FUJITSU

shaping tomorrow with you

PostgreSQLにおける 透過的データ暗号化の実装

富士通株式会社
データマネジメント・ミドルウェア事業部
綱川　貴之

本書の内容

FUJITSU

- 情報漏えいを防ぐためのデータベース暗号化の必要性
- 透過的データ暗号化の特徴
- 透過的データ暗号化の使い方
- 透過的データ暗号化の実装
- 透過的データ暗号化を実装した製品の紹介

どんな情報を漏えいから保護したいか

FUJITSU

個人識別
氏名、住所、年齢、
電話番号、メールアドレス、
家族構成、勤務先

財産
クレジットカード番号
銀行の口座番号

医療記録
買い物履歴
友人/知人との会話

知的財産
自動車の構成部品
新薬の成分
出願前の特許

財務情報

多発する情報漏えい事件

FUJITSU

- Sony PlayStation Network、7700万件の個人情報が流出（2011/4）
- エクスコムグローバル、約11万人のクレジットカード情報が流出（2013/4）
- Yahoo! JAPAN、2200万件のYahoo! JAPAN IDが流出の可能性（2013/5）
- NTT ComのOCN、400万件の暗号化パスワード流出の可能性（2013/7）
- 米NASDAQ、米7-Elevenなど十数の大手企業のネットワークに侵入、1億6000万人以上のクレジットカード番号が盗難、被害額は数百万ドル（2013/7）
- LINEのNAVER、会員情報約170万件流出（2013/7）
- 2ちゃんねる、クレジットカード番号を含む約3万件の個人情報が流出（2013/8）
- @PAGES、平文パスワードを含む17万件超のユーザデータが流出（2013/9）

PostgreSQLのセキュリティ機能

FUJITSU

○	**認証**	**ユーザ認証、ホストベース認証**
○	**アクセス制御**	DAC（GRANT/REVOKE） MAC（SEPostgreSQL）
○	**通信データの暗号化**	SSL
△	**格納データの暗号化**	Pgcrypto
△	**監査**	SQL**文のロギング** （log_statement = ‘ddl | mod | all’）
×	**ファイアウォール**	SQL**インジェクションへの防衛**

なぜデータベースに暗号化が必要か

FUJITSU

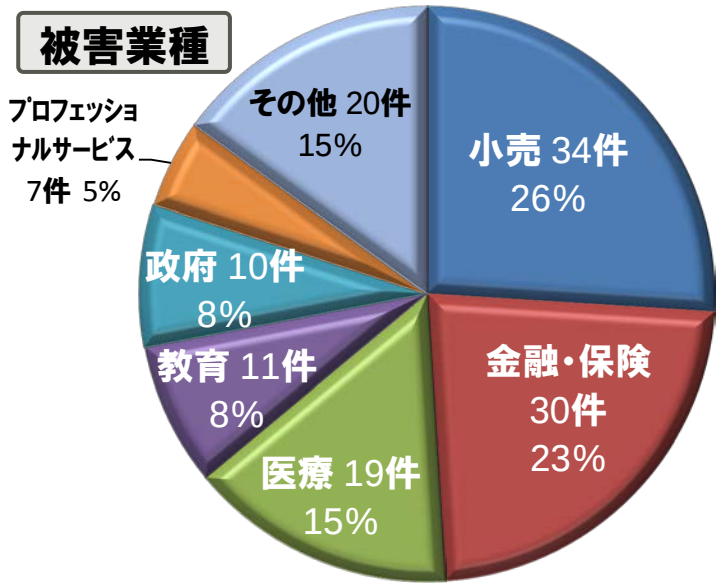
- データベースは情報の宝庫
  侵害されたレコードの96%がデータベースから（2012 Verizon Data Breach Report）
- 情報を漏えいから守りたい、情報漏えいの代償は大きい
  - 信用低下とイメージダウン → 顧客離れ、契約の打ち切り → 収益減少
  - 多大なコスト: 顧客への告知と謝罪、徹底した再発防止など信頼回復、イメージアップ活動
  - 知的財産の情報が競合企業の手に → 競争で不利に
  - 罰金や取引停止、刑事訴追
- 法律や業界規制に従わねばならない
  - 個人情報保護法: 高度な暗号化が施されていれば、個人情報の提供者への連絡を省略できる
  - 不正競争防止法: 適切な保護をしていないデジタル情報は保護対象とみなされない
  - PCI DSS: クレジットカードなどのカード会員データの保護
  - HIPAA: 医療情報の保護

