Oracle Advanced Security Technical White Paper

An Oracle White Paper
June 2007

NOTE:

Oracle Advanced Security 11g
Technical White Paper

Oracle Advanced Security 11g
Technical White Paper

INTRODUCTION

Operating in today's global business environment presents numerous security and compliance challenges. The economic benefits of outsourcing must be coupled with adequate protections to safeguard intellectual property and privacy related information. In recent years there have been numerous incidents of identity theft and credit card fraud resulting in damages reaching into the tens of millions of dollars. Protecting against these types of threats requires security solutions that are transparent by design. Universities and health care organizations are tightening security around personally identifiable information (PII) such as social security numbers while retailers are working to comply with PCI-DSS requirements. Oracle Advanced Security provides transparent, standards-based security that protects data on the network, on disk and on backup media.

ORACLE DATABASE ENCRYPTION OVERVIEW

Encryption is a key component of the defense-in-depth principle and is important for protecting data in transit and at rest. Oracle first introduced database encryption API's in Oracle8i. Oracle database encryption API's are used by many customers today to encrypt sensitive application data. Achieving transparency while using an encryption API requires embedding function calls within the application itself or using pre-insert database triggers. Application views may also be required to decrypt the data before it reaches the application. In addition, key management must be handled programmatically.

Oracle Advanced Security Transparent Data Encryption (TDE), first introduced in Oracle Database 10g Release 2, is the industry's most advanced encryption solution. TDE provides built-in key management and complete transparency for encryption of sensitive application data. The database encryption process is turned on using DDL commands, completely eliminating the need for application changes, programmatic key management, database triggers, and views.

Package Feature	DBMS OBFUSCATION TOOLKIT (Oracle 8i and higher) SE & EE	DBMS CRYPTO (Oracle Database 10g R1 and higher) SE & EE	Oracle Advanced Security Transparent Data Encryption EE Only Option
Cryptographic algorithms	DES, 3DES	DES, DES, AES, RC4, 3DES_2KEY(1)	3DES, AES (128, 192, and 256 bit)
Padding forms	none supported	PKCS5, zeroes	PKCS5(2)
Block cipher chaining modes	CBC	CBC, CFB, ECB, OFB	CBC(2)
Cryptographic hash algorithms	MD5	SHA-1, MD4(1), MD5(1)	SHA-1(2)
Keyed hash (MAC) algorithms	none supported	HMAC_MD5, HMAC_SH1	n/a
Cryptographic pseudo-random number generator	RAW, VARCHAR2	RAW, NUMBER, BINARY_INTEGER	n/a
Database types	RAW, VARCHAR2	RAW, CLOB, BLOB	All but: OBJ., ADT, LOB

1) Provided for backward compatibility
2) Used Internally, not available to developers

Table 1. Oracle Database Encryption Overview

TRANSPARENT DATA ENCRYPTION

TDE encrypts data before it's written to disk and decrypts data before it is returned to the application. The encryption and decryption process is performed at the SQL layer, completely transparent to applications and users. Subsequent backups of the database files to disk or tape will have the sensitive application data encrypted. Optionally, TDE can be used in conjunction with Oracle RMAN to encrypt the entire Oracle database during backup to disk.

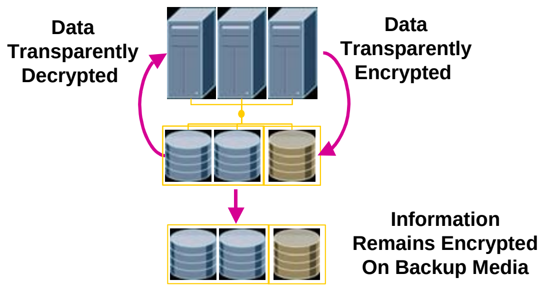
Data Transparently Decrypted
Data Transparently Encrypted
Information Remains Encrypted On Backup Media

