データベース暗号化ガイドライン

第 0.9 版

2011 年 9 月 21 日

データベース・セキュリティ・コンソーシアム

DB 暗号化 WG

目次

1　はじめに 1

1.1 目的 1

1.2 本ガイドラインの前提 1

1.3 本ガイドラインに関する注意事項 2

2　DB 暗号化概略 3

2.1 想定システムモデル 3

2.2 暗号化対策概要 3

2.3 脅威の定義 7

3　DB の暗号化 8

3.1 HDD 暗号化 8

3.1.1　HDD 暗号化概要 8

3.1.2　HDD 暗号化のメリット 9

3.1.3　HDD 暗号化デメリット 9

3.1.4　HDD 暗号化導入にあたって 9

3.2　DB テーブルの暗号化 10

3.2.1　DB テーブル暗号化概要 10

3.2.2　DB テーブル暗号化のメリット 11

3.2.3　DB テーブル暗号化のデメリット 12

3.2.4　DB テーブル暗号化導入にあたって 12

3.3　DB カラムの暗号化 12

3.3.1　DB カラム暗号化概要 12

3.3.2　DB カラム暗号化のメリット 14

3.3.3 DB カラム暗号化のデメリット 14
3.3.4 DB カラム暗号化導入にあたって 15
3.4 バックアップデータの暗号化 15
3.4.1 バックアップデータ暗号化概要 15
3.3.2 バックアップデータ暗号化導入にあたって 16
4 暗号鍵管理 17
4.1 ファイルとしての暗号鍵管理 18
4.1.1 ファイルとしての暗号鍵管理概要 18
4.1.2 ファイルとしての暗号鍵管理メリット 18
4.1.3 ファイルとしての暗号鍵管理デメリット 18
4.1.4 ファイルでの暗号鍵管理導入にあたって 18
4.2 専用 H/W (HSM) での暗号鍵管理 19
4.2.1 HSM での暗号鍵管理概要 19
4.2.2 HSM での暗号鍵管理メリット 20
4.2.3 HSM での暗号鍵管理デメリット 21
4.2.4 HSM での暗号鍵管理導入にあたって 21
4.3 暗号鍵のアクセス制御 22
4.3.1 暗号鍵のアクセス制御概要 22
4.3.2 暗号鍵のアクセス制御メリット 23
4.3.3 暗号鍵のアクセス制御デメリット 23
4.3.4 暗号鍵のアクセス制御導入にあたって 23
4.4 暗号鍵の世代管理 24
4.4.1 暗号鍵の世代管理概要 24
4.4.2 暗号鍵の世代管理メリット 25
4.4.3 暗号鍵の世代管理デメリット 25
4.4.4 暗号鍵の世代管理導入にあたって 25
5 通信経路の暗号化 26
5.1 DB-アプリケーション間 26

5.1.1 DB-アプリケーション間の暗号化概要 26
5.1.2 アプリケーション-DB 間の暗号化メリット 27
5.1.3 アプリケーション-DB 間の暗号化デメリット 27
5.1.4 アプリケーション-DB 間の暗号化導入にあたって 27
5.2 DB-管理者端末間 28
5.2.1 DB-管理者端末間の暗号化概要 28
5.2.2 DB-管理者端末間の暗号化メリット 28
5.2.3 DB-管理者端末間の暗号化デメリット 28
5.2.4 DB-管理者端末間の暗号化導入にあたって 28
5.3 DB-鍵管理デバイス間 29
5.3.1 DB-鍵管理デバイス間の暗号化概要 29
6 DB 暗号化導入事例 30
7 DB 暗号化ガイドライン執筆者 35

1　はじめに

1.1 目的

情報漏えいに起因した事件が社会において頻発する中、多くの重要な情報の主たる格納場所であるデータベース（以下 DB）に対するセキュリティ対策が益々重要となりつつある。システムの根幹を構成する DB に関して、高度なセキュリティに係わる標準的な技術/手法の確立を図っていくことは、安心/安全な利用環境を維持したシステムを構築/運用していく上で急務である。

上記については、オンプレミスな DB にとどまらず、近年のクラウド市場における DB セキュリティなどにおいて重要課題として議論がなされている。

昨今の DB セキュリティにおいて、DB 暗号化は情報漏えいに対する抜本的な対策であり、正しい運用を実装することで、リスクの低減もしくは漏えい防止を実現できるものである。

しかし DB の暗号化については、その手間や煩雑な運用イメージやパフォーマンスの低下、心理的不安要素などから敬遠されてきた。

本ガイドラインは、DB 暗号化に対する正しい知識と運用方法を示し、各企業/団体の「DB セキュリティ対策」の導入の為の指標としてセキュリティ向上に貢献することを目的とするものである。

1.2 本ガイドラインの前提

本ガイドラインでは、DB 暗号化を検討するにあたり、次の想定をもとに検討を進める。

- 本ガイドラインで検討する暗号化対策は特定の DB 暗号化機能での対策に限定せず、DB の暗号化を実装する上で必要となる事象について言及する。
- 本ガイドラインの利用者は、DB セキュリティに携わる責任者を想定する。
- 本ガイドラインにおける DB の定義は、現在もっとも多くのシステムで稼動している RDB とする。
- 本ガイドラインにおける「特権ユーザ」とは、DB 管理権限を有する DBA、一般的な管理責任者を想定する。
- 他のセキュリティ対策については、既存のガイドラインを参照する。

1.3 本ガイドラインに関する注意事項

○著作権の所在

本ガイドラインの版権は、データベース・セキュリティ・コンソーシアム（DBSC）に属する。

○利用制限

本ガイドラインの販売は禁止する。それ以外の本ガイドを利用したサービス提供に関しては一切制限しない。

○引用元の明記

本ガイドラインの全文もしくは一部を引用する場合には、必ず引用元として「データベースセキュリティガイドライン」を明記する。営利目的、非営利目的の区別はない。

①ガイドラインの全部あるいは一部をそのまま、使用する場合：

【出典】「データベース暗号化ガイドライン(1.0版)」

データベース・セキュリティ・コンソーシアム（DBSC）

http://www.DB-security.org/

---

②ガイドラインを一部加工して、使用する場合：

【参考文献】「データベース暗号化ガイドライン(1.0版)」

データベース・セキュリティ・コンソーシアム（DBSC）

http://www.DB-security.org/

○免責事項

本ガイドラインを利用したことによって生じるいかなる損害に関しても、DBSCは一切責任を負わないものとする。

○利用時窓口

本ガイドラインを報道、記事などメディアで用いる場合には、DBSC 事務局（info@DB-security.org）まで連絡する。

2　DB 暗号化概略

2.1 想定システムモデル

本ガイドラインでは、インターネットとイントラネットからアクセスされるWeb 3 階層モデルを想定システムモデルとする。
この想定システムモデルでは、DBサーバはWeb/アプリケーションサーバからのクエリアクセス、またDBAが操作する管理端末からのクエリアクセスだけではなく、*物理的なDBへのアクセスも考慮する。

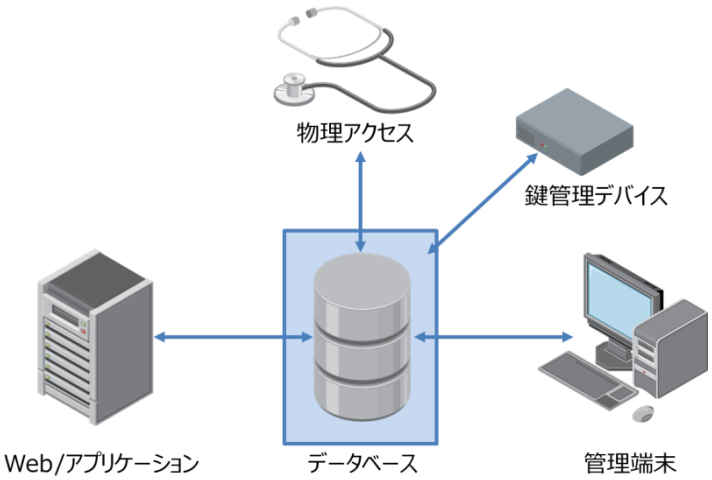


図1．想定システムモデル

*物理的なDBへのアクセス… HDDやバックアップテープに対する盗難、メモリダンプの盗聴など、DBMSが動作しているH/Wに関わるすべての物理的なアクセスが含まれる。

