AS SECUREFILE ;
```

```
CREATE TABLE t1 (c1 CLOB)
  LOB (c1) STORE AS SECUREFILE (ENCRYPT);
```

次の条件でデータの暗号化が実行

- 暗号化されるLOB : c1
- SALT (*1) : 有り
- 暗号化アルゴリズム : AES192（デフォルト）

- 既存表のLOBの暗号化
  - ALTER TABLE文のLOB列もしくはLOB属性にENCRYPT句を付加

```
ALTER TABLE t1 MODIFY (c1 CLOB ENCRYPT);
```

```
ALTER TABLE t1 MODIFY LOB(c1) (ENCRYPT);
```

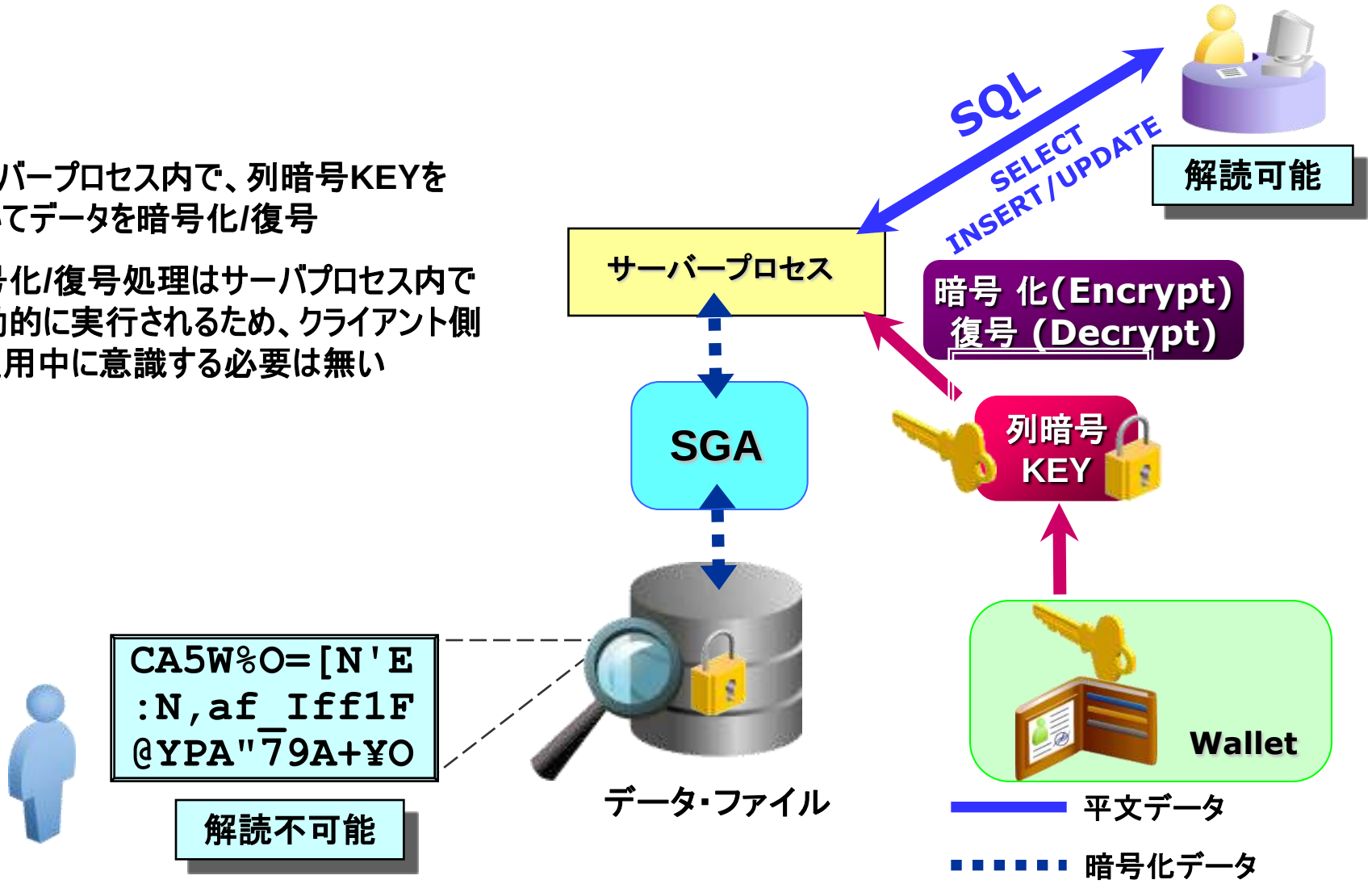
次の条件でデータの暗号化が実行

- 暗号化されるLOB : c1
- SALT (*1) : 有り
- 暗号化アルゴリズム : AES192（デフォルト）

(*1)「NO SALT 」はサポート対象外

列の暗号化

- サーバープロセス内で、列暗号KEYを用いてデータを暗号化/復号
- 暗号化/復号処理はサーバプロセス内で自動的に実行されるため、クライアント側で運用中に意識する必要は無い



データファイルの暗号化

- 標準のPL/SQLパッケージ
  - DBMS_OBFUSCATION_TOOLKIT ..... 8i ～
  - DBMS_CRYPTO ..... 10g R1～
- Transparent Data Encryption(TDE) … 10g R2～
  - 表領域の暗号化..... 11g R1～
  - 列の暗号化 ..... 10g R2～

従来の暗号化方法との比較

従来の暗号化方法

パッケージによる暗号化・復号

- DBMS_OBFUSCATION_TOOLKIT(R8.1～)
- DBMS_CRYPTO(R10.1～)

– 暗号化/復号 のたびにパッケージを呼び出してデータを処理
→ TDEに比べ追加のコストがかかる

■ パッケージを使用した場合

事前のパッケージ呼び出しによる暗号化オーバヘッドがかかりパフォーマンス面で不利

SQL文中に挿入するため、状況により改変も必要

暗号化してデータを挿入
SQL> INSERT INTO customers(cust_id)