Fig 1. Transparent Data Encryption Overview

Transparent Data Encryption Benefits

1. Built-in key management
2. Transparent encryption of sensitive application columns
3. Transparent encryption of entire tables paces **(New in 11g)**
4. Transparent encryption of Secure Files/LOBS **(New in 11g)**
5. Hardware Security Module (HSM) integration **(New in 11g)**

Key Management Overview

TDE automatically creates an encryption key when an application table column is encrypted. The encryption key is table specific. If more than one column in a single table is encrypted, the same encryption key is used for all columns. Each table key is stored in the Oracle data dictionary and is encrypted using the TDE master encryption key. The master encryption key is stored outside of the database, in an Oracle Wallet, a PKCS#12 formatted file that is encrypted using a password supplied either by the designated security administrator or DBA during setup. New in Oracle Database 11g Advanced Security is the ability to store the master key in an HSM device using the PKCS#11 interface.

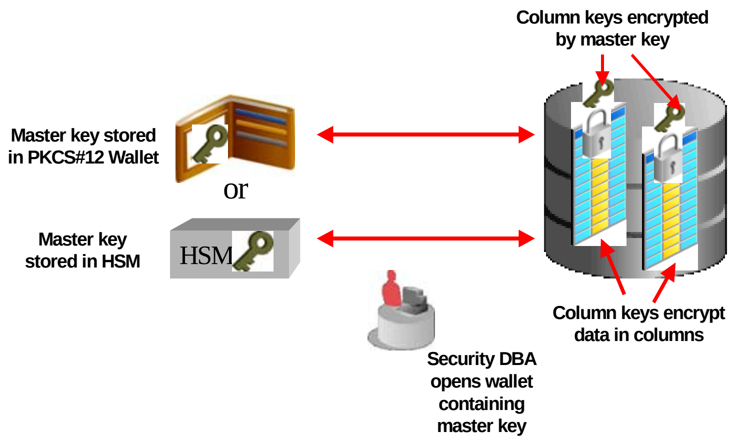
Column keys encrypted by master key
Master key stored in PKCS#12 Wallet
or
Master key stored in HSM
HSM
Column keys encrypt data in columns
Security DBA opens wallet containing master key

Fig 2. TDE Key Management Architecture

IMPLEMENTATION STEPS

Since TDE is transparent to existing application code (database triggers and views are not required) the encryption process is simple compared to traditional API based encryption solutions. The following steps can be used to apply TDE:

1. Initialize the master key
2. Identify the sensitive data to encrypt (PII data, credit cards)
3. Verify TDE supports the data type and check foreign key usage
4. Encrypt the sensitive data using TDE

Initialize the Master Key

Master keys are specific to each database. However, any master key can be copied to a secondary database as long as a master key has not been previously established for the secondary database. The master key must be created before any application tables can be encrypted. The syntax to initialize the master key is as follows:

```
SQL> alter system set key identified by “password”;
```

This command creates a wallet and uses the password to encrypt the Wallet, based on the recommendations of the PKCS#5 standard. The Oracle Wallet stores a history of retired master encryption keys and makes them available when data encrypted under an older key is read back from a backup tape.

Open the Oracle Wallet

The Wallet containing the master encryption key must be opened before the database can decrypt the table keys to encrypt or decrypt application data. The database can be up and running without the Wallet being opened. However, attempts to access encrypted data will return an error. Closing the Wallet can be useful during maintenance operations when access to the database must be granted to a support person.

Changing the Master Key

The master key can be changed by issuing the alter system command again.

```
SQL> alter system set key identified by “password”;
```

Changing the master key will re-encrypt all of the table keys in the Oracle data dictionary. The PCI Data Security Standard (DSS) 1.1 requires ‘updating the encryption key frequently, at least annually’. Changing the master key will re-encrypt the column keys using the new master encryption key; the encrypted data is not touched.

Changing the Wallet Password

The wallet password can be changed independently of the master encryption key; it is only used to encrypt the wallet file on disk. Use either Oracle Wallet Manager (started with 'owm' on the command line) or the 'orapki' command line.

