

The Fully Encrypted Data Center

Encrypting Your Data Center on Oracle's SPARC Servers

ORACLE TECHNICAL WHITE PAPER | JUNE 2016





Table of Contents

Introduction	1
Target Audience and Assumed Knowledge	1
The Role and Relevance of Oracle’s SPARC Processors	1
SPARC M7 Processor—Integrated Cryptographic Acceleration	1
SPARC Processor Cryptographic Operational Model	2
End-to-End Application Security Using SPARC Processors	3
SPARC Cryptography Performance	3
Configuring Security	4
Oracle WebLogic Server Security Acceleration Using Oracle Ucrypto Provider	4
Accelerating SSL Using Oracle Ucrypto Provider	5
Accelerating Web Services Security Using the Oracle Ucrypto Provider	6
Using the Latest Version of SSL/TLS	7
Verifying Hardware-Assisted Security for Oracle Application Server	8
Database Tier Security	9
Oracle Database Security: Applied Scenarios	9
Enabling TDE with Oracle Database 12c	10
Master Key Management Using Oracle Solaris PKCS#11 Softtoken	11
Securing the Master Key Management for Transparent Data Encryption	12
Tablespace and Column Encryption	13
Encrypting and Decrypting Database Backup and Restore	14
Network Data Encryption	15
Securing Data at Rest Using ZFS Encryption	15



Summary	16
References	17



Introduction

This document discusses how to secure applications using Oracle Solaris 11 security and the hardware-assisted cryptography capabilities of Oracle's SPARC servers. This document explores the end-to-end application security scenarios, technical prerequisites, configuration, deployment, and verification guidelines for multitier application deployments running on Oracle Solaris 11–based SPARC servers. In addition, this document covers the Oracle hardware-assisted cryptographic acceleration of the SPARC processor, a key feature when performance and data protection are deemed critical. The derived security benefits can be leveraged into a variety of solutions including application software, middleware, and infrastructure software.

Target Audience and Assumed Knowledge

This document is intended for security practitioners as well as developers and administrators of applications who can benefit from secure communications. Developers and administrators should be familiar with Oracle's SPARC servers, Oracle Solaris 11, Oracle Advanced Security and its Transparent Data Encryption feature, network encryption, Oracle HTTP Server, and application security techniques for secure communication using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols.

The Role and Relevance of Oracle's SPARC Processors

Because security has taken unprecedented importance in all facets of the IT industry, organizations are proactively adopting cryptographic mechanisms to protect their businesses and information from unauthorized access and ensure the confidentiality and integrity of data during transit and in storage. Cryptographic operations are heavily compute-intensive, burdening the host system with additional CPU cycles and network bandwidth and resulting in significant degradation of the overall throughput of the system and its hosted applications. For example, a host server capable of processing 1,000 transactions per second can perform only 10 transactions per second after deploying SSL to secure communications with the hosted application.

To speed up cryptographic performance, security experts often recommend and use cryptographic accelerator appliances to offload cryptographic operations and save CPU cycles, enhancing the system's throughput and its hosted applications. While useful, adopting a specialized appliance for offloading cryptographic operations introduces a new set of costs, complexities, and issues in terms of procurement, additional installation, configuration, testing procedures, management, and support that significantly increases the power demands and costs of deployment projects. Foreseeing the need for special-purpose hardware that can outpace workload demands, Oracle introduced the industry's first and fastest on-chip hardware cryptographic capabilities as part of the Ultra SPARC T1 processor, which was launched during 2005, and then Oracle continued to augment the cryptography support in each new generation of SPARC processors.

SPARC M7 Processor—Integrated Cryptographic Acceleration

With Oracle's new Software in Silicon capabilities coupled with an innovative cache and memory hierarchy, Oracle's SPARC M7 processor delivers dramatically higher processing speed and revolutionary protection against malware and software errors.

The Silicon Secured Memory feature of the SPARC M7 processor provides real-time data integrity checking to guard against pointer-related software errors and malware. It replaces very costly software instrumentation with low-overhead hardware monitoring. Silicon Secured Memory enables applications to identify erroneous or unauthorized memory access, diagnose the cause, and take appropriate recovery actions. The SPARC M7 processor incorporates hardware units that accelerate specific software functions or primitives. The eight on-chip accelerators offload database query processing and perform real-time data decompression. The In-Memory Query Acceleration feature delivers performance that is up to ten times faster compared to other processors. The In-Line Decompression feature allows up to three times more data to be stored in the same memory footprint, without any performance penalty.