暗号化の効果の実例

FUJITSU

出典：カリフォルニア州、Data Breach Report 2012

- **どれだけのデータ侵害があったか**
  - 2012年、住民500人超に影響したデータ侵害の報告は131件



侵害された情報
- 社会保障番号 56%
- クレジットカード情報 40%
- 医療情報 17%

攻撃者
- 部外者の侵入 45%
- 内部犯行 10%
  （従業員、受託業者、ベンダ、顧客）

紛失・盗難の経路
- ハードウェア 22件（17%）
- メディア 8件（6%）
- 文書 6件（5%）

- **データを暗号化していれば・・・**
  - 影響を受けた250万人のうち、140万人は危険にさらされなかった
  - 28%の侵害は告知が不要だった

クラウドでデータ暗号化は一般的に

FUJITSU

- クラウドサービスと利用者との間の通信データは、すでにSSLで暗号化
- クラウドで格納データの暗号化も求められている
- 格納データを暗号化するクラウドサービスの例
  - Amazon S3: APIでデータ書き込み時に暗号化を指定
  - Google Cloud Storage: 自動的に全データを暗号化
  - Oracle Database Cloud Service: TDEで自動的に全データを暗号化
  - Amazon RDS for Oracle: TDEで自動的に全データを暗号化

透過的データ暗号化(TDE)の概要

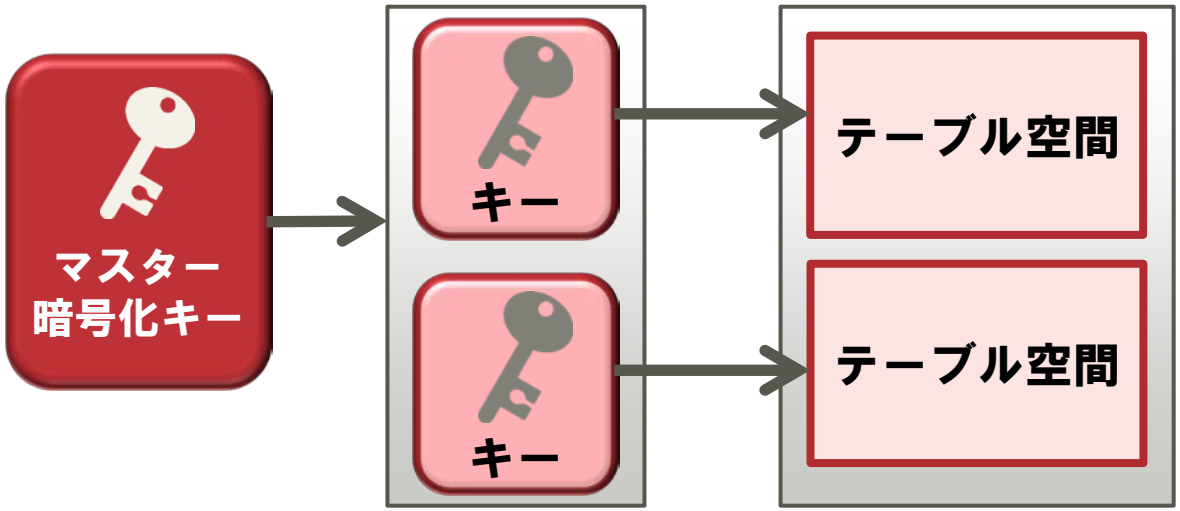
FUJITSU

- TDE = Transparent Data Encryption
- **指定テーブル空間内のすべてのユーザデータ、WAL、一時ファイルが暗号化**
  → ディスク/ファイルが盗まれても、情報は漏えいしない
- **バックアップデータも暗号化**
  → バックアップ・メディアが盗まれても、情報は漏えいしない
- **強力な暗号化アルゴリズム: キー長128/256ビットのAES**
- **ユーザの意識やアプリケーションの変更なしでデータが自動的に暗号化/復号**
  データファイルへの読み書き時にデータを暗号化・復号、アプリケーションでのキー管理が不要
- **ハードウェアに基づく暗号化/復号の高速化**
  Intel Xeonプロセッサの5600番台以降に搭載されたAES-NIで暗号化と復号のオーバヘッドを極小化
  →性能を気にせず、アプリケーションのすべてのデータを暗号化できる
- **記憶領域のゼロ・オーバヘッド**
- **ストリーミングレプリケーションのサポート**