2.2 暗号化対策概要

想定システムモデルに対する全体のセキュリティ対策の概要を以下に示す。全体のセキュリティ対策は大きく以下の種類を想定する。

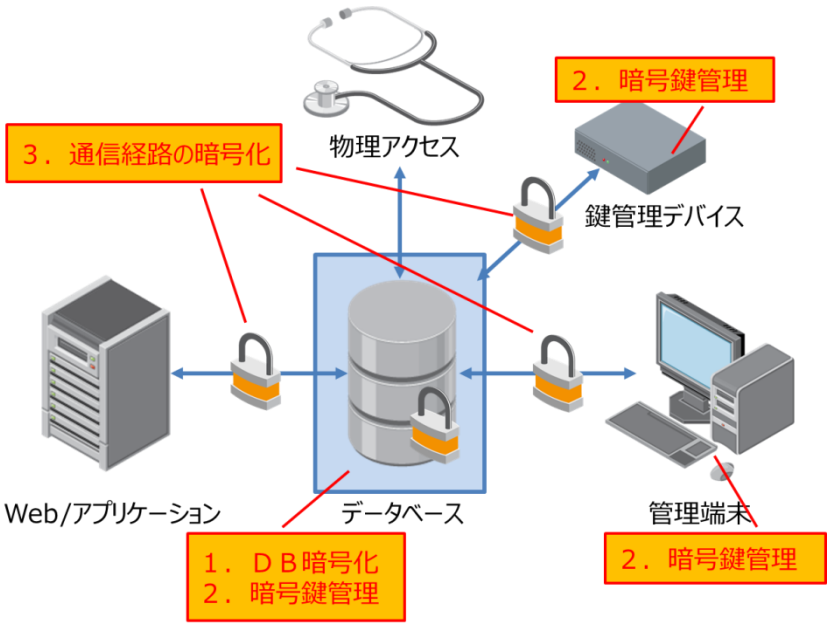


図2．暗号化対策概要

上記図にあるように、データの暗号化だけではなく、暗号鍵へのアクセス管理が併せて必要となる。これは暗号鍵の管理が脆弱な環境（例えば復号のための鍵が誰からでもアクセスできる）においては、暗号化したデータは簡単に復号されてしまうからである。

従って、本ガイドラインにおける暗号化対策は、「暗号化」・「暗号鍵の管理」・「通信経路の暗号化」の対策度合いに応じて、各脅威に対する対応レベルを設けることとする。以下の内容については大項目にてそれぞれ詳細について言及する。

１．DB 暗号化

一般的に DB の暗号化とはデータベースに格納する、格納データに対する暗号化を指すが、ここでは DB が動作しているサーバ HDD に対する暗号化についても考慮する。以下がこのガイドラインで定める DB 暗号化の種別である。

- **HDD 暗号化**：HDD 暗号化は、サーバ起動時には内部データが復号されるが、適切なユーザログインのない状態では暗号化された状態で保存される。このソリューションは、物理的な DB サーバのディスクの盗難において効果的である。
- **DB テーブルの暗号化**：DB 自体の暗号化については前段のアプリケーションで暗号化を実装する場合と DB サーバにて実装する 2 つのケースが存在するが、アプリケーションでの暗号化の場合、該当テーブルに格納されたデータを参照する際にすべてのデータが復号され、メモリ上にキャッシュされるため、メモリダンプ解析などに対するリスクがある。その反面、以後の同様の領

域に対するアクセスはメモリ上の復号データに直接アクセスできるため、パフォーマンスの劣化はない。

- **DB カラムの暗号化**：カラム単位の暗号化についてでも上記と同様にアプリケーション側での実装と DB 側での実装が挙げられ、アプリケーションでの実装はメモリダンプ解析などのリスクが存在する。しかしテーブル単位の暗号化に比べ、カラム単位で暗号鍵を使い分け、セキュリティ強度をコントロールできる面で優れている。
- **バックアップデータの暗号化**：DB バックアップデータの保存については、バックアップ前のデータがすでに暗号処理されている場合については保護の考慮の必要がない。ただしバックアップデータ復元の際に鍵の世代管理が考慮されていないと、データが復号できなくなる。

2．暗号鍵管理

- **ファイルとしての暗号鍵管理**：DB 内ないし外部サーバでの鍵の管理については、特権ユーザによる鍵のアクセスが容易であり、また鍵のバックアップを行った際、そのバックアップメディアの管理についても検討が必要となる。また DB 内での鍵管理はデータと暗号鍵が同じ HDD 内に存在するため、HDD 盗難時の漏えいリスクが存在する。暗号化処理についても DB にて行うため、導入時はパフォーマンスへの影響を十分検討する必要がある。また、鍵の更新などにおける世代管理についても十分検討されているか、確認すべきである。
- **専用 H/W での暗号鍵管理**：ハードウェアセキュリティモジュール (以下 HSM) は暗号鍵を管理する専用 H/W であり、鍵に対する耐タンパー性を持ち、暗号化のオフロードや鍵の世代管理、管理者の職務分掌機能などによる暗号化データ・アクセスに対する管理が容易な H/W である。導入時には効果を最大限発揮できるよう、しっかりとした内部統制に沿った設定および運用が必要である。

  *HSM とは：FIPS140-2 にて定義されるような耐タンパー性を備えた暗号化管理 H/W。暗号化の実行および暗号鍵のライフサイクル管理を行う。外部からの鍵盗難や改ざんを防御し、鍵管理者に対する職務分掌の徹底を実行させる機能を持つ。近年では DB の暗号鍵管理および暗号化処理を行うものとして利用もされる。

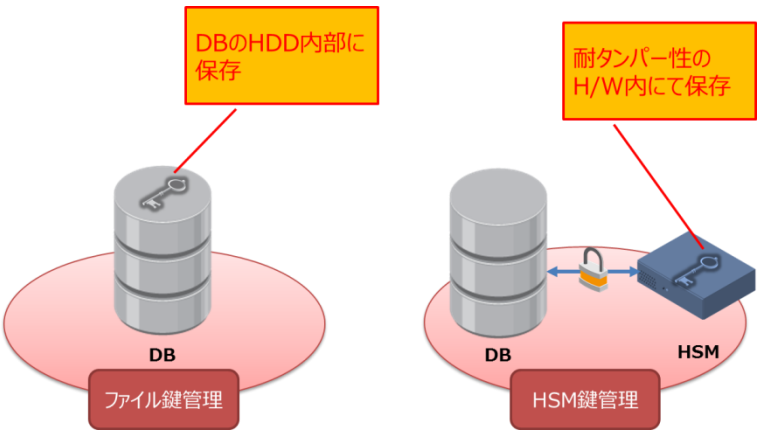


図3．鍵管理イメージ図

- **暗号鍵のアクセス制御**：暗号化実装において最も大事な項目は暗号鍵の管理であるが、それと同様に気をつけなければならないのが、適切な暗号鍵アクセス制御である。しかるべきユーザだけがしかるべき暗号鍵にアクセスできるよう制御が必要であり、場合によっては使用できる時間やその回数を制限することも考慮が必要である。
- **鍵の世代管理**：同じ暗号鍵を使用し続けることはセキュリティ上望ましくない（いずれ解読される危険性があるため）、定期的な鍵の更新は適切な期間を以って実行される必要がある。これについては、バックアップデータ復元の際の操作も考慮した、鍵の世代管理が必要となる。また、鍵管理の操作が特定の管理者のみで実行される場合、その管理者の ID 漏えいや不正利用が情報漏えいとなる可能性があるため、鍵の分散管理に代表されるような、鍵管理における管理者の職務分掌も併せて考慮が必要である。

3．通信経路の暗号化：

- **DB-アプリケーション間**：アプリケーションから DB に対しては SSL に代表されるような暗号化によって、パケット盗聴などによる情報漏えいを防ぐことが可能となる。
- **DB-管理者端末間**：一方、管理者端末からのアクセスについては、一般的には管理マネージャーソフト自体に通信暗号化機能がついているものがほとんどだが、事前に確認が必要である。またコマンドラインによるアクセスを行う際は、SSH に代表されるような暗号化アクセスを行うなどの徹底が必要となる。
- **DB-鍵管理デバイス間**：DB からしかるべきデータの暗号化や復号処理を行うための鍵のロードや暗号化関連通信について、一般的には SSL に代表されるような暗号化によって、パケット盗

聴などによる情報漏えいを防ぐことが可能となる。HSM の場合、独自の暗号化通信機能がついているため、それを利用することが望ましい。

2.3 脅威の定義

DBセキュリティにおける脅威のうち、暗号化を適用することによって回避できる脅威と、暗号化を適用した上でも考慮の必要がある脅威について以下の通り示す。

１．脅威一覧

- 暗号化による回避可能な脅威
  - ① DBのHDD抜き取りによる情報漏えい
  - ② バックアップメディア盗難による情報漏えい
  - ③ 通信経路のパケット盗聴による情報漏えい
- 暗号化後も考慮の必要な脅威
  - ① メモリダンプ解析による情報漏えい
  - ② 暗号鍵の盗難による情報漏えい
  - ③ 特権ユーザによる暗号鍵悪用による情報漏えい
- 暗号化では対応できない脅威
  - ① 正規でないユーザアクセス(成りすまし等)による情報漏えいは、DBSCガイドライン(項：2.4.7 脅威の一覧) を参照のこと。(全体的な脅威については本ガイドラインでは対象外とする。)

２．脅威と対策のマッピング

各脅威とその各対策のマッピングを以下のとおり占めす。

対応箇所	HDD 抜き取り	バックアップメディア盗難	メモリダンプ解析	暗号鍵の盗難	パケットの盗聴	特権ユーザによる暗号鍵の悪用
DB暗号化（HDD暗号化）	●	x	x	△	x	x
DB暗号化（DBテーブルの暗号化）	●	●	●	△	x	x
DB暗号化（DBカラムの暗号化）	●	●	●	△	x	x
アプリでのDB暗号化（DBカラムの暗号化）	●	●	x	△	x	x
暗号鍵管理（ソフトウェア）	-	-	●	△	-	△
暗号鍵管理（HSM）	-	-	●	●	●	●
通信経路の暗号化	-	-	-	●	●	-
暗号鍵アクセス制御	-	-	-	-	-	●
暗号鍵の分散管理	-	-	-	-	-	●

