Oracle Transparent Data Encryption
im
Compliance Umfeld

Thomas Knauber
Atos Worldline GmbH
Frankfurt

Schlüsselworte

Transparent Data Encryption, TDE, Compliance, Security, PCI-DSS, HSM, dbms_redefinition

Einleitung

Zertifizierungen nach Security Standards wie z.B. PCI-DSS (Payment Card Industry Data Security Standard) erfordern jedes Jahr den Einsatz von immer neuen Technologien und Features um den wachsenden Sicherheitsbedürfnissen gerecht zu werden. Ein vorgeschriebener Baustein ist dabei auch die Absicherung der gespeicherten Daten durch Verschlüsselung. Oracle bietet dafür den Einsatz von Transparent Data Encryption (TDE) an.

TDE im PCI-DSS Umfeld

Die PCI Zertifizierung ist ein umfangreiches Regelwerk, dessen Einhaltung Unternehmen die Kreditkarten-Daten verarbeiten, jährlich in einer Zertifizierung nachweisen müssen. Die Anforderungen verändern sich dabei ständig.
Diese Regeln sind oft sehr allgemein formuliert. Eine dieser Security-Regeln fordert z.B. nur, dass gespeicherte Kreditkarten-Daten gesondert vor Zugriffen geschützt werden müssen. Seit Jahren wird daher erwartet, dass auch Daten die in einer relationalen Datenbank wie Oracle gespeichert werden, davor geschützt werden müssen mit Fremd-Tools aus den Datafiles ausgelesen zu werden. Dazu ist es erforderlich, dass die Daten verschlüsselt gespeichert werden. Oracle erfüllt die PCI Anforderungen und bietet dafür seit Version 10.2 das Feature TDE an. Entscheidend ist and dieser Stelle das Wort „Transparent“. Dies bedeutet hier, dass die Verschlüsselung für die Datenbank und die Applikationen die diese verwenden transparent ist. TDE verschlüsselt also nicht die eigentlichen Daten in den Tabellen, sondern nur deren Speicherung in den Datafiles.
TDE ist also keinesfalls ein umfassendes Security- oder Verschlüsselungs-Feature sondern kann nur ein Baustein eines Security-Gesamtkonzeptes sein.

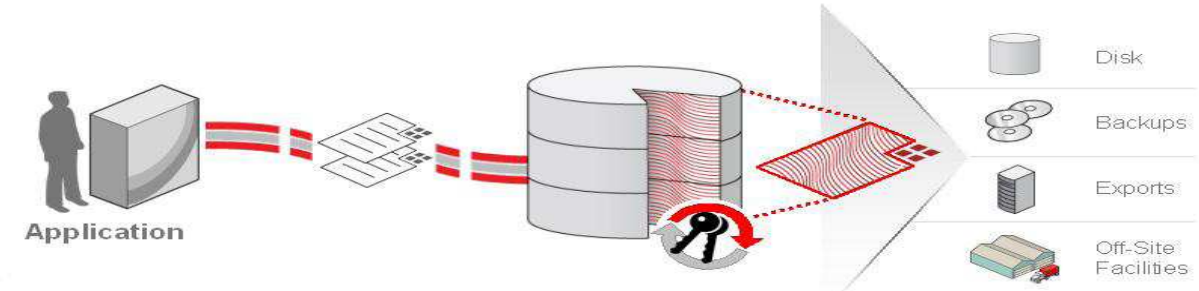

Abb. 1: Oracle Transparent Data Encryption (Quelle Oracle)

TDE – Funktionsübersicht

TDE realisiert Verschlüsselung über Keys und Verschlüsselungs-Algorithmen. Dabei wird eine 2-Schicht Architektur verwendet:
Es gibt einen sogenannten Master-Key und einen oder mehrere Tabellen- bzw. Tablespace-Keys. Der Master-Key wird dazu benutzt die Tabellen- und Tablespace-Keys zu verschlüsseln. Gespeichert wird der Master Key in einem externen Security Modul z.B. einer Wallet oder einem Hardware Security Module (HSM). Der Master-Key wird bei der initialen Einrichtung von TDE generiert.
TDE kann auf Column oder seit Version 11 auch auf Tablespace-Ebene verschlüsseln. Bei Column-Encryption muss man gezielt die Columns einer Tabelle festlegen, die man verschlüsseln möchte, währen die Tablespace Verschlüsselung alle Daten im Tablespace verschlüsselt abspeichert.