VALUES (encrypt_function('xxxxxx'));

復号してデータを取得
SQL> SELECT decrypt_function(cust_id)
FROM customers;

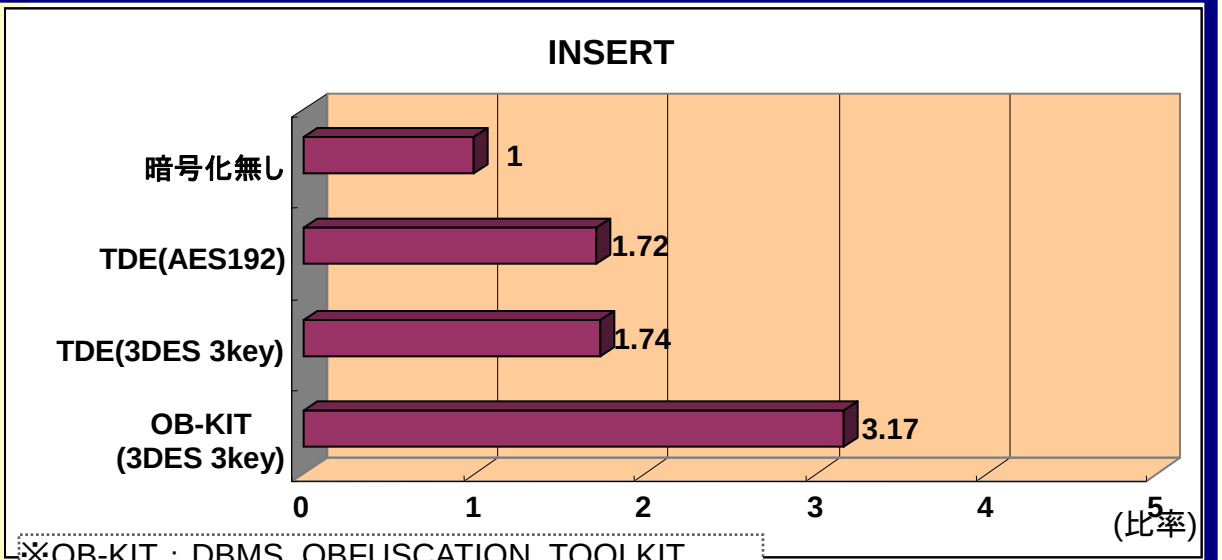
従来の暗号化方法との性能比較

- TDE vs DBMS_OBFUSCATION_TOOLKIT 性能比較結果

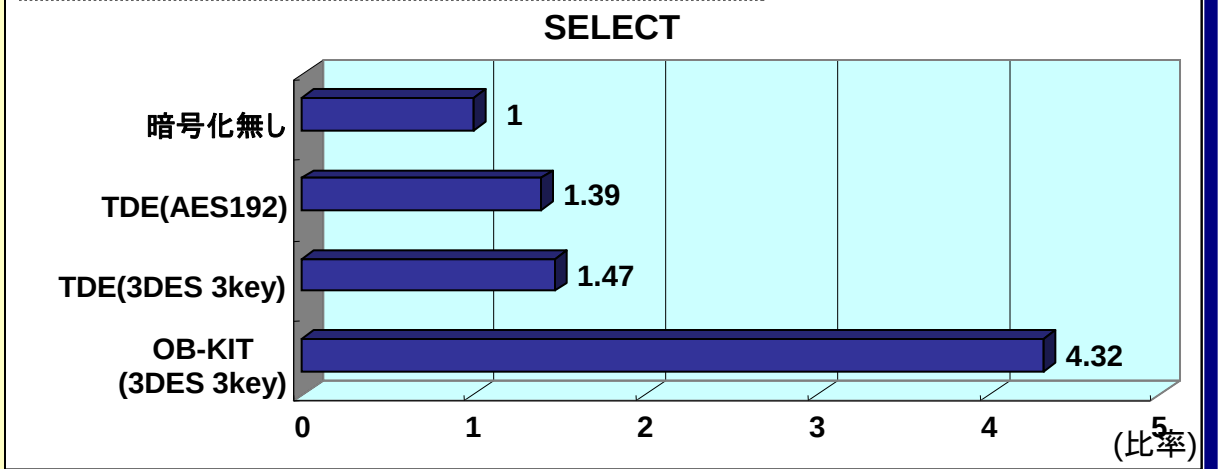
- 暗号化処理を行うことにより、処理性能が劣化する
  <TDE>
  - INSERT：約1.7倍
  - SELECT：約1.4倍

  <DBMS_OBFUSCATION_TOOLKIT>
  - INSERT：約3.2倍
  - SELECT：約4.3倍
- TDEのほうが、DBMS_OBFUSCATION_TOOLKITよりも明らかに処理性能が高い
  - INSERT：約2倍
  - SELECT：約3倍

※100万行の暗号化データの挿入、検索を行った際の処理時間を比較（シリアル処理）



※OB-KIT：DBMS_OBFUSCATION_TOOLKIT



【参考】列単位の暗号化と表領域暗号化の比較

	列単位の暗号化	表領域暗号化
暗号化の単位	レコード	ブロック
暗号化・復号化のタイミング	SQL発行時	ディスクI/O時
暗号化箇所	ディスク・メモリすべて	ディスクのみ
対象オブジェクト	表の列のみ 暗号化列に対する索引は、B-Tree索引の一意検索のみ可能	表領域内すべてのオブジェクト BITMAP索引の作成やB-Tree索引の範囲検索も利用可能

ORACLE

TDE、表領域暗号化を利用する為の事前準備

- TDE、LOBの暗号化、表領域の暗号化を利用する際には、事前に以下の設定が必要
  - Oracle Walletの作成とOPEN
  - Oracle Walletの操作（オープンとクローズ）
  - Oracle Wallet自動ログオン

Oracle Walletの作成とOPEN

- SQLNET.ORAにWALLETのロケーションを記述
  - 以下の記述を追加

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = D:¥oracle¥WALLET)
    )
  )
```