- ●-対応可能、 △-製品によっては対応可能、 x -対応不可能、 - -範疇外

3　DB の暗号化

ここでは DB 暗号化の実装について、先ほど整理した内容に沿ってその詳細を以降に記述する。

DB 暗号化は、「HDD 暗号化」、「DB テーブル暗号化」、「DB カラム暗号化」および「バックアップテープ暗号化」に分類される。近年ではトークナイゼーション (Tokenization) 技術についても新たなアプローチとして登場しているが、この場合も対象となるデータについては対象カラムごとに暗号化されたデータがトークンと紐付けられて管理されるため、機密データの取り扱いについては上記の「DB カラム暗号化」に分類される。なお本ガイドラインではトークナイゼーションについては言及しない。

3.1 HDD 暗号化

3.1.1　HDD 暗号化概要

HDD 暗号化は一般的に PBA (プリブート認証) をサポートしたものが多く、OS が起動する以前に該当*製品*がユーザ認証を求める仕組みにより、適切なユーザのみ HDD の復号および OS の起動を実行できる。このソリューションの用途としては HDD の不正な持ち出しに対し、第 3 者に不正にデータを盗聴されないようにするものであるが、万が一第 3 者がその PBA 認証情報を入手していた場合、効力がなくなってしまうため、ユーザ認証情報の管理については当然ながら最善の管理が求められる。

データの暗号･ 復号自体はその OS が動作する H/W に一存するため、十分なリソース (CPU、メモリ) が確保されていることが必要である。

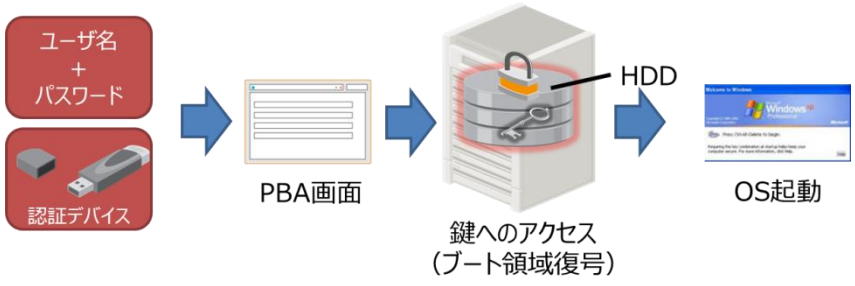


図4．HDD暗号化動作概要

3.1.2　HDD 暗号化のメリット

以下に HDD 暗号化実装のメリットについて言及する。

- 暗号化に伴う DB ファイルサイズについては変化しないため、既存テーブル設計を変更することなく暗号化が実施できる。
- PBA 認証にあたっては、ユーザ名パスワードによる認証だけでなく、認証トークンやスマートカードなどを用いた二要素認証をサポートしている製品が多く、ユーザのニーズに応じたセキュリティ強度を選択可能。
- 既存 DB サーバにインストールないし該当の機能を持つ HDD を利用するだけで導入が可能（アプリケーション等の変更の必要なし）。
- HDD の物理的な盗難に対しては、上記の PBA 認証情報の安全が確保されている限り効力がある。
- 検索やデータタイプに依存しないため、従来どおりの DB 利用が可能となる。

3.1.3　HDD 暗号化デメリット

以下に HDD 暗号化実装のデメリットについて言及する。

- 一度 PBA を突破し、OS が起動すると、HDD 暗号化はデータ･ アクセスがあった箇所を適宜復号するため、不正アクセスや内部 DBA の不必要なデータ参照を防ぐことはできない。
- 上記のとおり復号されたデータは通常メモリ上にあるため、メモリダンプ解析などの攻撃に対して脆弱である。
- データのバックアップを取る際も、データ･ アクセスと同時に復号されたデータがバックアップ取られるため、バックアップ自体は平文のデータが保存されることになる。
- HDD の暗号鍵自体がファイルとして存在している場合、その鍵ファイルの盗難に対しては厳重な注意が必要である。ただし製品の中には鍵を HDD 内に持たせず、正しいユーザの認証情報を元に鍵を生成するものがあるため、この点についてはよく考慮された製品を選択すべきである。

3.1.4　HDD 暗号化導入にあたって

HDD 暗号化の実装にあたっては、当然ながら上記に述べたメリット･ デメリットを把握した上で検討すべきである。HDD 暗号化は導入が容易な対策であるが、いったんシステム起動してしまうと、OS 上で許容するすべてのユーザアクセスを暗号化データに対して許容してしまう。したがって情報漏えい対策として

効力を発揮するためには、内部統制がしかるべき形で施行され、不正アクセスや内部 DBA に対する信頼がある前提となる。

なお一般的に導入にあたって本ガイドラインが定める導入におけるポイントは以下の通り。

- 導入前に HDD のデフラグおよびチェックディスクを実行する必要の有無を確認すること。（必須）
- 上記によるシステムへの負荷もしくはシステム停止の影響を事前に考慮すること。（必須）
- DBMS の OS が該当 HDD 暗号化製品にてサポートされているかどうか確認すること。（必須）
- インストール前にデータのバックアップを取得すること。（必須）
- PBA の権限を適切な管理者に与え、パスワードを強固なレベルにて実装すること。（必須）
- 事前に暗号化にかかる時間を検証すること。（必須）
- PBA に二要素認証をかけること。（推奨）
- 暗号鍵のバックアップ管理について適切な管理者が保護されたメディアないし物理的な場所にて保管すること。（推奨）

3.2 DB テーブルの暗号化

3.2.1 DB テーブル暗号化概要

DB には、表や索引、ビューといった論理的なオブジェクトが存在しており、そのオブジェクトを暗号化することでオブジェクトに紐づく物理的なファイルが暗号化される。これにより、DB のファイルに関連するハードディスクの盗難やファイルコピーによる持ち出しなどでデータが漏えいしたとしてもその情報の機密性を担保できる。

本ガイドラインでは、それらオブジェクトを総称して「DB テーブル」と呼ぶこととする。DB テーブルの暗号化については製品によって実装の差異はあるものの、テーブル全体を暗号化するという意味においてはどの製品についても同様にテーブルに格納されたデータが暗号化されるため、該当テーブルにアクセス権限を持つものであれば復号してデータを参照できる。

日本においては個人情報保護の観点から保護対象となるカラムは多く、パフォーマンスの観点から表単位での暗号化を実装する例が多い。

*DB テーブル：後述にあるカラム暗号化と比較する上で、便宜上表領域の暗号化を含めた DB テーブル（領域）全体の暗号化対象をこのように表現することとする。

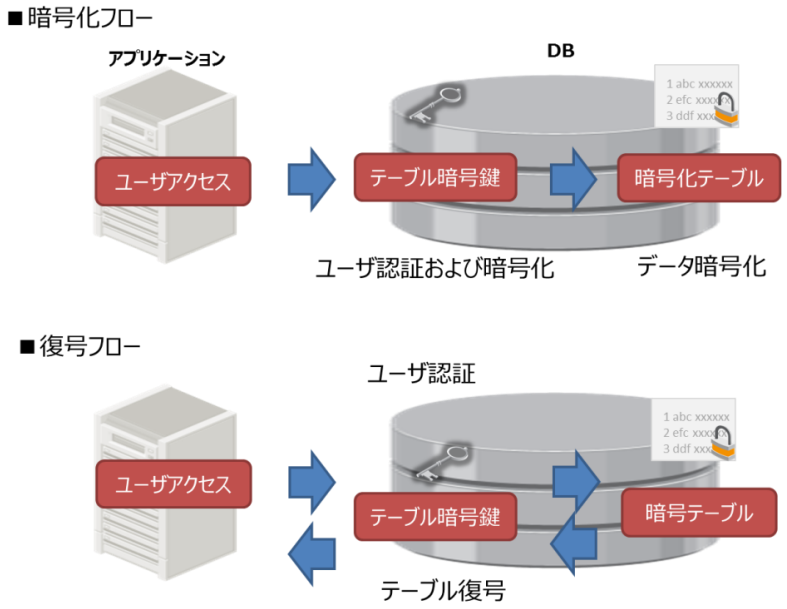


図5．DBテーブル暗号化動作概要

3.2.2　DB テーブル暗号化のメリット

以下に DB テーブル暗号化のメリットについて、アプリケーションおよび DB 実装の双方について言及する。

- DB テーブル暗号化は、アプリケーションに修正を必要としないのが一般的である（透過的暗号化）。よって既存の SQL をそのまま流用できる。
- 該当するテーブルのデータをすべて暗号化できる。
- 暗号管理は DB 側で統一できるため、管理が容易である。
- 暗号化テーブル情報が漏えいしたとしても、暗号鍵が漏えいリスクに晒されない限り、データの担保が可能となる。