2層の暗号化キーとキーストア

FUJITSU

- 各暗号化キーはランダムなビット列で、個別のキーストアというファイルに暗号化して格納
- 各テーブル空間には、そのデータを暗号化/復号するテーブル空間暗号化キーがある
- テーブル空間暗号化キーは、マスター暗号化キーで暗号化して保存
- マスター暗号化キーは、利用者が指定するパスフレーズに基づいて暗号化



フルディスク暗号化(FDE)との比較

FUJITSU

- FDE = Full Disk Encryption
  - ハードウェアやソフトウェアでパーティションまたはディスク全体を暗号化
  - 例: 自己暗号化HDD、OSSのTrueCrypt、暗号化ファイルシステム

比較項目	FDE	TDE
OS侵入への耐性	OSにログインできた攻撃者は、復号されたデータをファイルから読み出せる（OSのアクセス制御は受ける）	ファイルの内容は暗号化されたままのため、攻撃者は内容を解読できない
バックアップ	復号されたファイルをコピー → バックアップ・メディア上のデータは平文になる	暗号化されたファイルをcpやpg_basebackupでコピー → バックアップも暗号化されている
効率性	ディスク上のすべてのデータを暗号化	ユーザデータが格納されたテーブルやインデックスのみを暗号化 → 暗号化と復号の処理が最少限に抑えられるため、よりよい性能が得られる

pgcryptoとの比較

FUJITSU

比較項目	pgcrypto	TDE
アプリの変更	選別した列の値を暗号化または復号するためにSQL文を書き換える → パッケージ・アプリケーションや古いカスタム・アプリケーションでは不可能	アプリケーションの処理を変更せずに、すべてのデータを容易に暗号化
キー管理	アプリケーションがキーを管理する必要がある	データベースサーバがキーを管理するため、アプリケーションでの意識は不要
インデックス走査	暗号化した列のインデックスは、レンジスキャンには使えない	このような制限はない
処理オーバヘッド	SQL文の実行のたびにDBキャッシュ内のデータを暗号化/復号 → オーバヘッド大	ディスクへの読み書きのときだけ暗号化/復号、DBキャッシュのアクセスではオーバヘッドなし
記憶域オーバヘッド	暗号化のパディングのために、暗号化データが平文より大きくなる	暗号化してもデータの大きさが変わらないため、ストレージのオーバヘッドはない

TDEの性能 - 測定の方法と環境

FUJITSU

- **暗号化あり/なし、AES-NIあり/なしでOLTP、バッチ、ロードの性能を比較**
  - **データベース・サーバ**

    [**ハードウェア**]
    - **サーバ**: PRIMERGY RX300 S7
    - CPU: Intel Xeon E5-2690 2.90 GHz （8**コア**） × 2、HT**有効**
    - DRAM: 160 GB
    - **ストレージ**: **フラッシュ・メモリ** PCIe SSD

    [**ソフトウェア**]
    - OS: Red Hat Enterprise Linux 6.2 for Intel64 （64**ビット**）
    - DBMS: PowerGres Plus 9.1 （64**ビット**）
  - **クライアント**
    - **データベース・サーバ上でクライアントを実行**

TDEの性能 - OLTP

FUJITSU

- スケールファクタ500(7.5GB)、64クライアント、16スレッドで5分間pgbenchを実行
- ディスクへの読み書きが少ないOLTPでは、暗号化のオーバヘッドは3%未満

OLTP性能

条件	AES-NI	tps	-tps(%)	平均RT	90%th RT
暗号化なし	ー	22326	ー	2.928	4.555
AES256	無効	21831	-2.2%	2.932	4.584
AES256	有効	22132	-0.9%	2.889	4.538
AES128	無効	21925	-1.8%	2.915	4.586
AES128	有効	21947	-1.7%	2.911	4.569

tps: 1秒あたりに実行したトランザクション数
-tps(%): WAL多重化なしの場合と比較したtps低下率
RT: ミリ秒単位の応答時間

TDEの性能-バッチ処理、ロードの方法

FUJITSU

- **バッチ処理**
  - **5000万行(6.5GB)のテーブル全体の更新時間**

```
UPDATE pgbench_accounts SET abalance = abalance + 100;
CHECKPOINT;
```

- **データロード**
  - **5000万行(6.5GB)のテーブルへのロード時間(インデックス作成を含む)**

```
ALTER TABLE pgbench_accounts DROP CONSTRAINT pgbench_accounts_pkey;
BEGIN;
TRUNCATE TABLE pgbench_accounts;
COPY pgbench_accounts FROM '/some/file';
COMMIT;
ALTER TABLE pgbench_accounts ADD PRIMARY KEY(aid);
CHECKPOINT;
```