Walletファイルが格納されるディレクトリ

Oracle Walletとは？

TDEのマスター鍵をセキュアに格納するための
場所としても利用できるよう、機能が拡張された。

Oracle Wallet Manager というGUIツールによる管理が可能。

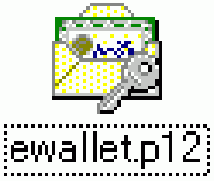
Oracle Walletの作成とOPEN

マスターキーの作成

sqlnet.oraを編集後、SQL*Plusを起動し、SYSユーザーで以下のコマンドを実行

```
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY “password”;
```

WalletをOPENするのに必要なパスワード



Walletを格納するディレクトリにWalletファイルができていることを確認

注）10g R2 でTDEを利用しており、11gにUpgradeした場合は、再度上記のコマンドを実行します

Oracle Walletの操作

- データベース起動時、Walletはクローズされている
- TDEによる暗号化列を操作、暗号化された表領域を利用するには、Walletのオープンが必要

- Walletのオープン（SYSユーザー）

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "password";
```

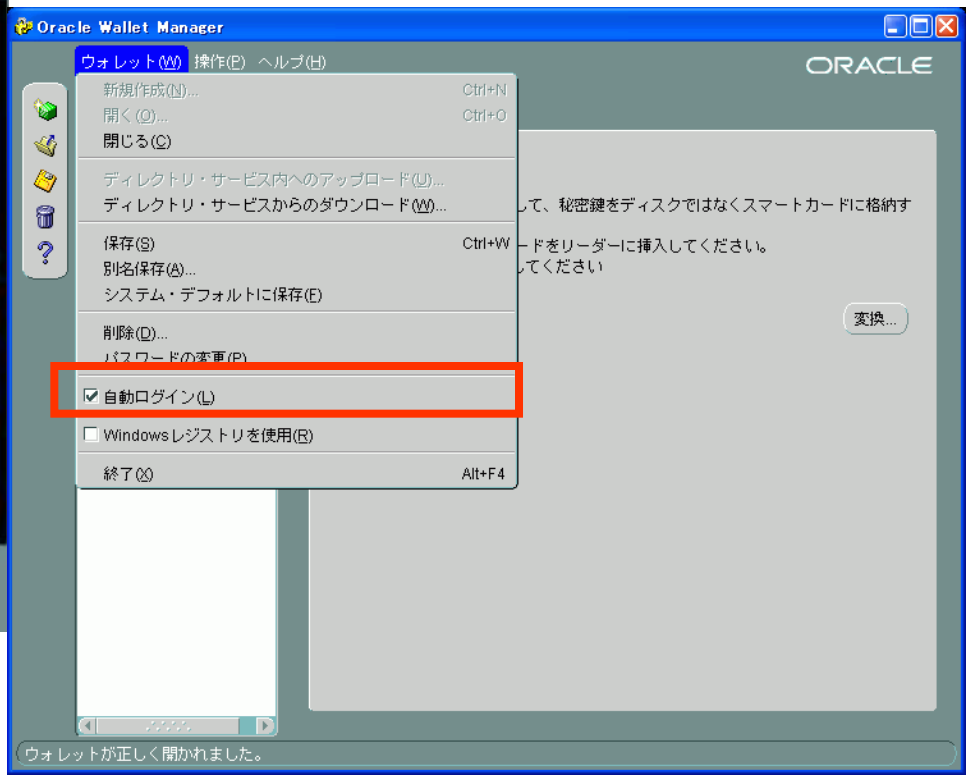
Wallet作成時に
指定したパスワード

- Walletのクローズ（SYSユーザー）

```
SQL> ALTER SYSTEM SET WALLET CLOSE;
```

Oracle Wallet 自動ログオン

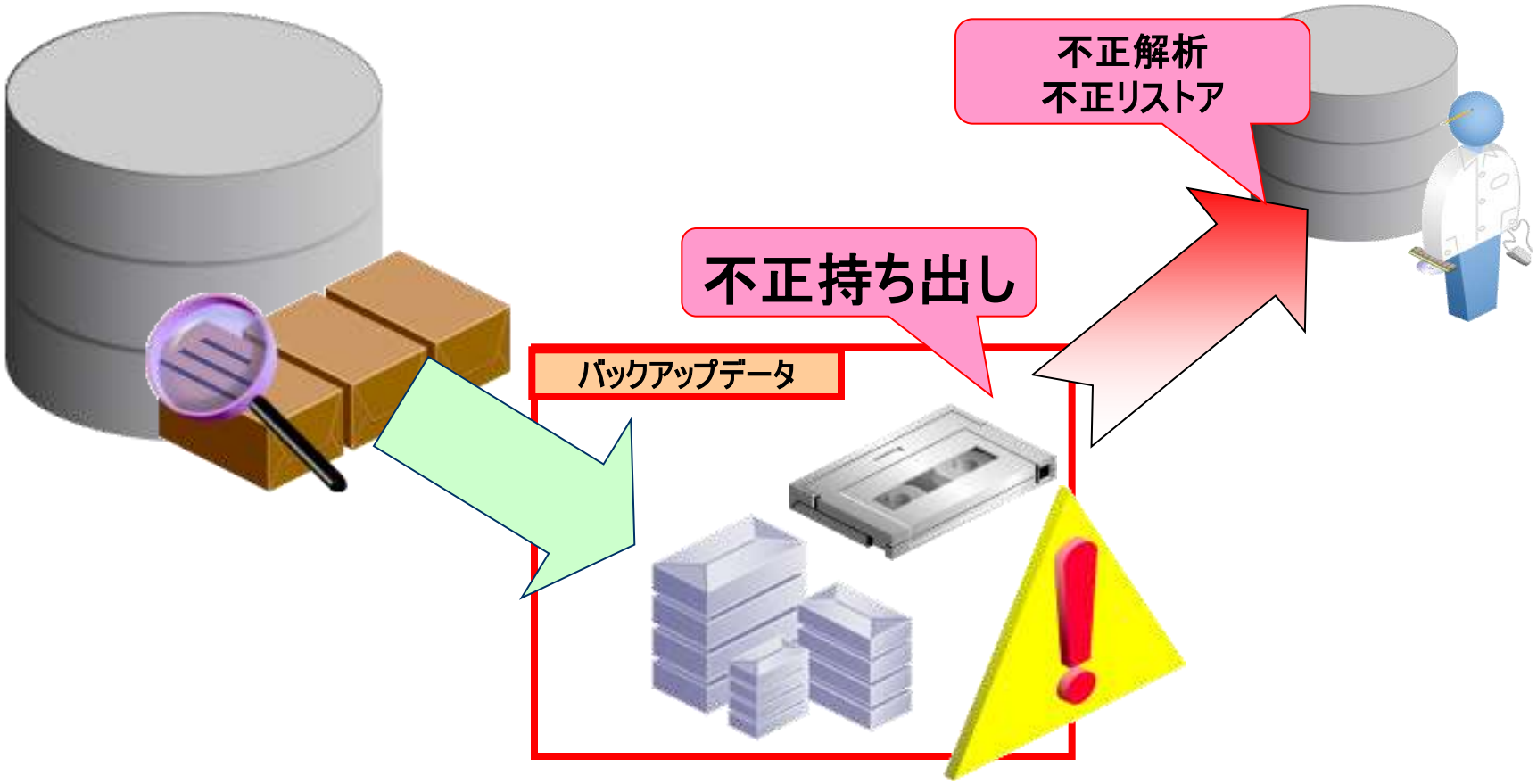
- Oracle Wallet Managerで作成したWalletを開く
- 自動ログインにチェックを入れて保存



Agenda

- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

③データの暗号化 – バックアップの暗号化
- バックアップファイル盗難の危険性の認識 -



バックアップは外部保存され、盗難にあいやすい

バックアップの暗号化

- Recovery Manager (RMAN)による
  バックアップの暗号化 ..... **10g R2 ～**
- Data Pump のダンプ・ファイル暗号化..... **11g R1～**

RMANによるバックアップの暗号化

- 透過的暗号化
  - バックアップ・リカバリ操作時にWalletを利用
  - バックアップ・リカバリ操作時に暗号化に関する特別な操作は行わない
- パスワード暗号化
  - バックアップ・リカバリ操作時にパスワードを指定
- デュアル・モード暗号化
  - バックアップ操作時にパスワードを指定
  - リストア操作時に、Walletが利用可能であれば暗号化に関する特別な操作をせずともリストア可能
  - リストア操作時にWalletが利用できなくとも、バックアップ操作時のパスワードを指定することでリストア可能

RMANによるバックアップ暗号化の設定

- 永続設定
  - Configure ENCRYPTION
  - 一度実行すると、後続処理では常に有効化される
  - 透過的暗号化では永続設定必須
- 実行時設定
  - set ENCRYPTION
  - パスワード暗号化、デュアル・モード暗号化では実行時にパスワードを設定
  - 必要に応じて永続設定を上書き可能

RMANによるバックアップ暗号化の単位

- RMANで作成されるバックアップ
  - バックアップ・セット（RMAN独自ファイル形式）に対して暗号化可能
  - イメージ・コピーは対象外（TED等により暗号化済）
- 暗号化の対象
  - すべてのデータベース・ファイルのバックアップ
  - 特定の表領域のバックアップ
- 暗号化のアルゴリズム
  - V$RMAN_ENCRYPTION_ALGORITHMS で確認
  - AES128、AES192、AES256

透過的暗号化

バックアップ

- TDEと同様にWalletを設定