3.2.3　DB テーブル暗号化のデメリット

以下に DB テーブル暗号化のデメリットについて、アプリケーションおよび DB 実装の双方について言及する。

- テーブル内すべてのデータを同一の鍵で暗号化するため、暗号化データと同時に暗号鍵情報が盗難された場合、すべてのデータを復号することが可能なため、情報が漏えいしてしまうリスクがある。
- DB にて暗号化処理が行われるため、既存のリソースをより多く消費することとなる。

3.2.4　DB テーブル暗号化導入にあたって

以下に DB テーブル暗号化を実装する上で考慮が必要な項目について纏める。

- DB に暗号化を動作させるにあたり十分なリソース（CPU、メモリ等）が確保されているか確認すること。（必須）
- 運用時のパフォーマンスについて事前に十分テストを行うこと。（必須）
- すべてのテーブルが暗号化完了するまでの時間を事前に確認することで、サービス断に必要な時間を割り出すこと。（必須）
- 暗号鍵へのアクセス制御を正当に設定すること。（必須）
- テーブルごとに暗号鍵を分けること。（推奨）

3.3　DB カラムの暗号化

3.3.1　DB カラム暗号化概要

カラム単位での暗号化は、PCI DSS のように保護対象が限定しているケースで多く採用が見られる。機密性の高いデータカラムのみを暗号化し、またカラムごとにユーザアクセス制御をかけられることからセキュリティレベルは DB テーブル単位の暗号化よりも高い。

カラム暗号化は一般的にアプリケーションで実装する方法と、DB で実装する方法があるが、それぞれ一長一短があるため、利用する場面に応じて適切な導入形態を選択する必要がある。

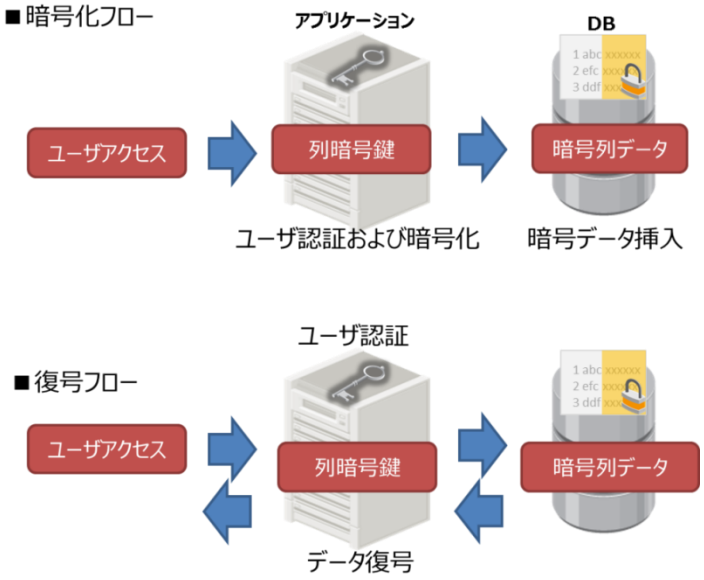


図6．アプリケーション統合でのカラム暗号化動作概要

一般的にカラム暗号化の実装で懸念されるものとして、「フルテーブルスキャン」による処理遅延がある。これは、暗号化に設定されているカラムに対して検索を走らせた結果、該当カラムのデータすべてを復号し、そこから検索を実行する処理が走るため発生するものである。ただし最近の製品では、自動的に検索キーを暗号化し、暗号化された検索キーにより暗号データの検索を行うものもあるため、事前に確認しておくとよいだろう。

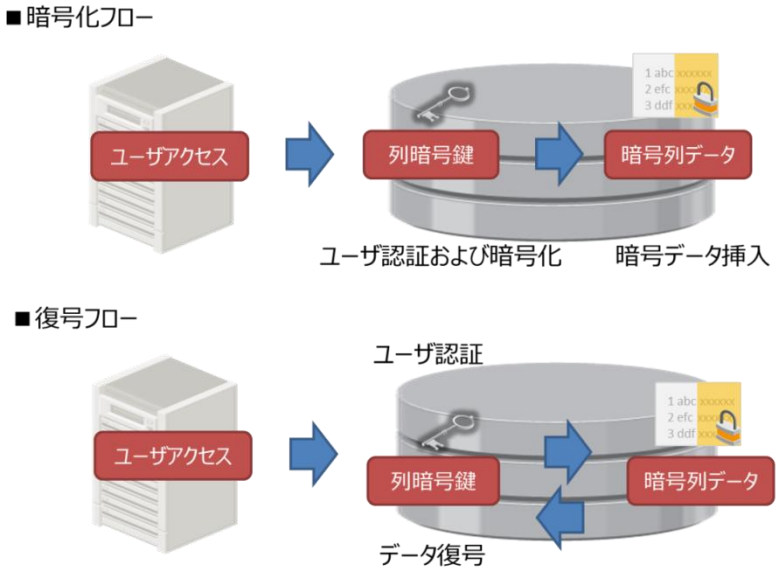


図7．DB統合でのカラム暗号化動作概要

3.3.2　DB カラム暗号化のメリット

1. **アプリケーションでの対応**

- データフロー上、DB の前段のサーバで暗号化が実装されるため、よりセキュアな実装が可能となる。
- アプリケーションで処理をするため、DB に負荷をかけず導入ができる。
- 暗号鍵をデータと物理的に分けて管理できる。

2. **DB での対応**

- 一般的にアプリケーション側での SQL 修正が不要。
- 製品によってはカラム単位でセキュリティレベル(鍵アルゴリズムの選定、鍵アクセスユーザの定義) が設定可能であるため、柔軟な運用が可能となる。
- 暗号化対象が狭いため、鍵交換などの運用における影響が小さい。

3.3.3　DB カラム暗号化のデメリット

1. **アプリケーションでの対応**

- アプリケーションに暗号化機能を組み込む必要があるため、開発工数を見込む必要がある。
- アプリケーションサーバで鍵を持つことになるため、十分な鍵管理を考慮する必要がある。
- 一般的にアプリケーションの負荷を上げる可能性がある。

2. **DB での対応**

- 暗号化データのデータ長変化に伴うサイジング増大。
- フルテーブルスキャンなどによるパフォーマンス低下の可能性がある。
- 暗号化データと同時に暗号鍵情報が盗難された場合、データを復号、漏えいしてしまうリスクがある。
- カラムごとのセキュリティレベルの設定および管理が煩雑になる。

3.3.4　DB カラム暗号化導入にあたって

以下に DB カラム暗号化を実装する上で考慮が必要な項目について纏める。

- 暗号化対象カラムとして機密データに該当する、もしくは個人を特定できないレベルのカラムを対象と選定すること。（必須）
- 暗号化に伴う DB 肥大化に対応できる HDD 空き容量を事前に確認すること。（必須）
- 暗号化したいデータタイプが暗号化製品にて対応しているか確認が必要。（必須）
- すべてのデータが暗号化完了するまでの時間を事前に確認することで、サービス断に必要な時間を割り出すこと。（必須）
- 暗号カラムを検索キーとするクエリがないか事前に確認すること。該当する場合、フルテーブルスキャン等、パフォーマンスに悪影響を起こす可能性があるため、設計の見直しが必要となる。（製品によって対応可能な場合を除く）（必須）
- 適切なユーザもしくはユーザグループに対し適切な鍵アクセスを与えること。詳細については　「4.3 項　暗号鍵のアクセス制御」を参照のこと。（必須）
- 最低限テーブル単位で暗号鍵を分けて暗号化すること。（必須）

3.4　バックアップデータの暗号化

3.4.1　バックアップデータ暗号化概要

近年ストレージを用いた DB 運用などが見られるものの、未だバックアップテープでの管理は主流である。バックアップテープに対する暗号化を提供する製品が市場には出回っているものの、前項までで紹介しているソリューションにて DB 暗号化が施されていれば、基本的にバックアップデータに対する個別暗号化は必要ない。

よって本ガイドラインでは、DB 自体が何がしかの方法で暗号化されており、その暗号化データをバックアップしたメディアに対する運用についてのみ言及する。

3.3.2　バックアップデータ暗号化導入にあたって

以下にバックアップデータ暗号化管理において考慮が必要となる点について纏める。

- 暗号鍵の鍵更新を行っている場合、どのバックアップデータがどの世代の鍵で暗号化されているか管理する仕組みが必要となる。（必須）
- 暗号鍵を保存したメディアとバックアップデータを同様の場所で保管しないこと。（必須）
- バックアップデータの管理場所については複数の管理者にて管理を把握できること。（推奨）

4　暗号鍵管理

DB 内ないし外部サーバで暗号鍵を管理する方式について説明する。特権ユーザによる鍵のアクセスが容易であり、また鍵のバックアップを行った際、そのバックアップメディアの管理についても検討が必要となる。ただし、鍵の更新などにおける世代管理についても十分検討されているか、確認すべきである。

外部での鍵管理を行う代表的なデバイスとして、HSM がある。HSM は物理的なデバイスであり、オペレーティング・システム・ファイルではないため、鍵は許可されていないアクセスの試みから保護される。マスター暗号化鍵を使用する暗号化および復号のすべての操作は、HSM の内部で実行される。これは、安全性の低いメモリ内でもマスター暗号化鍵が危険にさらされないことを意味する。