TDEの性能-バッチ処理、ロードの結果

FUJITSU

- ディスクへの読み書きが多いバッチ処理とロードでは
  - AES-NIなしではオーバヘッドは30%～50%
  - AES-NIがあるとオーバヘッドは10%未満

バッチ処理

条件	AES-NI	時間（秒）	時間増加率(%)
暗号化なし	ー	361.333	ー
AES256	無効	515.911	42.8%
AES256	有効	380.900	5.4%
AES128	無効	487.770	35.0%
AES128	有効	376.399	4.2%

データロード

条件	AES-NI	時間（秒）	時間増加率(%)
暗号化なし	ー	156.661	ー
AES256	無効	240.821	53.7%
AES256	有効	170.769	9.0%
AES128	無効	221.177	41.2%
AES128	有効	170.906	9.1%

TDEの使い方-1: 暗号キーの準備

FUJITSU

- 1.キーストアの配置ディレクトリをpostgresql.confに設定（初回のみ）

```
keystore_location = ‘/key/store/location’
```

- 2.マスター暗号化キーを設定（初回のみ）

```
SELECT pgx_set_master_key(‘passphrase’);
```

- 3.キーストアをオープン（サーバ起動毎）

```
SELECT pgx_open_keystore(‘passphrase’);
```

TDEの使い方-2: データの暗号化

FUJITSU

- 1.暗号化テーブル空間を作成
  - あらかじめ暗号化アルゴリズムを設定しておき、テーブル空間を作成する

```
SET tablespace_encryption_algorithm = 'AES128';
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir';
```

- 2.暗号化テーブル空間にテーブルやインデックスを作成
  - 暗号化テーブル空間に作成されたリレーションは自動的に暗号化される
    - 例1: 作成時に暗号化テーブル空間を指定

```
CREATE TABLE my_table (...)
TABLESPACE secure_tablespace;
```

    - 例2: 作成時にはテーブル空間を明示せず、デフォルト・テーブル空間を使用

```
SET default_tablespace = 'secure_tablespace';
CREATE TABLE my_table (...);
```

TDEの使い方-3: 管理

FUJITSU

- 1.暗号化されているテーブル空間の確認

```
SELECT spcname, spcencalgo
FROM pg_tablespace ts, pgx_tablespaces tsx
WHERE ts.oid = tsx.spctablespace;
```

- 2.マスター暗号化キーの変更
  - 米国NIST発行の"NIST Special Publication 800-57"では、マスター暗号化キーは1年ごとに1度変更することが推奨されている

```
SELECT pgx_set_master_key('passphrase');
```

- 3.キーストアのパスフレーズの変更

```
SELECT pgx_set_keystore_passphrase('old_passphrase', 'new_passphrase');
```

TDEの使い方-4: キーストアの自動オープン

FUJITSU

- 1.手動でのキーストアのオープン
  - 方法1: サーバを起動してからSQL関数を実行

```
$ pg_ctl start
SELECT pgx_open_keystore('passphrase');
```

  - 方法2: サーバ起動時にパスフレーズを入力してオープン

```
$ pg_ctl --keystore-passphrase start
パスフレーズを入力してください:
```

- 2. DBサーバ起動の度パスフレーズを入力するのは煩雑
  HAクラスタでの自動フェイルオーバには向かない

⇨ キーストアの自動オープンを有効にしておくとパスフレーズの入力が不要に

```
$ pgx_keystore --enable-auto-open keystore.ks
パスフレーズを入力してください:
キーストアの自動オープンが有効になりました
```

構築済みアプリケーションの導入

FUJITSU

- 既存のアプリケーションの導入時に、そのデータを暗号化したい。しかし・・・
  - 購入したパッケージ・アプリケーションは変更できない
  - 自社開発の古いカスタム・アプリケーションは、当初の開発者がおらず、変更のコスト大
  - どのテーブルや列を暗号化すべきか判断することが困難
- TDEならアプリケーション変更なしですべてのデータを暗号化できる
  - 1.アプリケーションのための所有者ユーザとデータベースを作成

```
CREATE USER app_user ...;
CREATE DATABASE app_db ...;
```

  - 2.アプリケーションのデータを格納する暗号化テーブル空間を作成

```
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE app_tablespace LOCATION '/app/data';
```