Identify sensitive data

Identifying PII related data such as social security numbers and credit card can be difficult, especially in a complex application. One technique that can be useful is to search the Oracle data dictionary for column names and data types that are frequently used to store such information.

```
SQL>  select column_name, table_name, data_type from
dba_tab_cols where column_name like '%SOCIAL%' or
column_name like '%SSN%' or column_name like '%SECNUM%' or
column_name like ''%SOC%' and owner='<owner>';
```

Verify TDE data type support

TDE support the most common data types used in the database. These include:

VARCHAR2	CHAR	DATE
NUMBER	NVARCHAR2	NCHAR
RAW	RAW	SECUREFILES (LOBS)
BINARY_DOUBLE	BINARY_FLOAT	

Check Foreign Key Usage

TDE can't be used to encrypt columns that are used in a foreign key. Verifying whether a column is used as part of a foreign key can be accomplished by examining the Oracle data dictionary.

```
select A.owner, A.table_name, A.column_name, A.constraint_name
from dba_cons_columns A, dba_constraints B
where
A.table_name = B.table_name and
A.column_name = 'YOURCOLNAME' and
B.constraint_type = 'R';
```

Encrypting Data Using TDE

To encrypt an existing column:

```
SQL> alter table customers modify (credit_card encrypt);
```

Read consistency is maintained while the encryption transaction is being executed, enabling select (read) operations to continue. DML transaction (insert, update, delete) performed during the encryption transaction will require 'online re-definition'.

Creating a new table with an encrypted column is easy. The default encryption algorithm is AES192.

```
SQL> create table billing_information (
first_name varchar2(40)
,last_name varchar2(40)
,card_number varchar2(19) encrypt using 'AES256');
```

Transparent Data Encryption works with indexes for equality searches, minimizing the overhead of searching on an encrypted column.

```
SQL> create index cust_idx on customers (credit_card);
```

When an indexed column is to be encrypted, it is recommended to first drop the existing index, encrypt the column, and lastly re-build the index.

Changing the Table/Column Key

The table or column key, key size and algorithm can be changed independently by issuing the alter table command:

```
SQL> ALTER TABLE employee REKEY;
SQL> ALTER TABLE employee REKEY USING 'AES256';
SQL> ALTER TABLE employee ENCRYPT USING 'AES128';
```

Changing the table or column key will re-encrypt all encrypted data stored in the table.

Best Practices When First Encrypting Data

During the lifetime of a table, data may become fragmented, re-arranged, sorted, copied and moved within the table space; this can result in old, inaccessible copies of data residing in unused data blocks within the database file. When encrypting an existing column, only the most recent 'valid' copy is encrypted, possible leaving behind older clear-text versions. To minimize the risk associated with the old clear text copies created prior to encryption, Oracle recommends creating a new tablespace, moving the application table to the new tablespace, and dropping the old tablespace.

1. Backup your database completely
2. Create a new table space specifying a new data file
3. Encrypt the sensitive column in the original table
4. Repeat step 3 for all tables that contain sensitive columns
5. Move the tables from the original table space into the new table space
6. Drop the original table space using the data file option. Optionally, do not use the '... with datafile' option with the 'drop tablespace' command, but use 'shred' or other platform specific commands to securely delete the old data files on the operating system.

Using an operating system command such as 'shred' lowers the probability of being able to find ghost copies of the database file, generated by either the operating system, or storage firmware.

BACKUP ENCRYPTION WITH RMAN

For improved security, RMAN backups created as backup sets can be encrypted using Oracle Advanced Security. Encrypted backups cannot be read if they are obtained by unauthorized people. Any RMAN backups as backup sets can be encrypted. However, image copy backups cannot be encrypted.

Encrypted backups are decrypted automatically during restore and recover operations, as long as the required decryption keys are available, by means of either a user-supplied password or the Oracle Encryption Wallet.

To use RMAN encryption, the COMPATIBLE initialization parameter at the target database must be set to at least 10.2.0.

When the *backup backupset* command is used with encrypted backup sets, the backup sets are backed up in their encrypted form. Because *backup backupset* just copies an already-encrypted backup set to disk or tape, no decryption key is needed during a *backup backupset* operation, and the data is never decrypted during any part of the operation. The *backup backupset* command can neither encrypt nor decrypt backup sets.