マスター暗号化鍵は、DB の外部にあり、セキュリティ管理者のみがアクセスできる外部セキュリティ・モジュールに格納する。通常のプログラム機能と暗号化操作が分離されるため、DB 管理者とセキュリティ管理者の任務を分離できる。DB 管理者にはパスワードを知らせずに、セキュリティ管理者にパスワードの提供を要求できるため、セキュリティが強化される。　(HSM による暗号鍵管理詳細については 4.2 項にて説明する。)

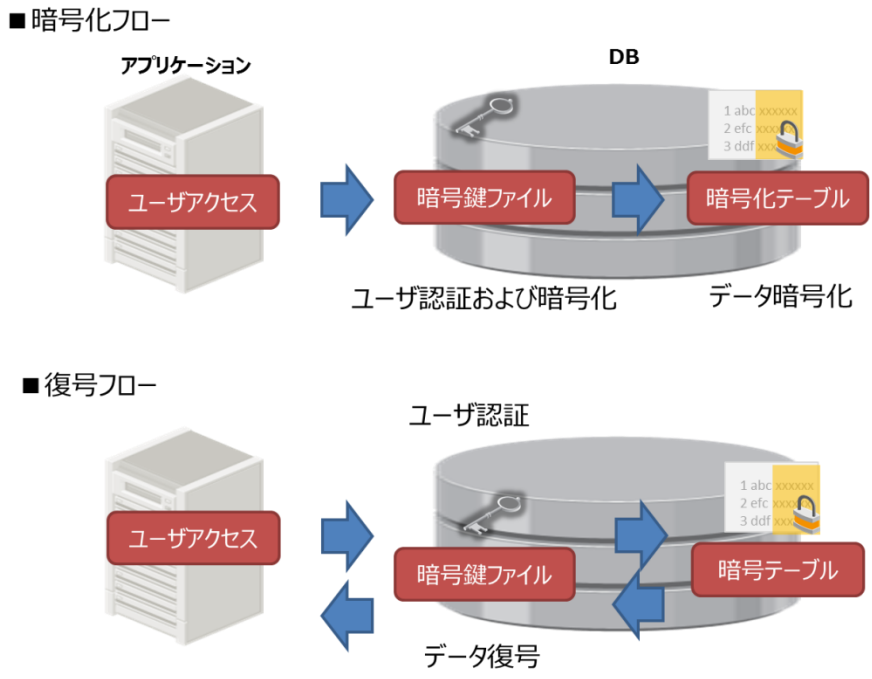


図8．ファイルによる鍵管理動作概要

4.1 ファイルとしての暗号鍵管理

4.1.1　ファイルとしての暗号鍵管理概要

暗号鍵の更新による世代管理が必要である。　暗号化データには、マスター暗号化鍵なしでアクセスすることはできない。マスター暗号化鍵は DB サーバ自体に格納されるため、安全な場所に定期的にバックアップする必要がある。新しいマスター暗号化鍵を設定するたびにバックアップ･ コピーを作成する必要がある。

4.1.2　ファイルとしての暗号鍵管理メリット

以下にファイルとしての暗号鍵管理のメリットについて言及する。

- 鍵の管理が容易 / 簡易である。

4.1.3　ファイルとしての暗号鍵管理デメリット

以下にファイルとしての暗号鍵管理のデメリットについて言及する。

- DB サーバ内での鍵管理はデータと暗号鍵が同じ HDD 内に存在するため、HDD 盗難時の漏えいリスクが存在する。
- パスワードによる鍵の保護が主となるため、管理者のパスワード管理が重要となる。
- 暗号化処理を DB にて行うため、導入時はパフォーマンスへの影響を十分検討する必要がある。

4.1.4　ファイルでの暗号鍵管理導入にあたって

ファイルでの暗号鍵を管理する場合、念頭に置かなくてはならないのはその機密性の確保である。DB 内で利用される鍵が改ざんや盗難、不必要なアクセスから守られなければならないため、これに対する対策がどれだけ取られるかが暗号データに対するセキュリティレベルの確保につながる。

本ガイドラインが定める導入におけるポイントは以下の通り。

- ソフトウェア側で提供される鍵の保護オプション（パスワード保護等）は必ず利用する。（必須）
- 上記オプションで保護される場合、パスワード盗難が情報漏えいに直結するため、パスワードを媒体に記載してはならない。（必須）
- 保護されたフォーマットの暗号鍵については、外部メディアに安全にバックアップを取得し、保管する。（必須）
- 暗号鍵の管理権限については、2人以上の管理者で職務分掌が徹底できることが望ましい。（推奨）

4.2 専用 H/W（HSM）での暗号鍵管理

4.2.1 HSM での暗号鍵管理概要

HSM は暗号鍵の格納および暗号処理を行う専用の H/W モジュールであり、主にサーバへ内蔵するボード型か、ネットワーク接続型のアプライアンス機器として提供される。

HSM 製品には次のような基本機能が求められる。

- 乱数生成器を用いた推測不可能な暗号鍵の生成
- 物理的な暗号鍵データの読み取り防止 (耐タンパー性)
- 暗号演算を H/W で高速に処理

HSM 製品の機能要件とセキュリティレベルは、米国連邦情報処理標準 FIPS140-2 を参照するのが一般的である。FIPS140-2 では暗号モジュールのセキュリティレベルを 4 段階にレベル分けしており、通常は対象システムのセキュリティ要件に応じて、同標準のレベル 2 もしくはレベル 3 の認定を受けた製品が利用される。

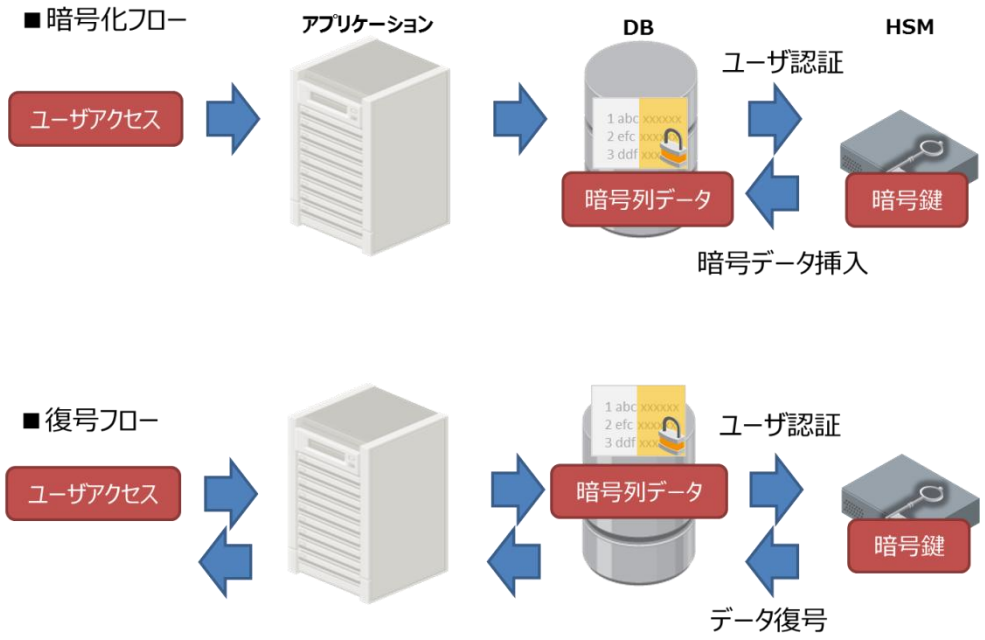


図9．HSMによる鍵管理動作概要

ホストシステムと HSM の通信は、PKCS#11 や Microsoft CAPI/CNG などの API 経由で行われる。いくつかの DB 製品ではネイティブの暗号化機能でこれらの API をサポートしており、暗号鍵の管理に HSM を利用する事が可能となっている。

＊FIPS140-2：主に米国連邦政府各省庁で利用される、「暗号モジュール」に対する要件で、ハードウェアおよびソフトウェアに対するセキュリティ品質を極め細やかに定義しており、4 つのレベル分けがされている。レベル 2 以降では製品に対する物理的改ざんに対する対策や、役割/ID ベースのオペレータ認証などが含まれている。（参照 URL: http://csrc.nist.gov/groups/STM/cmvp/standards.html ）

4.2.2 HSM での暗号鍵管理メリット

以下に HSM による暗号鍵のメリットについて言及する。

- 暗号鍵の生成から廃棄までのライフサイクルが HSM 内部で一元的に管理されるため、暗号鍵が HSM 外部に漏えいしない事が保証される。
- 複数の DB サーバで暗号化を実施する際の暗号鍵管理を HSM に集約する事によって、暗号鍵管理の運用を容易にする。

- DB サーバから独立した独自のアクセス制御機構を持っているため、DB 管理と暗号鍵管理を分離する事が可能となる。
- 暗号処理を HSM へオフロードする事によって、サーバの CPU 負荷を軽減する。

4.2.3 HSM での暗号鍵管理デメリット

以下に HSM による暗号鍵のデメリットについて言及する。

- 導入時の機器調達、運用の見直し等に伴う初期費用の発生。
- API 組み込みのための工数。
- 管理が必要なデバイスの増加。
- 効果的な管理のためには複数の管理者による運用が必須。