  - 3.アプリケーションの所有者ユーザのデフォルト・テーブル空間として、暗号化テーブル空間を設定

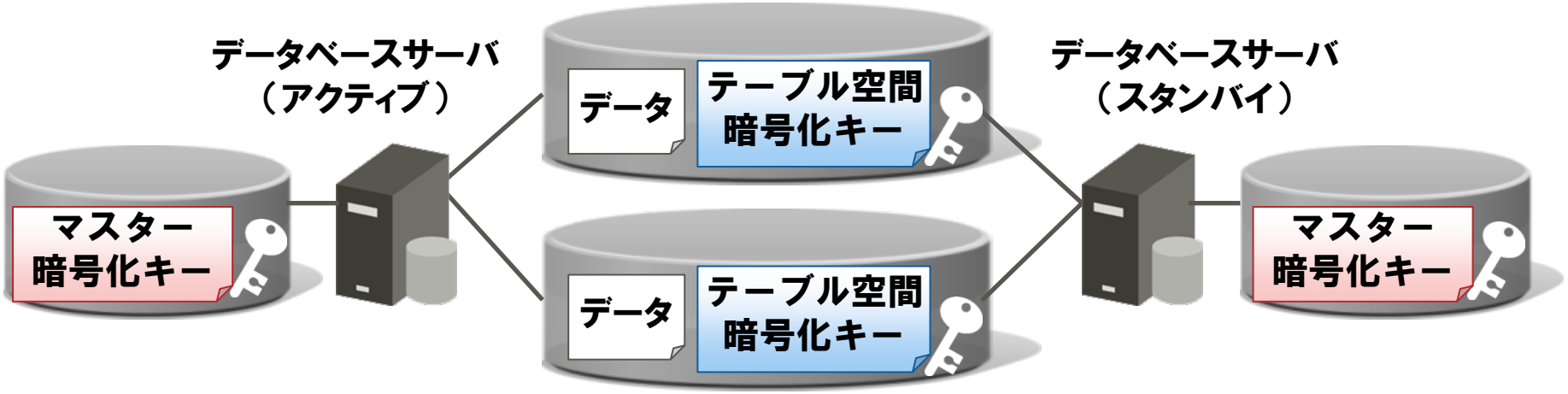
```
ALTER USER app_user SET default_tablespace = 'app_tablespace';
ALTER USER app_user SET temp_tablespaces = 'app_tablespace';
```

  - 4.アプリケーションをインストール

共有ディスク型HAクラスタとTDE

FUJITSU

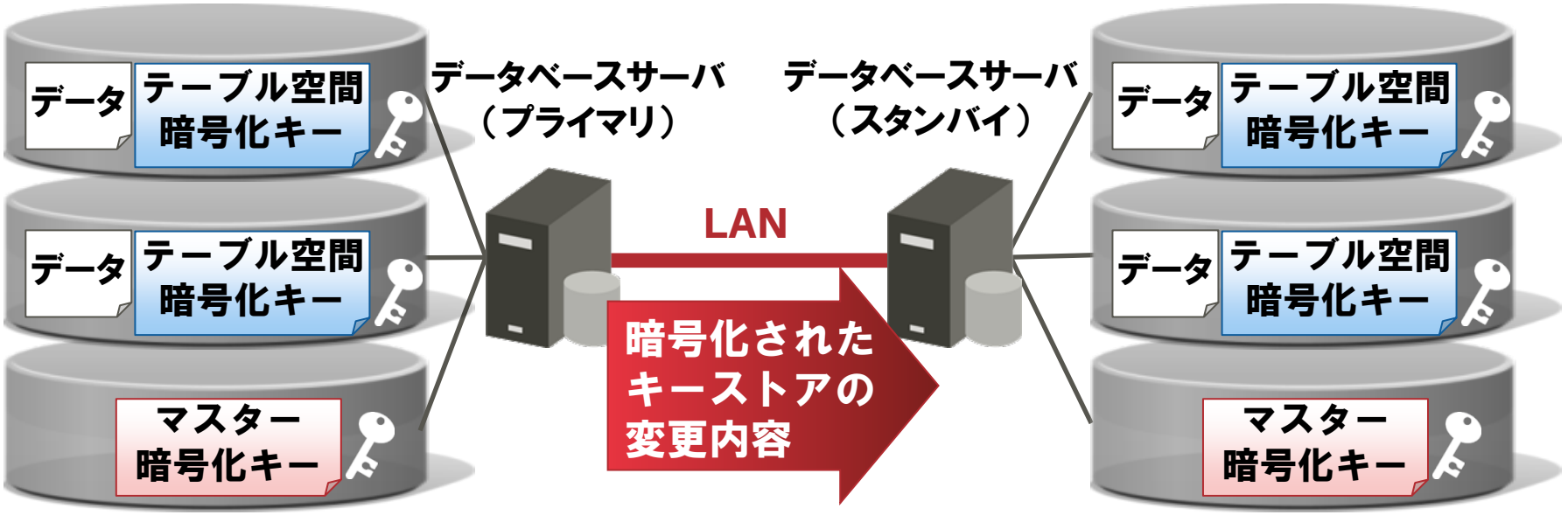
- キーストア・ファイルはアクティブ/スタンバイサーバそれぞれのディスクに配置
- アクティブサーバでマスター暗号化キーを設定・変更、パスフレーズを変更
- 変更されたキーストア・ファイルを、アクティブサーバからスタンバイサーバに手動でコピー
- コピーしたら、スタンバイサーバで再び自動オープンを有効化