If some columns in the database are encrypted using Transparent Data Encryption, and those columns are backed up using backup encryption, then those columns will be encrypted a second time during the backup. When the backup sets are decrypted during a restore, the encrypted columns are returned to their original encrypted form. If no encryption algorithm is specified, the default encryption algorithm is 128-bit AES.

RMAN Encryption Modes

RMAN offers three encryption modes: transparent mode, password mode, and dual mode. Both transparent mode and dual mode depend upon the Oracle Encryption Wallet.

Transparent Encryption of Backups

Transparent encryption can create and restore encrypted backups with no DBA intervention, as long as the required Oracle key management infrastructure is available. Transparent encryption is best suited for day-to-day backup operations, where backups will be restored at the same database that they were backed up from. Transparent encryption is the default mode for RMAN encryption.

When using transparent encryption, you must first configure the Oracle Encryption Wallet, as described in the documentation for Oracle's Transparent Data Encryption feature. After the Oracle Encryption Wallet is configured, encrypted backups can be created and restored with no further DBA intervention.

Password Encrypted Backups

Password encryption requires that the DBA provide a password when creating and restoring encrypted backups. Restoring a password-encrypted backup requires the same password that was used to create the backup. Password encryption is useful for backups that will be restored at remote locations, but which must remain secure in transit. Password encryption cannot be persistently configured. The Oracle Encryption Wallet need not be configured if password encryption is to be used exclusively.

If you forget, or lose, the password that you used to encrypt a password-encrypted backup, you will be unable to restore that backup.

To use password encryption, use the SET ENCRYPTION ON IDENTIFIED BY password ONLY command in your RMAN scripts.

Dual-mode Encrypted Backups

Dual-mode encrypted backups can be restored either transparently or by specifying a password. Dual-mode encrypted backups are useful when you create backups that are normally restored on-site using the Oracle Encryption Wallet, but which occasionally need to be restored off-site, where the Oracle Encryption Wallet is not available.

When restoring a dual-mode encrypted backup, you can use either the Oracle Encryption Wallet or a password for decryption.

If you forget, or lose, the password that you used to encrypt a dual-mode encrypted backup and you also lose your Oracle Encryption Wallet, then you will be unable to restore that backup.

To create dual-mode encrypted backup sets, specify the SET ENCRYPTION ON IDENTIFIED BY password command in your RMAN scripts.

TRANSPARENT DATA ENCRYPTION ENHANCEMENTS

Oracle Database 11g Advanced Security Transparent Data Encryption includes several important enhancements.

Additional Data Type Support

Oracle Database 11g Advanced Security Transparent Data Encryption supports encryption of the new Oracle Database 11g SecureFiles. This enables transparent encryption of scanned medical images, contracts and other highly sensitive documents stored in the database.

Tablespace Encryption

Oracle Database 11g Advanced Security Transparent Data Encryption introduces support for encryption of entire database table spaces. Table space encryption means entire application tables can be transparently encrypted. Data blocks will be transparently decrypted as they are accessed by the database.

Only new table spaces can be encrypted.

```
SQL> CREATE TABLESPACE securespace
DATAFILE '/home/user/oradata/secure01.dbf' SIZE 150M
ENCRYPTION USING 'AES192' DEFAULT STORAGE(ENCRYPT);
```

All database objects created in the new table space will be encrypted. Table space encryption eliminates the foreign key restriction of column encryption and enables range scans on encrypted data.

Hardware Security Modules (HSM) Master Key Protection

For even tighter security, the master encryption key can be stored in a PKCS#11 compliant HSM device, which also allows multiple databases or database instances in a RAC environment that share the same encrypted data, to share the same master encryption key. Since Oracle adheres to the PKCS#11 standard, customers can choose from a wide variety of HSM providers.