```
RMAN> CONFIGURE ENCRYPTION FOR DATABASE ON;
```

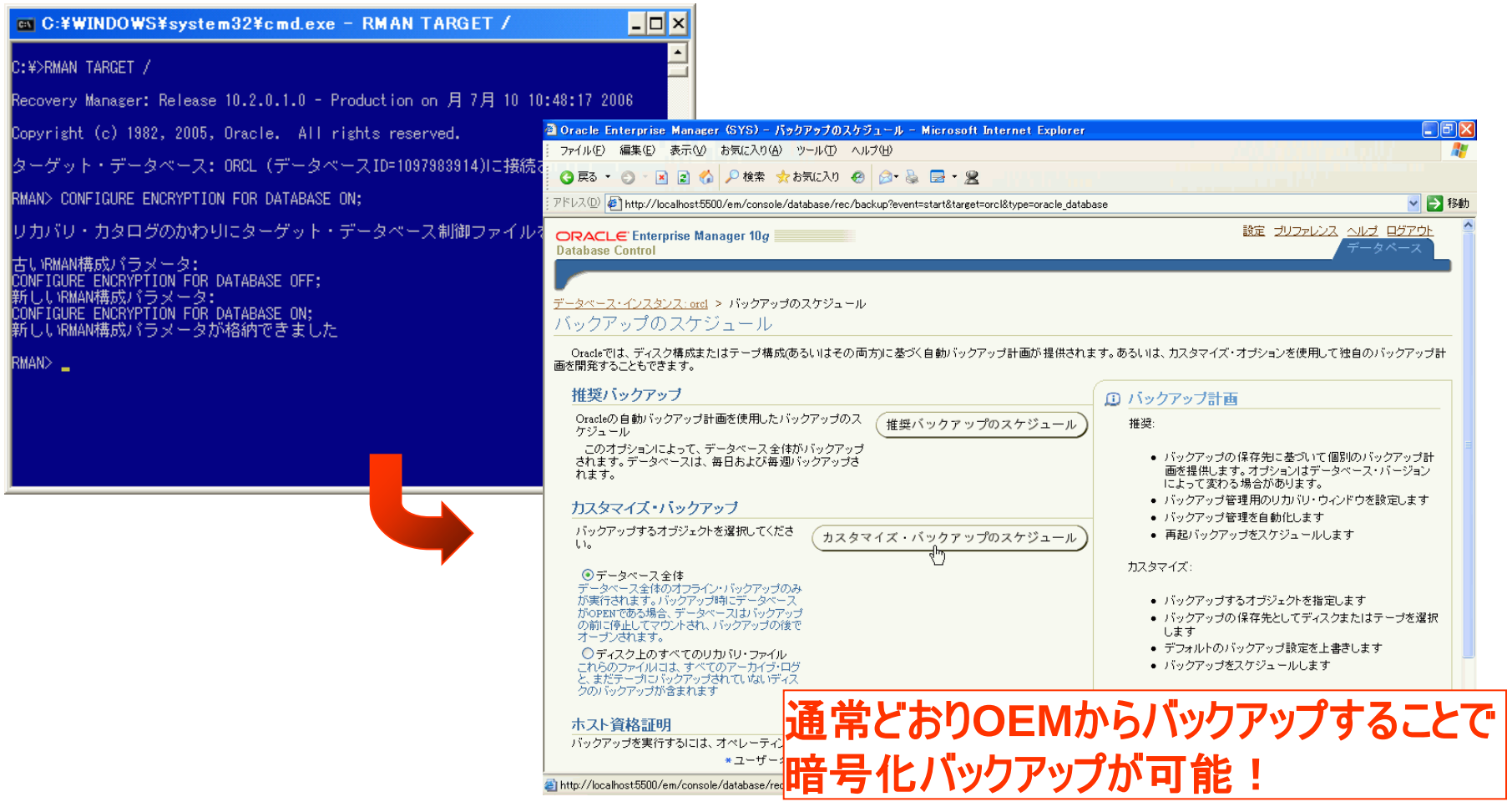
  - WalletがOPENした状態で、通常どおりバックアップを実施
  - WalletがOPENしていないとエラー

リカバリ

- WalletがOPENした状態で、通常どおりリカバリを実施
  - WalletがOPENしていないとエラー

透過的暗号化　OEMからのバックアップ

RMANで永続的暗号化を設定



パスワード暗号化

バックアップ

- Walletの設定は不要
- RMANスクリプトで、以下のコマンドを実行してバックアップを実施
  - 透過的暗号化の設定を上書き可能（データファイルの範囲）