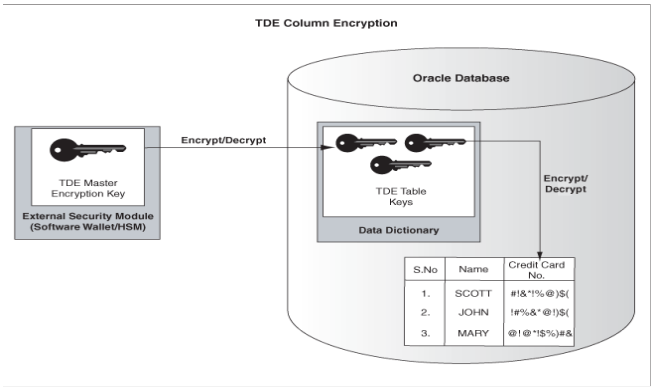

Abb. 2: Bsp. TDE Column-Encryption (Quelle Oracle)

Key – Confusion

Von Version 10.2 nach 11.1 und schließlich zur Version 11.2 hat sich einiges bzgl. des Master-Keys geändert. Das ist in der Dokumentation manchmal etwas verwirrend dargestellt:
In 10.2 gab es nur den Master-Key für die Column-Encryption.
Seit 11.1 gibt es auch die Tablespace-Encryption. Wird diese initial eingerichtet, wird ein **separater** Master-Key in der Wallet gespeichert. In 11.2 wird nun ein sogenannter Unified-Master-Key eingeführt. Das bedeutet, dass bei einer Re-Key Operation (Generierung eines neuen Schlüssels) evtl. vorhandene separate Schlüssel für Column- und Tablespace-Encryption zusammengeführt werden. Beim initialen Einrichten unter 11.2 wird sofort ein Unified-Master-Key generiert.
Master-Key und Wallet Passwort sind unabhängig voneinander. Der Master-Key wird von einem Zufallsgenerator generiert. Das Wallet-Passwort dient nur zur Verschlüsselung der Wallet.

Regelmäßige Re-Key Operationen

Inzwischen schreibt PCI jährliche Re-Key Operationen für die Master-Keys vor. Dies ist für Tablespace-Verschlüsselung erst seit 11.2 möglich:

Master Re Key Support	Column-Encryption	Tablespace-Encryption
10gR2	Yes	n/a
11gR1	Yes	No
11gR2	Yes	Yes

Abb.3: Master Key Re-Key Support

In der Praxis bedeutet dies, dass man sich genau überlegen muss welche TDE-Encryption man verwenden möchte, bzw. ob man durch Compliance Re-Key Vorschriften zu einem Upgrade gezwungen wird. Ein Re-Key der eigentlichen Keys ist aktuell nicht vorgeschrieben und wäre aufgrund des Aufwandes auch nicht ganz einfach (Update aller Tabellen-Rows bzw. Migration in einen neuen Tablespace).

Welche Option ist die Richtige ?

Bei Column-Encryption muss man genau wissen welche Columns man verschlüsseln möchte. Nicht immer lässt sich das am Namen (`CardNbr` etc.) ablesen und nicht immer ist jemand in der Lage alle Spalten zu benennen, die sensible Daten enthalten.
Verschlüsselte Spalten bringen auch einige Limitierungen für den Gebrauch von Indices mit sich. Es sind nur B-Tree Indices auf verschlüsselte Spalten möglich und die indizierten, verschlüsselten Spalten dürfen auch nur mit dem Gleichheitsoperator verwendet werden. Es können auch nur Spalten indiziert werden, die mit ‚no salt' verschlüsselt wurden. Range-Scans („between" Operations) sind auf verschlüsselten Indices nicht möglich. Man sollte sich also vorher überlegen, ob solche Scans z.B. auf Kreditkarten-Nummern von der Applikation benötigt werden bzw. die Auswirkungen testen. Hier ist eine enge Zusammenarbeit mit der Fachabteilung bzw. dem Software-Development erforderlich, sonst kann es zu erheblichen Performanceproblemen kommen.

Bei Tablespace-Verschlüsselung hat man diese Probleme nicht. Die Ausführungspläne bleiben gleich, da existierende Indices nach der Verschlüsselung weiterhin korrekt funktionieren.
Es gilt zu beachten, dass nur neue Tablespaces verschlüsselt werden können. D.h. bestehende Objekte müssen z.B. per Data Pump, CTAS oder bei partitionierten Tabellen mit MOVE, dbms_redefinition etc. in den neuen, verschlüsselten Tablespace migriert werden. Es kann also im Betrieb zu erheblichen Ausfallzeiten durch die Migration kommen. Auch bei der Column-Encryption muss man mit dbms_redefinition arbeiten, um größere Ausfallzeiten zu vermeiden. Der Einsatz von dbms_redefinition hat bei großen Tabellen diverse Tücken. Wir haben das mehrfach ausprobiert und raten dringend dazu das geplante Vorgehen vorher auf einem Testsystem mit realistischer Datenmenge durchzuspielen. Es funktioniert zwar grundsätzlich gut, aber es gibt diverse Überraschungen (TEMP-Tablespace etc.). Vor solchen Operationen ist natürlich auch eine Datensicherung Pflicht.
Weiterhin sollte man bei Column-Encryption großer Tabellen bedenken, dass laut Oracle der Platzbedarf pro verschlüsseltem Wert zwischen 1 und 52 Bytes höher ist.
Wir haben uns entschieden seit Version 11.2 nur noch Tablespace-Encryption einzusetzen. Das Handling ist letztendlich einfacher und flexibler. Nur wenn man z.B. genau weiß, dass man nur eine Spalte in einer Tabelle verschlüsseln muss und diese auch nicht performancekritisch ist, würde ich zu Column-Encryption raten.