When upgrading from column-level encryption with TDE in Oracle 10g R2 to tablespace encryption in Oracle11g R1, you need to perform a re-key operation, which generates a new master encryption key for the encrypted columns *and* a new master encryption key for the tablespace encryption.

When upgrading from a master encryption key that is stored in the Oracle Wallet to a master encryption key stored in an HSM device, perform the following steps:

1.) Re-key the master key to generate a tablespace encryption key just in case you need one later

2.) Change the sqlnet.ora file to:

```
ENCRYPTION_WALLET_LOCATION=
SOURCE=(METHOD=HSM)
```

3.) SQL> alter set key identified by “<user_id:password>” migrate from <wallet password>

<user_id:password> is the authentication information for the HSM device.

DataPump Encryption

Transparent data encryption can be used to encrypt the entire output contents from Oracle Datapump. This is a new feature in Oracle Database 11g. Please refer to the Oracle Datapump documentation for additional details.

Support for Oracle Streams and Logical Standby

Oracle Database 11g Advanced Security TDE supports Oracle Streams and logical standby databases. Oracle Database 10g Release 2 Advanced Security TDE provided support for physical standby only.

NETWORK ENCRYPTION

Oracle Advanced Security protects privacy and confidentiality of data over the network by eliminating data sniffing, data loss, replay and person-in-the-middle attacks. All communication with an Oracle Database can be encrypted with Oracle Advanced Security. Databases contain extremely sensitive information and restricting access by strong authentication is one of first lines of defense. Oracle Advanced Security provides strong authentication solutions leveraging a business’s existing security framework including Kerberos, Public Key Cryptography, RADIUS and DCE for Oracle Database 10g.

Industry Standard Encryption and Data Integrity

Oracle Advanced Security protects all communications to and from the Oracle Database. Businesses have a choice between using Oracle Advanced Security's native encryption/data integrity algorithms and SSL to protect data over the network. Some of the typical scenarios requiring network level encryption include:

- Database Server is a behind a firewall and users access the server via client server applications
- Communication between the application server in a DMZ and the Database which is in behind a second firewall must be encrypted

Native Encryption and Data Integrity algorithms in Oracle Advanced Security require no PKI deployment. With each subsequent release of the database, newer encryption algorithms are included as they gain industry approval. The latest addition is the Advanced Encryption Standard (AES), an algorithm improved in security and performance over DES. The complete list of Encryption and Data integrity algorithms are

- AES (128, 192 and 256 bits)
- RC4 (40, 56, 128, 256 bits)
- 3DES (2 and 3 keys; 168 bits)
- MD5
- SHA1

SSL based encryption is available for businesses that have elected to provide Public Key Infrastructure to their IT deployments. Oracle Advanced Security 10g release introduced support for the TLS 1.0 protocol. Oracle Advanced Security provides AES cipher suites with the TLS 1.0 protocol starting in Oracle Database 10g.

SSL

Oracle implements the SSL protocol for encryption of data exchanged between database clients and the database. This includes data in Oracle Net Services (formerly known as Net8), LDAP, thick JDBC, and IIOP format. SSL encryption provides users with an alternative to the native Oracle Net Services encryption protocol which has been supported in Oracle Advanced Security (formerly known as Advanced Networking Option) since Oracle7. A benefit of SSL is that it is a de facto Internet standard, and can be used with clients using protocols other than Oracle Net Services.

In a three-tier system, SSL support in the database means that data exchanged between the middle tier and the database can be encrypted using SSL. The SSL protocol has gained confidence of users, and it is perhaps the most widely-deployed and well-understood encryption protocol in use today. Oracle's implementation of SSL supports the three standard modes of authentication, including anonymous

(Diffie-Hellman), server-only authentication using X.509 certificates, and mutual (client-server) authentication with X.509.

Oracle Application Server also supports SSL encryption between thin clients and the Oracle Application Server, as well as between Oracle Application Server and Oracle Data Server. As in Oracle, anonymous, server-only, and client-server authentication via X.509 are supported.