4.2.4 HSM での暗号鍵管理導入にあたって

HSM は暗号鍵管理機能を自動化するものであるが、HSM 管理者の権限を正しく分掌しない限り、その特定の管理者アカウントがセキュリティ・ホールとなってしまう。一般的な HSM ではそのような職務分掌のための機能が実装されているため、これを最大限に生かすことが理想的である。

本ガイドラインが定める導入におけるポイントは以下の通り。

- DB-HSM 間の通信は SSL などの通信手段で保護をする。詳細については後述の「5.3 DB-鍵管理デバイス間」を参照のこと。（必須）
- 必要なパフォーマンスが HSM で保障されることを事前に検証し、確認すること。（必須）
- HSM の管理者については２人以上をアサインし、鍵の生成や削除、世代交代等の機能について職務分掌を徹底すること。（推奨）
- HSM を冗長構成で導入することで、H/W に問題があった場合でも処理が継続できるような運用をすること。（推奨）

4.3 暗号鍵のアクセス制御

4.3.1　暗号鍵のアクセス制御概要

暗号化の正当性を保障するためには、暗号鍵の物理的な保管に対する制御だけではなく、暗号鍵にアクセスできるユーザの制御も合わせて重要である。暗号鍵のアクセス制御を適切に設定することができなければ、暗号化の効力を十分に発揮することはできない。暗号鍵へのアクセスを制御するということは、その鍵で暗号化されているデータに対するアクセス制御と同義である。

一般的には、ユーザアプリケーションからのアクセスについてはサービスに影響のない形でのアクセス制御が必要だが、DBA からの通常アクセスの部分については、運用上必要な部分を除いて最小限のアクセスに止めることが重要である。

アクセス制御の効果範囲については、暗号鍵がその暗号対象としている範囲によって変化する。

- HDD 暗号化：ディスク全体が範囲、PBA 突破後はすべてのユーザがすべてのデータにアクセス可能
- DB テーブル暗号化：指定されたテーブル（オブジェクト）が範囲、認証後は該当テーブルすべてのデータにアクセス可能
- DB カラム暗号化：指定されたカラムが範囲、認証後は該当するカラムのデータにのみアクセス可能

またアクセス制御については以下のような項目があり、これらをソフトウェアの機能およびセキュリティポリシーを適切に定義することで適切なアクセス制御および暗号化が運用できる。

- ユーザまたはユーザグループ毎へのアクセス制御：適切なユーザ（グループ）のみが対象となるデータにアクセスできることを保証
- 鍵の利用できる時間帯の制限：アクセス権限を持つユーザが常にデータ・アクセスするのではなく、必要最小限のアクセスが必要な時間帯に鍵アクセスを与える（24 時間 365 日の運用が必要な場合を除く）ことで、想定外の時間からのアクセスによる情報漏えいを防ぐ
- 一度に参照できるデータ量の制限：上記 2 つの制限にかからないアクセスであっても、不正に情報を抜き取るようなアクセス（大量のデータの復号）を防ぐために、一度に参照可能なデータ量を制限することで情報漏えいを事前に防ぐ

アクセス制御についてはソフトウェア的な機能で対応できる部分と、セキュリティポリシーに沿った人的な運用が必要な部分があり、製品によって強制力が変わる。

またすべてのアクセス制御設定、運用については、その正当性を保証するためにポリシー管理者の運用ログを取得し、過去の変更を把握できる体制作りも考慮したい。

4.3.2　暗号鍵のアクセス制御メリット

以下に暗号鍵のアクセス制御のメリットについて言及する。

ただし内容は鍵アクセスが適切に設定されている場合のメリットとする。

- 権限を持つユーザからの悪意ある情報漏えいを最低限に防ぐことが可能となる
- アクセスログの量を最適化し、解析を容易にすることが可能となる

4.3.3　暗号鍵のアクセス制御デメリット

以下に暗号鍵のアクセス制御のデメリットについて言及する。

ただし内容は鍵アクセスが適切に設定されている場合のデメリットとする。

- カラム単位の暗号化の場合、単一の鍵で運用する場合に比べ、鍵のユーザマッピングが煩雑になりやすい
- 管理者の設定ミスが大きな情報漏えいに繋がる可能性がある
- ポリシー外のアクセスを許容する場合は新たにアクセスポリシーを追加するなど運用が負荷となる場合がある
- セキュリティポリシーに従った人的な運用が必要な場合、その正当な運用を確認するためのアクセスログ運用が必要不可欠となる

4.3.4　暗号鍵のアクセス制御導入にあたって

暗号鍵の適切なアクセス制御の実装にはいくつかの手法がある。ひとつは「ルール」として鍵に対するアクセス権限を設定する方法であり、一般ユーザおよび管理者がその制御対象であり、ソフトウェア的な制御により実現可能である。もうひとつは「権限」を持った管理者に対するアクセス制御である。「権限」を持っている以上、何らかの方法で不必要なアクセスがないことを保証しなければならない。

本ガイドラインが定める導入におけるポイントは以下の通り。

- カラムごと、ないしはテーブルごとに暗号鍵を定義し、それに対するアクセスはユーザないしユーザグループ単位で制限する。（必須）
- 管理者の鍵に対するアクセス（暗号データに対するアクセス）は、必要最低限であることを証明するため、管理者の鍵に対するアクセスを監査する仕組み（ログ解析）を実装する。（必須）
- 一般ユーザの鍵アクセスについても正しく運用されているか、不審または過剰なアクセスがないか定期的にアクセスログを監査する。（必須）
- 管理者に対する暗号鍵の作成・更新・削除などの権限については、2人以上の管理者で権限を分掌する仕組みを利用する。（推奨）
- 上記の職務分掌がソフトウェア機能により提供されない場合、セキュリティポリシーにより複数の管理者で運用が分掌されることを徹底すること。（推奨）
- 鍵のアクセス制御については、参照できる時間および短時間における参照回数を制限できる場合、その機能を利用すること。（推奨）

4.4 暗号鍵の世代管理

4.4.1　暗号鍵の世代管理概要

暗号化したデータに対するセキュリティを確実なものにするためには、前述の項にもあるように暗号鍵自体の盗難および不正利用防止が重要であるが、それと同時に暗号鍵解読からも守る必要がある。

長期間同じ鍵で暗号化を実行することは、悪意あるユーザが暗号化データから暗号鍵を解析するのに十分な時間を与えてしまうものである。一般的に解読可能かどうか、また解読にかかる時間は暗号アルゴリズムの強度と解読側の資源（スーパーコンピュータークラス）に依存するが、昨今ではクラウドの潤沢な設備を悪用してこのような解析を行う例も見られているため、なるべく強固なアルゴリズムの選定と、ある頻度での暗号鍵世代管理が必要である。

ここで言う暗号鍵の世代管理とは、「暗号鍵の鍵更新（Rekey）」であり、すでに暗号化されているデータに対する鍵更新（Key Rotation）を実行、何世代にも渡って鍵を更新することを想定し、各バックアップデータがどの世代の鍵によって暗号化されているか、鍵とデータのマッピングを管理することが必要となる。

4.4.2　暗号鍵の世代管理メリット

以下に暗号鍵の世代管理によるメリットについて言及する。

- 適切な期間での鍵更新がされている場合、暗号鍵解析による攻撃からのリスクを限りなくゼロに抑えることが可能。
- 暗号鍵が危殆化するリスクを最小限に抑えることが可能。

4.4.3　暗号鍵の世代管理デメリット

以下に暗号鍵の世代管理によるメリットについて言及する。

- 前項の「適切な期間」は、利用するアルゴリズムの強度にも依存するため、鍵の更新期間の定義が難しい。
- 暗号化されたバックアップデータと利用した世代の鍵のマッピングを、鍵更新する度に管理する仕組みが必要不可欠。

4.4.4　暗号鍵の世代管理導入にあたって

鍵の世代交代を定期的に行うことは、実装している暗号化が定常的にそのセキュリティレベルを保って運用していることを保障するものである。しかしその頻度については、暗号強度やシステムの規模、暗号鍵自体の管理手法などによって適切となる度合いが異なる。本ガイドラインではあえてその頻度については明言しないが、導入におけるポイントを以下の通りとする。

- 鍵の世代交代については、利用するアルゴリズムに対する脆弱性が発見された際には速やかに十分な強度の鍵にて再暗号化すること。（必須）
- 鍵の定期的な世代交代を行う場合、古い世代の鍵で暗号化されたバックアップデータとの紐付けを明確に記録すること。（必須）
- 古い世代の暗号鍵については、引き続き安全に保管すること。（必須）
- 古い世代の暗号鍵については、暗号化に利用されないよう暗号化の機能を無効にするか、そのような制御を徹底すること。（必須）

5　通信経路の暗号化

本章では、DB と外部システム (アプリケーションサーバ、鍵管理デバイス、管理端末) 間の通信経路を暗号化する方式について説明する。

DB と外部システム間の通信では、ある定められた通信プロトコルを用いて通信が行われる。例えば、SQL を用いたアプリケーションサーバと DB 間の通信プロトコルとしては、JDBC や ODBC が知られている。また、管理端末が DB と通信する場合、telnet を用いて DB (の OS) にログインすることになる (この場合、DB の OS 上から、DB に改めてログインすることが必要) 。