ストリーミングレプリケーションとTDE

FUJITSU

- キーストア・ファイルはプライマリ/スタンバイサーバそれぞれのディスクに配置
- プライマリサーバでマスター暗号化キーを設定・変更、パスフレーズを変更
- キーストアの変更が暗号化されたWALとしてスタンバイに伝搬し、自動的にスタンバイのキーストアを更新
- スタンバイサーバでは、最初にプライマリサーバからキーストアを手動でコピーするだけでよい



TDEの実装

FUJITSU

- 暗号化・復号にはOpenSSLを使用
  OpenSSLがプロセッサの種類を判別し、自動的にAES-NIを使用
- 永続データファイル、一時ファイルの暗号化
  - 各サーバプロセスがディスクへの読み書き時にページ全体を暗号化/復号
  - ユーザデータを含む主フォークを暗号化、VM/FSMフォークは暗号化しないことでオーバヘッド最小化
- WALの暗号化
  - WALバッファにのせる際にユーザデータ部分のみを暗号化、REDOのときに復号
  - 更新対象データと同じ暗号化アルゴリズムでWALも暗号化
- 暗号化キーの暗号化
  - テーブル空間暗号化キーはマスター暗号化キーを使ってAES-256で暗号化
  - マスター暗号化キーはパスフレーズを使ってPBKDF2で暗号化

今後の強化ポイントの候補

FUJITSU

- HSM(Hardware Security Module)による堅牢なキー管理
  クラウドではAmazon CloudHSMとの連携など
- TPM(Trusted Platform Module)による堅牢なキー管理
  仮想環境では仮想TPM
- pg_dumpおよびpg_dumpallの出力ファイルの暗号化
- COPYコマンドの出力ファイルの暗号化
- テーブル空間pg_global、pg_defaultの暗号化

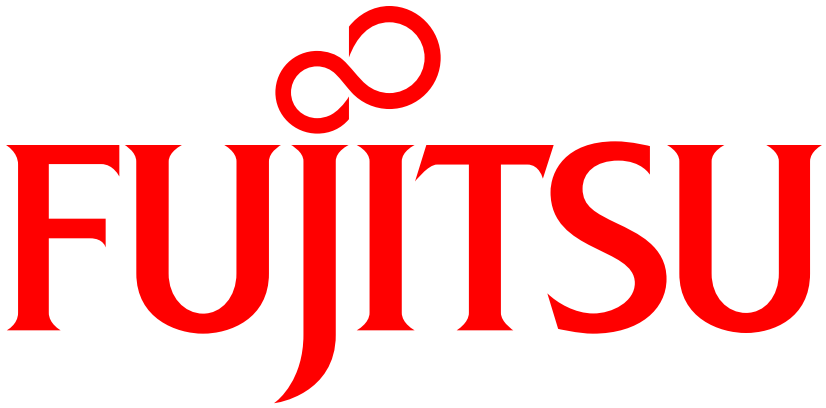
TDEを実装した製品のご紹介

FUJITSU

PowerGres Plus 9.1

SRA OSS, Inc.の製品
PostgreSQL 9.1をベースに、
信頼性とセキュリティ、使いやすさ、安心サポートをプラス

- SRA OSS, Inc.
  http://powergres.sra.co.jp/s/ja/product/PlusV91.php
- **富士通株式会社**
  http://software.fujitsu.com/jp/powergresplus/
- **その他、複数の販売代理店から販売中**



shaping tomorrow with you