```
RMAN> SET ENCRYPTION ON IDENTIFIED BY password ONLY;
```

リカバリ

- リカバリ時は、RMANスクリプトで以下のコマンドを実行してリカバリを実施

```
RMAN> SET DECRYPTION IDENTIFIED BY password;
```

デュアル・モード暗号化

バックアップ

- Walletの設定
- 透過的暗号化の設定
- RMANスクリプトで、以下のコマンドを実行してバックアップを実施
  - 透過的暗号化の設定を上書き可能（データファイルの範囲）

```
RMAN> SET ENCRYPTION ON IDENTIFIED BY password;
```

リカバリ

- Walletが使用できる環境 →パスワード不要、透過的にリカバリ可能
- Walletが使用できない環境　→パスワードモードでリカバリ可能

```
RMAN> SET DECRYPTION IDENTIFIED BY password;
```

Data Pumpのダンプファイルの暗号化（11g ～）

- RMANによるバックアップ暗号化と同様
  - 透過的暗号化
  - パスワード暗号化
  - デュアル・モード暗号化
- パラメータ指定

ENCRYPTION_MODE = TRANSPARENT	透過的暗号化
ENCRYPTION_MODE = PASSWORD ENCRYPTION_PASSWORD = *password*	パスワード暗号化
ENCRYPTION_MODE = DUAL ENCRYPTION_PASSWORD = *password*	デュアル・モード暗号化

Data Pumpのダンプファイルの暗号化の単位

- 暗号化の対象
  - パラメータ指定（ENCRYPTION）

パラメータの値	暗号化対象
ALL	すべて（デフォルト値）
DATA_ONLY	データのみ
ENCRYPTED_COLUMN_ONLY	暗号化された列のみ
METADATA_ONLY	メタデータのみ
NONE	暗号化なし

  - SecureFilesの暗号化が必要な場合は「ALL」を指定
  - 10g R2ではENCRYPTION_PASSWORDパラメータを設定し、暗号化された列のみを暗号化することが可能

- 暗号化アルゴリズム
  - パラメータ指定（ENCRYPTION_ALGORITHM）
  - AES128（デフォルト値）、AES192、AES256

[参考] Data Pump 暗号化利用例

- **EXPDP: 透過的暗号化**

```
% expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc.dmp ENCRYPTION=all ¥
ENCRYPTION_MODE=transparent
```

- **EXPDP:パスワード暗号化**（＊1）

```
% expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc.dmp ENCRYPTION=all ¥
ENCRYPTION_MODE=password ENCRYPTION_PASSWORD=123456
```

- **EXPDP:デュアル・モード暗号化**

```
% expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc.dmp ENCRYPTION=all ¥
ENCRYPTION_MODE=dual ENCRYPTION_PASSWORD=123456
```

- **IMPDP:パスワード利用する場合のみ**