JDBC Security

JDBC is an industry-standard Java interface that provides a Java standard for connecting to a relational database from a Java program. Sun Microsystems defined the JDBC standard, and Oracle Corporation, as an individual provider, implements and extends the standard with its own JDBC drivers. Oracle implements two types of JDBC drivers: Thick JDBC drivers built on top of the C-based Oracle Net Services client, and thin (pure Java) JDBC drivers to support downloadable applets.

Since thick JDBC uses the full Oracle Net Services communications stack on both client and server, it can take advantage of existing Oracle Advanced Security encryption and authentication mechanisms. Because the thin JDBC driver is designed for use with downloadable applets used over the Internet, Oracle includes a 100% Java implementation of Oracle Advanced Security encryption and integrity algorithms for use with thin clients.

Easy Configuration, No Changes to your Applications

Configuring the network parameters for the server and/or client enables the network encryption/integrity function. Most businesses can therefore easily uptake this technology as there are no changes required in the application.

Strong Authentication Services for Oracle Database 10g

Unauthorized access to information is a very old problem. Business decisions today are driven by information gathered from mining terabytes of data. Protecting sensitive information is key to a business's ability to remain competitive. Access to key data repositories such as the Oracle Database 10g that house valuable information can be granted once users are identified and authenticated accurately. Verifying user identity involves collecting more information than the usual user name and password. Oracle Advanced Security provides the ability for businesses to leverage their existing security infrastructures such as Kerberos, Public Key Infrastructure (PKI), and RADIUS for strong authentication services to the Oracle Database 11g.   Certificate Revocation Lists can be stored in the file system, Oracle Internet Directory or using CRL Distribution Points.

The ability for Oracle Database Servers or Database Clients /Users to use PKI Credentials stored in Smart Cards or other Hardware Storage Modules using industry's PKCS#11 standard. This is especially useful for users as it provides roaming access to the database via client server applications or web applications.

Storing server credentials in a hardware module provides an additional level of security that some deployments require.

Kerberos Authentication

Oracle Advanced Security includes a Kerberos client that is compatible with a Kerberos v5 ticket that is issued by any MIT v5 compliant Kerberos server or Microsoft KDC. Businesses can continue to operate in a heterogeneous environment using Oracle Advanced Security's Kerberos solution. Once an Oracle database is registered with a Kerberos Server and configured to support a Kerberos Service, enterprise users can authenticate to the database without any additional complications. Organizations that are already using a Kerberos Server and Oracle Advanced Security's Kerberos adapter can migrate their external database users to the directory to benefit from centralized user management.

Kerberos Enhancements

Oracle Database 11g Advanced Security Kerberos enhancements include support for principal names up to 2000 characters in length. In addition, Oracle Database 11g Advanced Security provides Kerberos cross realm support allowing Kerberos principals in one realm to authenticate to Kerberos principals in another realm.

Here's an example of creating an externally authenticated Oracle user. The username should correspond to the Kerberos user.

```
SQL> CONNECT / AS SYSDBA;
SQL> CREATE USER "KRBUSER@SOMECO.COM" IDENTIFIED
         EXTERNALLY;
SQL> GRANT CREATE SESSION TO "KRBUSER@SOMECO.COM";
```

Please refer to the Oracle Advanced Security administrator's guide for additional information on configuring the Oracle database and client for Kerberos authentication.

PKI Support

Oracle Advanced Security's SSL client can be used in any PKI that is industry standards compliant and accept standard PKCS7 certificate requests and issue X509v3 certificates. Oracle Advanced Security's provides an Entrust adapter that allows business applications to leverage Entrust's PKI with Oracle Database 11g.

Oracle Wallet Manager continues to be the tool to use for certificate requests and other certificate management tasks for the end user. Additional command line utilities that assist in managing Certificate Revocation Lists (CRLs) and other Oracle Wallet operations are also available in this release.

Certification Revocation Lists published to an LDAP server, a file system or a URL are supported by Oracle's SSL infrastructure.