この DB と外部システム間の通信において、DB に格納される保護情報 (例 : カード番号、個人情報) が通信経路を盗聴されることにより漏えいする可能性がある。このため ,DB と外部システム間の通信では、盗聴されるリスクを考慮した上で必要に応じて通信経路を暗号化する必要がある。

以降では、DB とアプリケーションサーバ間、DB と管理者端末間、DB と鍵管理デバイス間の通信経路の暗号化について、通信経路の暗号化方法の概要、メリット、デメリット、通信経路の暗号化に際しての留意事項について述べる。

5.1 DB-アプリケーション間

5.1.1　DB-アプリケーション間の暗号化概要

DB とアプリケーションサーバ間の通信経路暗号化方式として、次の方式がある。

1) 標準化された暗号化方式を用いる方法 (例 : SSL)
2) DB のシステム仕様が規定する暗号化方式を用いる方法

これらの通信経路暗号化方式では、共通鍵暗号方式 (例 : AES、3DES) 、公開鍵暗号方式 (例 : RSA) などの暗号アルゴリズムが用いられる。

通信経路暗号化方式によっては、DB の変更、外部サーバの変更が求められる可能性がある。DB の変更としては、

1) DB への暗号化モジュールの組み込み
2) 通信ドライバの変更
3) DB 設定の変更

が挙げられる。また、外部サーバの変更としては、

1）外部サーバへの暗号化モジュールの組み込み

2）通信ドライバの変更

3）外部サーバ上のアプリケーションの修正

が挙げられる。

5.1.2 アプリケーション-DB 間の暗号化メリット

以下に DB とアプリケーションサーバ間の通信経路暗号化のメリットを述べる

- 通信経路が暗号化されているために、通信経路内の情報を解読することが困難となる。

5.1.3 アプリケーション-DB 間の暗号化デメリット

以下に、DB とアプリケーションサーバ間の通信経路暗号化のデメリットを述べる

- 通信経路を暗号化するため、処理性能が低下する可能性がある。（例：処理時間の増加、スループットの低下）

5.1.4 アプリケーション-DB 間の暗号化導入にあたって

以下に、DB とアプリケーションサーバ間の通信経路暗号化における留意事項を述べる．

- 使用する通信経路暗号化方式によるスループット低下など、システムへの影響を確認すること。（必須）
- 使用する通信経路暗号化方式に求められる設定を行うこと。（必須）
- 暗号アルゴリズムは、CRYPTOREC が推奨する暗号アルゴリズムを用いること。（推奨）

5.2 DB-管理者端末間

5.2.1　DB-管理者端末間の暗号化概要

まず、DB と管理者端末間の通信は、一般的に次の方式がある。

１）管理者端末が、DB 管理コンソールにログインし通信する方式（例：SSH）

２）管理者端末が、外部サーバと同じ通信方式を用いて DB と通信する方式

前者の方法では、管理者端末を用いる運用者は、DB が動作している OS 上で、DB 管理者権限を有するユーザのためのユーザ認証を行う必要がある。後者の方法は、項 5.1.1 と同じ通信方式である。以降では、前者の方式について述べる。

5.2.2　DB-管理者端末間の暗号化メリット

以下に、DB と管理者端末間の通信経路暗号化のメリットを述べる。

- 通信経路が暗号化されているために、通信経路内の情報を解読することが困難となる。

5.2.3　DB-管理者端末間の暗号化デメリット

以下に、DB と外部サーバ間の通信経路暗号化のデメリットを述べる。

- 通信経路を暗号化するための処理性能が低下する可能性があること。（例：処理時間の増加、スループットの低下）

5.2.4　DB-管理者端末間の暗号化導入にあたって

以下に、DB と外部サーバ間の通信経路暗号化における留意事項を述べる。

- 使用する通信経路暗号化方式によるスループット低下など、システムへの影響を確認すること。（必須）

- 使用する通信経路暗号化方式に求められる設定を行うこと。（必須）
- 暗号アルゴリズムは、CRYPTOREC が推奨する暗号アルゴリズムを用いること。（推奨）

5.3 DB-鍵管理デバイス間

5.3.1 DB-鍵管理デバイス間の暗号化概要

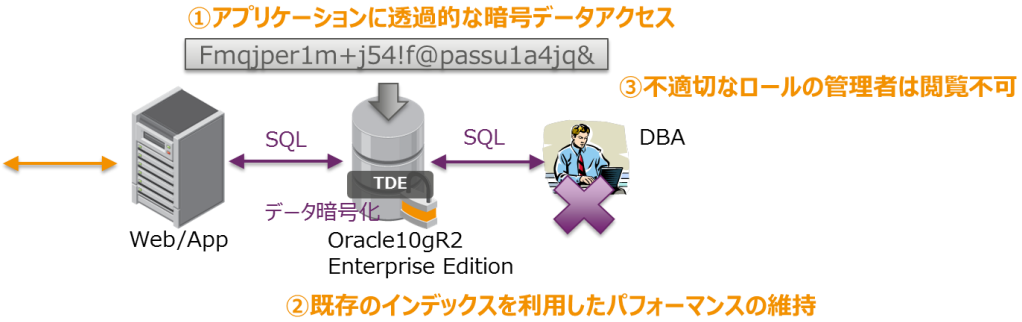
DB によっては、鍵管理デバイスである HSM との連携のためのインターフェースを用意しているものがある。下記はその一例であるが、いわゆる汎用的なインターフェース API を用いて連携することも可能なため、HSM 側が DB で対応している API を用いて作りこむケースも考えられる。

- Microsoft SQL Server : EKM(Extend Key Management)による HSM 連携。暗号化と鍵の生成については各 API を使用する。
- Oracle Database：PKCS#11 による暗号化と鍵生成連携。

なお HSM では暗号化通信による保護が通常機能として提供されており、ハードウェアによる暗号アクセラレーションにより、暗号化のデメリットである通信遅延を考慮する必要がほぼないため、メリット・デメリットの項について割愛する。

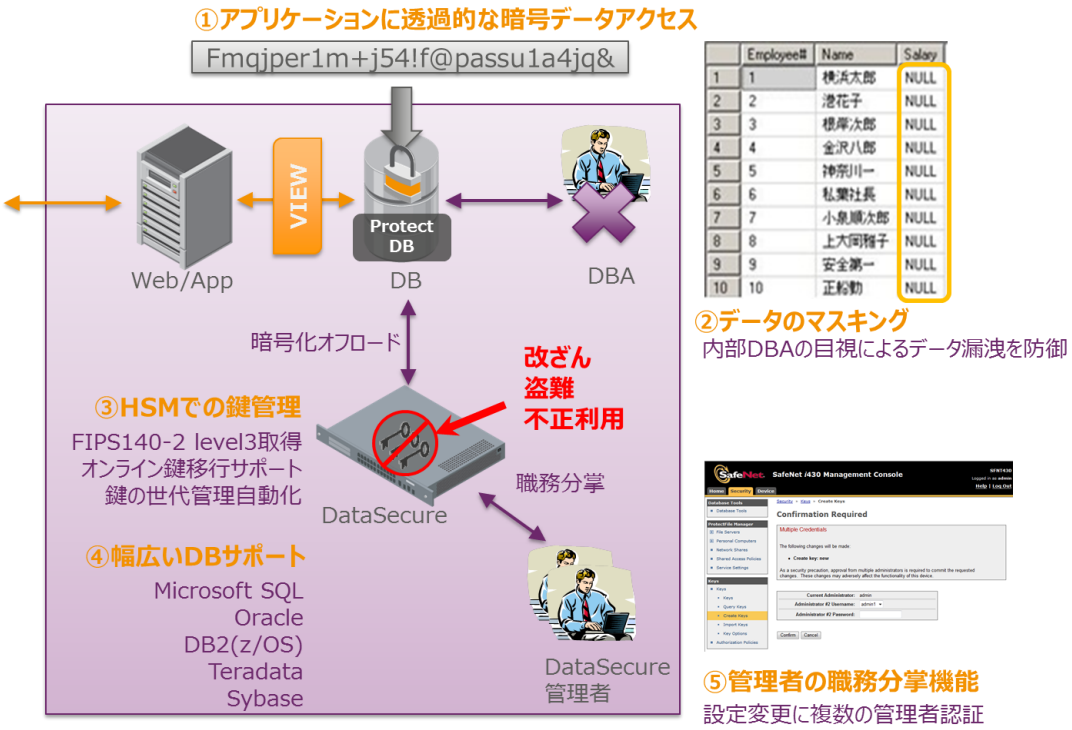
6　DB 暗号化導入事例

事例①：総務省制定ガイドラインへの対応



- ユーザ：　地方公共機関
- 対象 DB：　Oracle 10gR2
- 暗号化対象：　総務省のポリシーに沿った機密データの暗号化
- テーブルサイズ：　-万件
- データ移行時間：　-分
- 暗号化製品：　Oracle Advanced Security Option
- 暗号化方式：カラム単位
- 暗号鍵管理：Oracle Wallet による保護
- アクセス制御：ソフトウェア (ロールベースのアクセス管理)
- 通信経路暗号化：-
- 導入までの期間：　２ヶ月
- 導入費用概算：　-万円
- 導入決定の主なポイント：　既存アプリケーションを修正することなく暗号化の適用が可能、既存のインデックスを暗号カラムに適用することでパフォーマンスを維持、OS のアップグレードのみで暗号化対応可能なため、コストと時間を最小限に抑えることが可能。