```
% impdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc.dmp ¥
ENCRYPTION_PASSWORD=123456
```

（＊1）expdpでENCRYPTION_PASSWORDのみ指定しているとENCRYPTION_MODE=passwordとみなします

ORACLE

Agenda

- セキュリティを取り巻く現在の状況
- Oracle Databaseの暗号化ソリューション
  - Oracle Advanced Security
    - ① 通信の暗号化
    - ② データファイルの暗号化
    - ③ バックアップの暗号化
- まとめ
- Appendix

まとめ

- 暗号化へのニーズは高いものの、高かったハードル
  - アプリケーションの改修など
- Oracle Advanced Security
  - 簡単に暗号化を実現
  - データベースに組み込まれた暗号化ソリューション
  - 通信、データファイル、バックアップまですべてを暗号化

OTN×ダイセミ　でスキルアップ!!

・一般的な技術問題解決方法などを知りたい！
・セミナ資料など技術コンテンツがほしい！

Oracle Technology Network(OTN)を御活用下さい。

http://otn.oracle.co.jp/forum/index.jspa?categoryID=2

一般的技術問題解決にはOTN掲示版の
「データベース一般」をご活用ください

※OTN掲示版は、基本的にOracleユーザー有志からの回答となるため100％回答があるとは限りません。
ただ、過去の履歴を見ると、質問の大多数に関してなんらかの回答が書き込まれております。

http://www.oracle.com/technology/global/jp/ondemand/otn-seminar/index.html

過去のセミナ資料、動画コンテンツはOTNの
「OTNセミナー　オンデマンド コンテンツ」へ

※ダイセミ事務局にダイセミ資料を請求頂いても、お受けできない可能性がございますので予めご了承ください。
ダイセミ資料はOTNコンテンツ　オン　デマンドか、セミナ実施時間内にダウンロード頂くようお願い致します。

OTNセミナー　オンデマンド　コンテンツ

期間限定にて、ダイセミの人気セミナーを動画配信中!!
ダイセミのライブ感はそのままに、お好きな時間で受講頂けます。

OTN　オンデマンド　　検索

※掲載のコンテンツ内容は予告なく変更になる可能性があります。
　期間限定での配信コンテンツも含まれております。お早めにダウンロード頂くことをお勧めいたします。

オラクル クルクルキャンペーン

あのOracle Database Enterprise Editionが超おトク!!

おトクな買い方
オラクル5年分

- ライセンス使用期間　を5年間に設定
- 初期のライセンスコストがなんと67%OFF　！
- テクニカル・サポート価格も53%OFF　！

Oracle Databaseの
ライセンス価格を大幅に抑えて
ご導入いただけます

多くのお客様でサーバー使用期間とされる
5年間にライセンス期間を限定
- 期間途中で永久ライセンスへ差額移行
- 5年後に新規ライセンスを購入し継続利用
- 5年後に新システムへデータを移行

Enterprise Editionはここが違う!!
- 圧倒的なパフォーマンス!
- データベース管理がカンタン!
- データベースを止めなくていい!
- もちろん障害対策も万全!

この機能でこの価格
ライセンスパック

- Oracle Databaseの機能を存分に使える!
- 2ノードRAC構成も可能!
- サーバー構成によって計4種類のパックから選べる!

詳しくはコチラ

http://www.oracle.co.jp/campaign/kurukuru/index.html

Oracle Direct 0120-155-096

お問い合わせフォーム
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

ORACLE

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。
システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096
※月曜~金曜　9:00~12:00、13:00~18:00

（祝日および年末年始除く）

以上の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Transparent Data Encryption
暗号化アルゴリズム – Oracleで利用可能な方式

- 3DESほどの実績はないものの新しく、強固かつ高速なアルゴリズムであるAESの利用が推奨されている
- TDEは、デフォルトで192bit鍵によるAESを利用する
- セキュリティ／性能要件やデータの機密度に応じて、非デフォルトのアルゴリズムや鍵の長さを選択することが可能

CRYPTO / OB-KIT / TDE

名称	方式	鍵の長さ	備考
DES	共通鍵ブロック	56bit	IBMが開発し、1977年より20年間にわたってNIST（米国国立技術・技術院）の米政府標準暗号として利用
3DES	共通鍵ブロック	112/168bit (*1)	DESの処理を3回繰り返すことで高速な演算能力を持つ現代の攻撃者に対するDESの脆弱性を補う
AES 推奨	共通鍵ブロック	128/192/256bit	DESに変わる強固かつ高速な暗号化方式として2000年に発表、2001年にNISTの米政府標準暗号として正式に承認
RC4	共通鍵ストリーム	可変長 (40bit/128bitが一般的)	SSL等、ネットワーク・データの暗号化で広く利用されている方式であり、格納データの暗号化では推奨されない

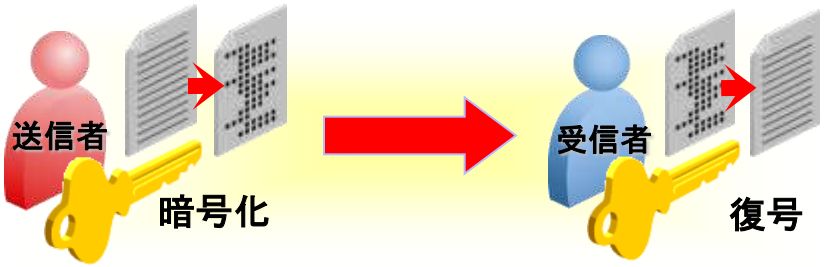
・OB-KIT: DBMS_OBFUSCATION_TOOLKIT (8i～)
・CRYPTO: DBMS_CRYPTO (10gR1～)
・TDE: Transparent Data Encryption (10gR2～)

*1: 2つの共通鍵を用いた3DES（E-D-E方式）を利用する場合鍵の長さは112bitに、3つの共通鍵を用いた3DES（E-E-E方式）を利用する場合鍵の長さは168bitになります。DBMS_OBFUSCATION_TOOLKITとDBMS_CRYPTOは両方の鍵の長さを、Transparent Data Encryptionは168bitの鍵の長さが利用可能です。

ORACLE

Transparent Data Encryption
暗号化アルゴリズム – 共通鍵と公開鍵

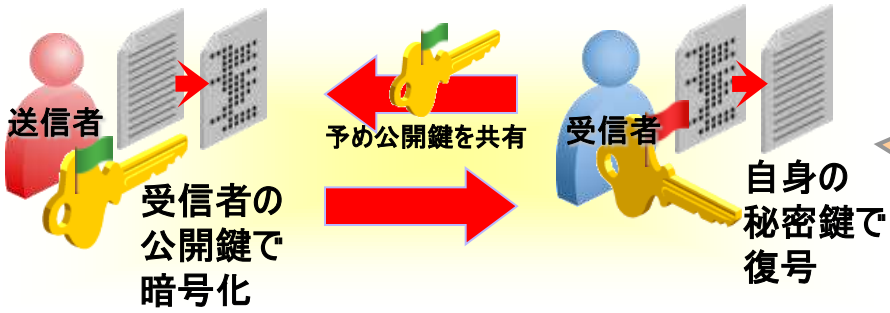
■共通鍵方式（対称鍵・秘密鍵とも呼ぶ）



単一の暗号鍵を用いて暗号化・復号を行う。鍵を奪取されてしまうと暗号化の意味をなさない反面、処理が単純化される。ブロック暗号方式とストリーム暗号方式が存在する。

代表的なアルゴリズム:
DES、3DES、RC4、AES

■公開鍵方式（非対称鍵とも呼ぶ）



予め公開されている公開鍵を用いてデータを暗号化する。復号には秘密鍵が必要で、公開鍵を利用して復号することはできない。暗号化を行う側は、秘密鍵を知らない。

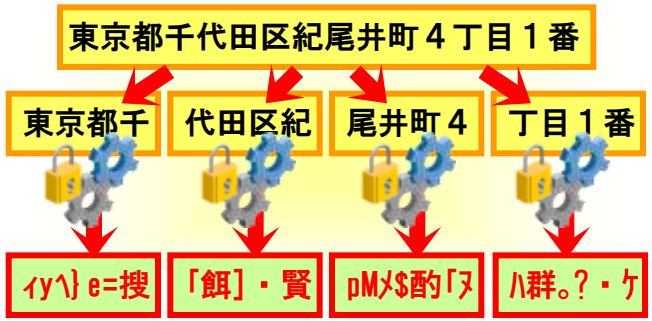
代表的なアルゴリズム:
RSA

ポイント: 格納データの暗号化では、共通鍵方式の暗号化方式を利用する

Transparent Data Encryption
暗号化アルゴリズム – ブロック暗号とストリーム暗号

■ブロック暗号方式

1. データを固定長のブロックに分割
2. ブロックごとに暗号化処理を実施

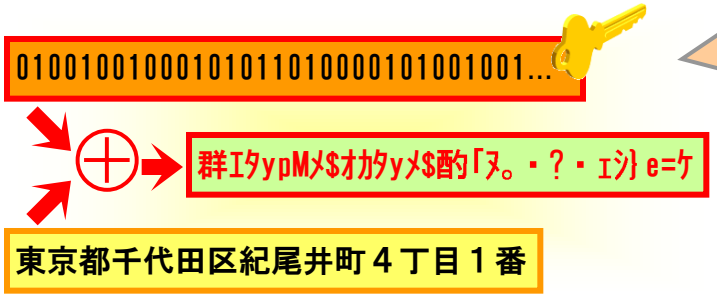


小さなデータの暗号化処理を効率的に実施することができる反面、bit誤りがブロック全体に波及してしまうという欠点を持つ。
1bitでも異なると暗号文が大きく変わるため、暗号文から平文が類推されにくい。

代表的なアルゴリズム:
DES、3DES、AES

■ストリーム暗号方式

1. 暗号鍵から擬似乱数を生成
2. 平文と擬似乱数を**1bitずつXOR**し、暗号文を生成



bit誤りに強く、大きなバイナリデータの暗号化に適している反面、同じ鍵を利用した場合にブロック暗号と比較して暗号文から平文を類推され易いというデメリットを持つ。

代表的なアルゴリズム:
RC4

ポイント: 格納データの暗号化では、ブロック暗号方式の利用が推奨される

運用上の注意および制限事項
Walletを運用する際の注意

- TDEのマスター鍵の窃取を防止し、不正なデータの復号を防止するためには、Walletをセキュアなデバイスや OSファイル・システムに格納する必要がある
  - ■Walletファイルのパーミッションを適切に設定