Wallet Management

Für den Betrieb einer Datenbank bei der TDE eingesetzt wird, ist das Management der Wallet von zentraler Bedeutung. Zum einen gibt es einige DBA-Operationen die dadurch beeinflusst werden, und zum anderen gibt es auch hier Randbedingungen der PCI-Zertifizierung.
Zunächst stellt sich die Frage wo die Wallet gespeichert und wie sie dort abgesichert wird. Die Location der Wallet kann man über die `sqlnet.ora` Datei festlegen oder die Default Location (`$ORACLE_BASE/admin/$ORACLE_SID/wallet`) benutzen. Es gab zunächst Probleme mit der Default Location, die vor 11.2 erst nach dem Einspielen eines Patches korrekt erkannt wurde. Wir haben uns aktuell für die Default Location entschieden. Das Wallet Verzeichnis ist bei uns mit `,chmod 000'` abgesichert. Der User `oracle` hat dann über eine ACL entsprechende Rechte.

Sehr wichtig ist auch die Sicherung der Wallet. Die Wallet sollte immer gleichzeitig mit der Datenbank gesichert werden, aber niemals mit dieser auf dem gleichen Sicherungsmedium gespeichert werden. Vor jeder Änderung der Wallet sollte diese auf jeden Fall gesichert werden.
Die Änderung des Wallet Passworts musste vor 11.1.0.7 über den Wallet Manager erfolgen. Seit 11.1.0.7 kann man dazu das `,orapki'`–Utility verwenden.

Betriebliche Auswirkungen

Seit Version 11.1 hat man die Möglichkeit ein sogenanntes Auto-Open-Wallet anzulegen. Dies wird dann automatisch beim Start der DB geöffnet, und funktioniert nur auf der Maschine auf der es angelegt wurde. Hier hatten wir allerdings das Problem, dass die PCI Auditoren ein Auto-Open-Wallet nicht zugelassen haben. Wir sind deshalb gezwungen manuell mit dem Vier-Augen-Prinzip zu arbeiten. Das bedeutet, dass der DBA das Wallet Passwort nicht kennt und derjenige, der das Passwort hat, nicht automatisch das `,alter system'` Recht hat. Dies führt z.B. nach einem Crash Recovery zu einigen Schwierigkeiten. Wird z.B. nach einem Server-Crash die Datenbank automatisch gestartet kommt es zu Fehlermeldungen, da die verschlüsselten Tablespaces nur bei geöffneter Wallet recovert werden können. Es muss also manuell eingegriffen werden. Wenn man dann das geforderte Vier-Augen-Prinzip anwenden will, wird es schnell aufwendig und zeitraubend. Der Oracle Support schlägt hier eine Lösung vor, die ich im Vortrag erläutern werde.

Hardware Security Module

Der optimale Weg scheint die Nutzung eines Hardware Security Moduls (HSM) zu sein. Dabei lässt PCI auch das Auto-Open zu und seit 11.2 ist auch der geforderte Re-Key des Unified-Master-Keys im HSM möglich. Die Datenbank schickt die Keys für die Column oder Tablespace Encryption zum HSM. Dort werden sie entschlüsselt und wieder zur DB zurückgeschickt. Die Keys für die Column-Encryption werden aber nicht gecached. D.h. bei jedem Zugriff müssen die Table-Keys erneut vom HSM entschlüsselt werden. Die Tablespace-Keys werden nur einmal entschlüsselt und verbleiben dann im DB-Cache.

Fazit

Oracle TDE als ein Baustein von Security Maßnahmen stellt den DBA vor einige Entscheidungen bei Einführung und Betrieb, insbesondere wenn die Randbedingungen durch Compliance Richtlinien zusätzlich verschärft werden. Ich möchte mit diesem Vortrag die zentralen Aspekte von TDE erläutern und aus unserer Erfahrung bei Einführung und Betrieb Denkanstöße bzw. Entscheidungshilfen für Aufbau und Betrieb von Datenbanken mit TDE geben.

Kontaktadresse:

Thomas Knauber
Atos Worldline GmbH
Hahnstraße 25
D-60528 Frankfurt/Main

Telefon: +49 (0) 69-6657 1485
E-Mail Thomas.Knauber@atos.net
Internet: www.atosworldline.de