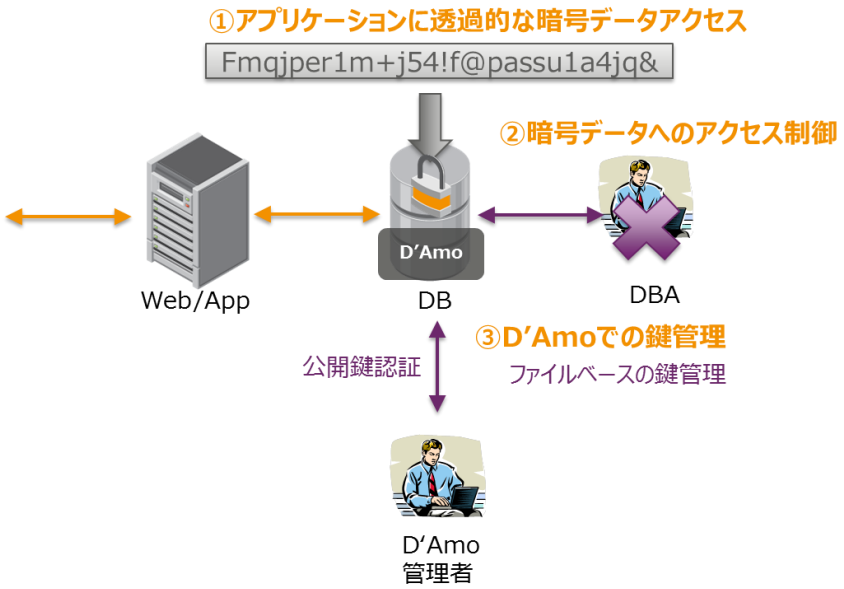
事例②：自社決済サービス用 DB の PCI-DSS 対応



- ユーザ： 製造業
- 対象 DB： Oracle 10gR2、Microsoft SQL2008
- 暗号化対象： クレジットカード番号、および一部個人情報を含む 5 カラム
- テーブルサイズ： 100～200 万件
- データ移行時間： 約 15 分
- 暗号化製品： SafeNet DataSecure i450
- 暗号化方式：カラム単位
- 暗号鍵管理：HSM（DataSecure）、鍵の世代管理自動化、オンラインでの鍵交換対応
- アクセス制御：ソフトウェア（鍵への時間単位、参照回数制限）
- 通信経路暗号化：SSL（DB-DataSecure 間、DataSecure-管理端末間）
- 導入までの期間： 4 ヶ月
- 導入費用概算： 1,200 万円
- 導入決定の主なポイント： 異なる DB の暗号化を単一の管理システムで提供、HSM による鍵管理の自動化、既存アプリケーションに修正を必要としない導入形態、DB に対するオンラインデータ移行・鍵更新の対応、PCI-DSS コンプライアンスに対する幅広い網羅性（要件 3、4、7、9、10）、内

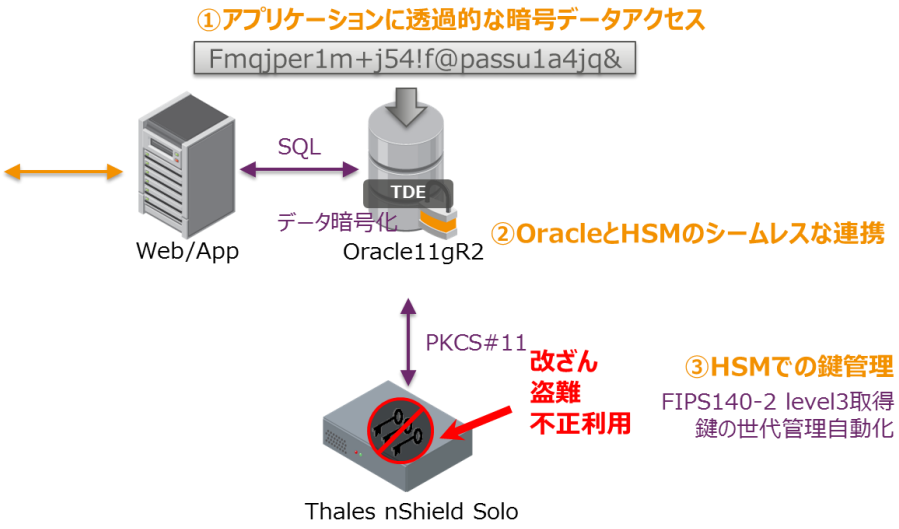
部 DBA に対するマスキングによるデータ保護、Active/Active クラスタ構成によるサービス継続性、サーバ単位での暗号化ライセンス体系。

事例③：社内コンプライアンス対応



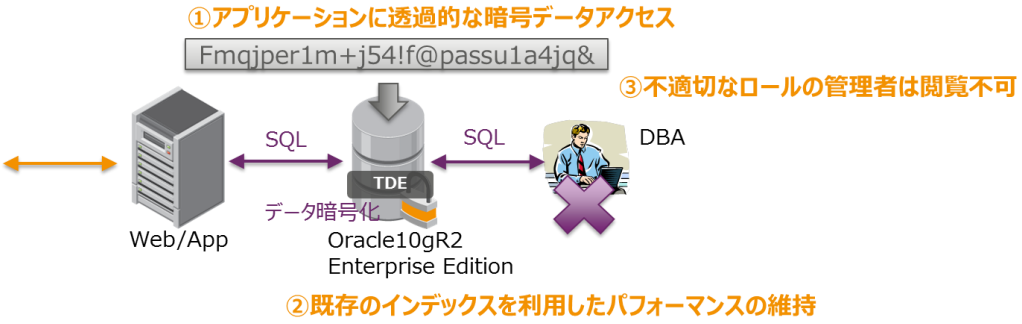
- ユーザ： 非公開
- 対象 DB： Microsoft SQL server
- 暗号化対象： 社員個人情報を含む 2 テーブル、各 2 カラム
- テーブルサイズ： 約 40 万件
- データ移行時間： 約 10 分
- 暗号化製品： D'Amo for SQL
- 暗号化方式：カラム単位
- 暗号鍵管理：ファイルでの暗号鍵管理 (D'Amo 内)
- アクセス制御：D'Amo によるアクセスコントロール設定 (暗号化・復号権限)
- 通信経路暗号化：なし
- 導入までの期間： 1 ヶ月程度
- 導入費用概算： 150 万円程度
- 導入決定の主なポイント： アプリケーションの再開発が不要で、暗号化・アクセスコントロールの設定が容易。

事例④：金融における PCI-DSS 対応



- ユーザ： 金融業
- 対象 DB： Oracle 11gR2
- 暗号化対象： クレジットカード情報
- テーブルサイズ： -万件
- データ移行時間： -分
- 暗号化製品： Thales nShield Solo, Oracle Advanced Security Option
- 暗号化方式：表領域暗号化
- 暗号鍵管理：HSM での暗号鍵管理 (Thales nSheild Solo)
- アクセス制御：-
- 通信経路暗号化：-
- 導入までの期間： ２ヶ月程度
- 導入費用概算： -万円程度
- 導入決定の主なポイント： FIPS140-2 Level3 に準拠した HSM による暗号鍵管理、HSM による PCI-DSS 要件 3 への準拠、アプリケーションに対して透過的に暗号化ができ、DBMS のセキュリティ機能と HSM のシームレスな連携が可能。

事例⑤：金融における PCI-DSS 対応



- ユーザ： 金融機関
- 対象 DB： Oracle 10gR2
- 暗号化対象： 顧客の個人情報
- テーブルサイズ： -万件
- データ移行時間： -分
- 暗号化製品： Oracle Advanced Security Option / Transparent Data Encryption
- 暗号化方式：表領域暗号化
- 暗号鍵管理：Oracle Wallet による保護
- アクセス制御：Oracle Database Advanced Security Option による暗号化
- 通信経路暗号化：Oracle Advanced Security Option / Network Encryption
- 導入までの期間： -
- 導入費用概算： Oracle Database ライセンス価格の 25%
- 導入決定の主なポイント： アプリケーションの再開発が不要で、暗号化の導入が容易。

7　DB 暗号化ガイドライン執筆者

DB 暗号化ガイドラインワーキンググループ（社名 50 音順）

主査：	日本セーフネット株式会社	髙岡　隆佳
監修：	デロイトトーマツ リスクサービス株式会社	丸山　満彦
	株式会社アシスト	小西　昭弘
	株式会社 NTT データ	花舘　蔵之
	タレスジャパン株式会社	上野 隆幸
	日本オラクル株式会社	西村 克也
		北野 晴人
	富士通株式会社	安澤　弘子
		今井　康雅
	株式会社ラック	大野　祐一
	株式会社ワールドスカイ	柳沢　智幸
		星野　樹昭