```
[10gR2 nemo01]$ ls -l ewallet.p12
-rw-------    1 oracle   dba      1309  8月 15 15:10 ewallet.p12
[10gR2 nemo01]$
```

- Oracleだけでなく、OSのID/パスワードの管理やアクセス制御を徹底し、Walletを保護する必要がある

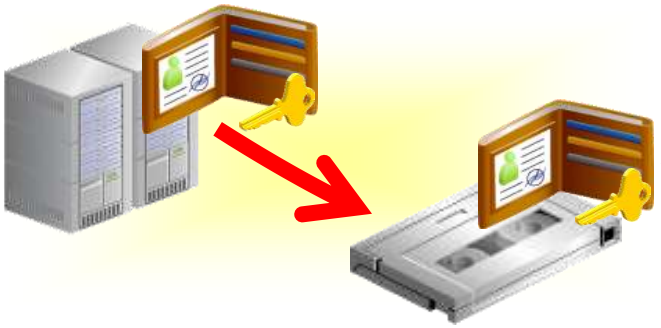
Oracle Walletとは？

従来のOracleリリースでは、公開鍵暗号を利用する際、PKCS#12形式でデジタル証明書を格納するための証明書ストアとしてOracle Walletが提供されていた。Oracle 10gR2 で、外部パスワード・ストアで利用するユーザー名・パスワードと接続記述子の組み合わせや、TDEのマスター鍵をセキュアに格納するための場所としても利用できるよう、機能が拡張された。Oracle Wallet Manager というGUIツールによる管理が可能。

運用上の注意および制限事項
暗号データの可用性を高めるための運用方法

- 暗号化されたデータの可用性を高めるためには、Wallet関連ファイル自体の可用性を高めるとともに、インスタンス自動再起動時にもデータがアクセスできるよう、手動によるWalletのオープン処理の必要性をなくす必要がある

■Wallet関連ファイルの物理バックアップを必ず取得する



Walletのバックアップを取得する場合は、sqlnet.ora にて指定した ENCRYPTION_WALLET_LOCATION あるいは WALLET_LOCATION に作成される以下のファイルを物理的にコピーする:

- **ewallet.p12（必須）**
- **cwallet.sso（自動オープン可能な Wallet を利用する場合）**

※ HA構成の場合は、データベース・ファイルに加えてこれらのファイルも共有ディスク上に配置してフェイルオーバーの対象とする

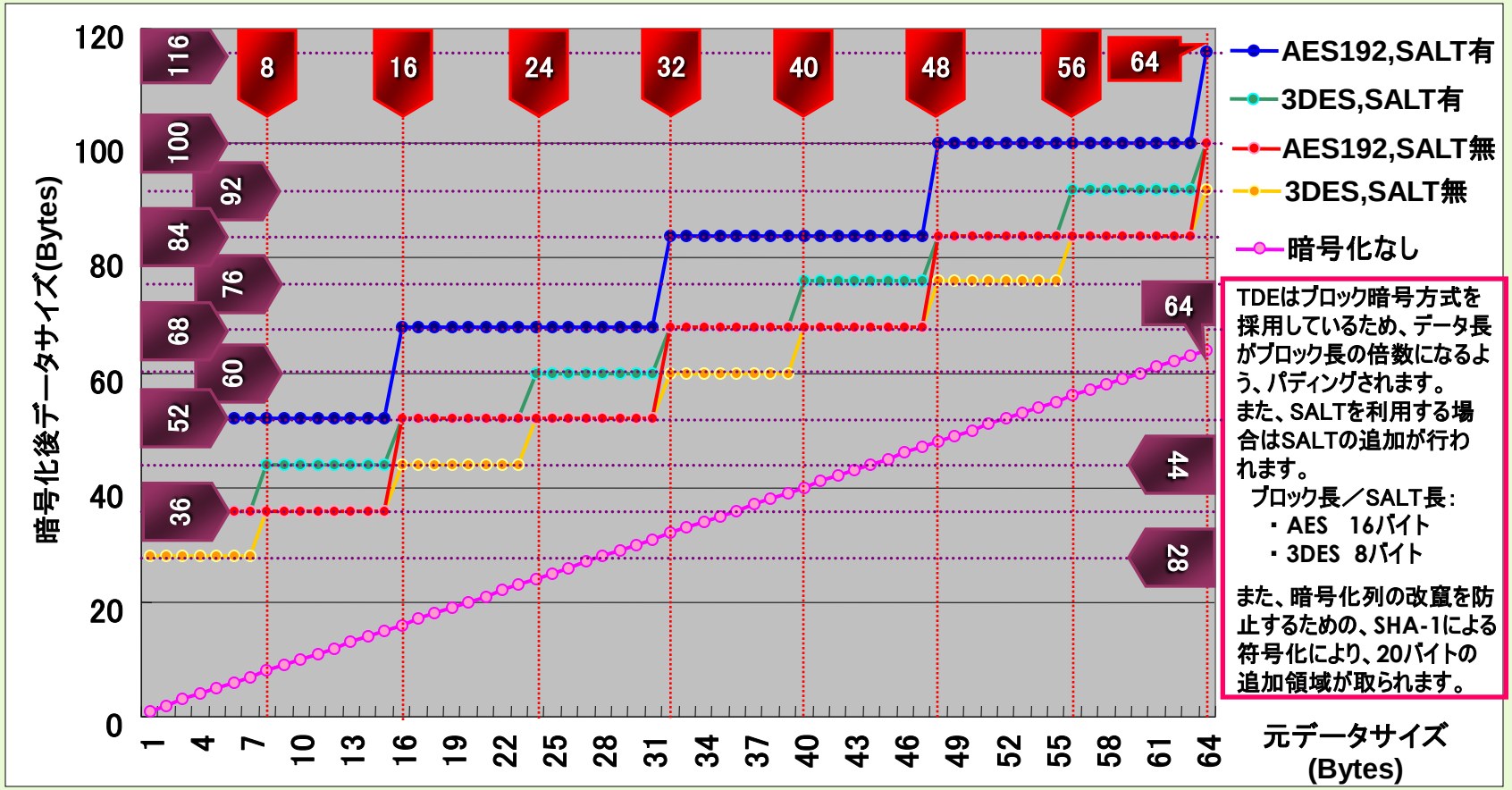
■Walletのオープン処理を安全かつ自動的に実施するための方法

- ✕ **手動によるWalletのオープン** ⇒ 障害発生の度に手動でWalletのオープンが必要
- ✕ **トリガーや運用スクリプトを利用した自動オープン** ⇒ 鍵の管理が課題となる
- ○ **Auto-loginを利用する** ⇒ 鍵管理の問題がなく、Walletの自動オープンが可能