PKCS#12 Support

Oracle Advanced Security supports X.509 certificates stored in PKCS #12 containers, making the Oracle wallet interoperable with third party applications like Netscape Communicator 4.x and Microsoft Internet Explorer 5.x, and providing wallet portability across operating systems. Users who have existing PKI credentials may export them in PKCS#12 format and reuse them in Oracle Wallet Manager, and vice versa. PKCS#12 thus increases interoperability and reduces the cost of PKI deployment for organizations.

PKCS#11 Support, Smart Cards/Hardware Security Modules

An Oracle Wallet is a software container that holds the private key and other trust points of the certificate. Oracle Advanced Security 10g supports the support PKCS#11 industry standard. This allows the private keys that were previously stored on the file system to be created and stored in secure devices such as Hardware Security Modules or Smart Cards that are available in the market.

PKI Authentication for Oracle Database 10g Enterprise Users

Since Oracle8i, Oracle Advanced Security has supported authentication for directory users to the Oracle database using digital certificates stored in the directory.

Oracle expands PKI integration and interoperability through:

- PKCS#11 support
- Wallet storage in Oracle Internet Directory
- Multiple certificates per wallet
- Strong wallet encryption
- OracleAS Certificate Authority

Wallets Stored in Oracle Internet Directory

Oracle Enterprise Security Manager creates user wallets as part of the user enrollment process. The wallet is stored in Oracle Internet Directory, or other LDAP-compliant directory. Oracle Wallet Manager can upload wallets to—and retrieve them from—the LDAP directory.

Storing the wallet in a centralized LDAP-compliant directory supports user roaming, allowing users to access their credentials from multiple locations or devices, ensuring consistent and reliable user authentication, while providing centralized wallet management throughout the wallet life cycle.

Multiple Certificate Support

Oracle Wallets support multiple certificates per wallet, including:

- S/MIME signing certificate

- S/MIME encryption certificate
- Code-signing certificate

Oracle Wallet Manager Version 3.0 supports multiple certificates for a single digital entity in a persona—with multiple private key pairs in a persona (each private key can match only one certificate). This enables consolidation of and more secure management of users' PKI credentials.

Strong Wallet Encryption

The private keys associated with X.509 certificates require strong encryption, over secure channels. Oracle replaces DES encryption with 3-key triple DES (3DES), which is a substantially stronger encryption algorithm and provides strong security for Oracle wallets.

RADIUS (Remote Dial-in User Service)

Oracle Advanced Security provides a RADIUS client that allows Oracle Database 11g to respect the authentication and authorizations asserted by a RADIUS server. This feature is especially useful for businesses that are interested in two-factor authentication that establishes your identity based on what you know (password or PIN information) and what you have (the token card) provided by some token card manufacturers. RADIUS (RFC #2138) is a distributed system that secures remote access to network services and has long been established as an industry standard for remote and controlled access to networks. RADIUS user credentials and access information are defined in the RADIUS server to enable this external server to perform authentication, authorization and accounting services when requested.

Oracle RADIUS support is an implementation of the RADIUS Client protocols that enables database to provide authentication, authorization and accounting for RADIUS users. It sends authentication requests to RADIUS server and acts upon the server's responses. The authentication can occur either in synchronous or asynchronous authentication modes and is part of Oracle configuration for RADIUS support.

Oracle Advanced Security provides authentication, respects authorizations stored in RADIUS and basic accounting services to RADIUS users when accessing the Oracle database.

SUMMARY

Encryption is a key component of the defense-in-depth principle and Oracle continues to develop innovative solutions to help customers address increasingly stringent security requirements around the safeguarding of PII data. Retailers can use Oracle Advanced Security TDE to address PCI-DSS requirements while university and healthcare organizations can use TDE to safeguard social security numbers and other sensitive information. Encryption plays an especially important role in safeguarding data in transit. Oracle Advanced Security network encryption protects data in transit on the intranet from network sniffing and modification. Oracle Advanced Security TDE protects sensitive data on disk drives and backup media from unauthorized access, helping reduce the impact of lost or stolen media.

Oracle Advanced Security Technical whitepaper
June 2007
Author: Paul Needham
Contributing Authors: Peter Wahl

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com