運用上の注意および制限事項
データ拡張への考慮

- 暗号化方式と論理的データの拡張

- 暗号化によりデータが拡張し、通常時より多くの領域が使用される
- 新規作成時のみではなく、既存表からの移行時にも考慮が必要

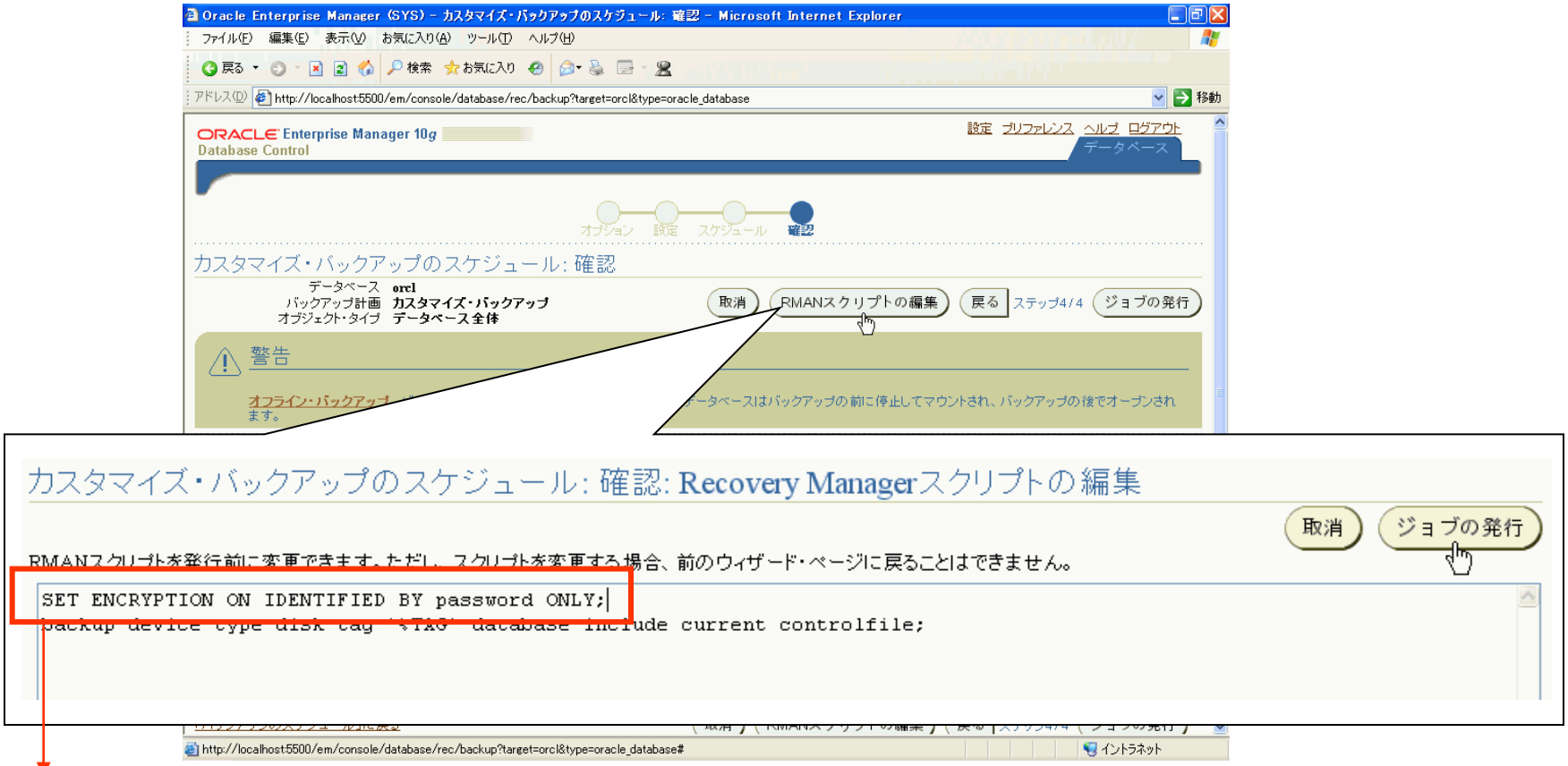


Transparent Data Encryption
従来の暗号化方法との比較【ご参考資料】

優位性 ■ > ■

	TDE 表領域の暗号化	DBMS_OBFUSCATION_TOOLKIT() DBMS_CRYPTO()
仕組み	透過的 暗号化/復号処理は全てDB上で透過的に実行される	非透過的 暗号化/復号毎にパッケージを呼び出して処理が実行される
設定方法	単純 表定義(CREATE/ALTER文)内で指定するのみ	複雑 パッケージを使用し専用ファンクションを作成、AP処理に埋め込む
設定単位	列、LOB、表領域 データ単位での設定は不可能	柔軟 データ単位での設定が可能
索引作成	可能 処理性能への影響小	不可能 処理性能への影響大
鍵の管理	不要	必要

パスワード暗号化　OEMからのバックアップ



RMANスクリプトに以下1行追加してジョブの発行

SET ENCRYPTION ON IDENTIFIED BY *password* ONLY;

列の暗号化の設定

- 暗号化列を含む表の作成
  - CREATE TABLE文の対象列にENCRYPT句を付加

```
CREATE TABLE customers(
   id          NUMBER,
   card_no VARCHAR2(10) ENCRYPT
);
```

次の条件でデータの暗号化が実行
- 暗号化される列　:CARD_NO
- SALT (*1)　:有り（デフォルト）
- 索引作成　:不可(SALT有りの場合は不可)
- 暗号化アルゴリズム:AES192（デフォルト）

- 既存表の列の暗号化
  - ALTER TABLE文の対象列にENCRYPT句を付加

```
ALTER TABLE customers MODIFY
  card_no VARCHAR2(10) ENCRYPT;
```

次の条件でデータの暗号化が実行
- 暗号化される列　:CARD_NO
- SALT (*1)　:有り（デフォルト）
- 索引作成　:不可(SALT有りの場合は不可)
- 暗号化アルゴリズム:AES192（デフォルト）