The SPARC M7 processor also has cryptographic instruction accelerators integrated directly into each processor core. These accelerators enable high-speed encryption for over a dozen industry-standard ciphers, eliminating the performance and cost barriers typically associated with secure computing.

Table 1 shows the cryptographic algorithms supported by SPARC processors. Compared to the alternative on-chip/on-core implementations of competitive processors, SPARC processors offer a comprehensive set of algorithms supporting a long list of public-key encryption, symmetric-key encryption, and message-digest algorithms.

TABLE 1. CRYPTOGRAPHIC ALGORITHMS SUPPORTED BY SPARC PROCESSORS

Algorithm Type	Algorithm
Accelerator Driver	Userland (no drivers required)
Public-Key Encryption	RSA, DSA, DH, ECC
Bulk Encryption	AES, DES, 3DES, R4, Kasumi, Camelia
Message Digest	CRC32c, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
APIs	PKCS#11 Standard, Ucrypto APIs, Java Cryptography Extensions, OpenSSL

SPARC Processor Cryptographic Operational Model

With SPARC processors, applications can directly access the on-core cryptographic functions, performing those functions in hardware without requiring the use of special configurations or drivers, kernel parameters, and administrative permissions (see Figure 1).

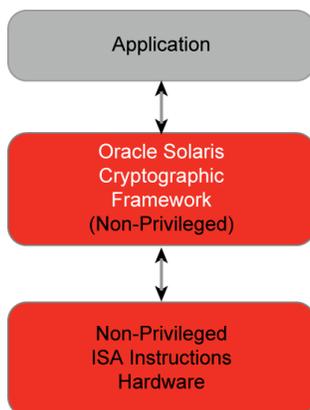


Figure 1. Cryptographic operational model of SPARC processors.

In practice, the Cryptographic Framework feature of Oracle Solaris acts as the core intermediary between applications and the underlying hardware. The framework enables user-level applications to automatically leverage the hardware-assisted cryptographic acceleration functions. The Cryptographic Framework libraries provide a set of cryptographic services and application programming interfaces (APIs) whereby both kernel-level and user-level application consumers can transparently delegate the cryptographic operations to hardware without adding any new code to applications.

End-to-End Application Security Using SPARC Processors

Applications can significantly increase security performance by offloading and delegating their cryptographic operations to the on-core cryptographic capabilities of SPARC processors. These on-core cryptographic acceleration capabilities can be accessed in a variety of ways by application infrastructure components including Oracle HTTP Server, Oracle Application Server, and the Oracle Database server.

A typical deployment scenario for an application infrastructure enabled with an end-to-end security topology (see Figure 2) requires the use of encryption at all levels to ensure secure data in transit, secure data during processing, and secure data in storage. The SPARC processor–based cryptographic acceleration can significantly contribute to the end-to-end security topology when the use of cryptographic mechanisms is deemed critical. The delivery of high-performance security is accomplished through the Oracle Solaris Cryptographic Framework, which enables applications to transparently offload and delegate their cryptographic operations to the on-core cryptographic capabilities of SPARC processors. In addition, with the support of the Cryptographic Framework, applications can leverage Oracle Solaris–facilitated key storage and management features.

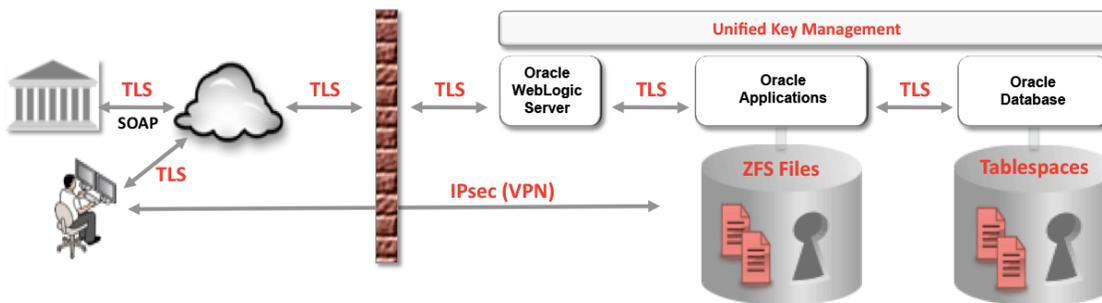


Figure 2. Deployment of a multitier application using an end-to-end security topology with SPARC processors.

SPARC Cryptography Performance

SPARC processor–based cryptographic acceleration eliminates cryptographic overhead when using the on-chip cryptographic acceleration features to improve SSL responsiveness. To evaluate the effectiveness of this technology, a workload can be generated using Oracle Application Testing Suite—a test and performance monitoring suite from Oracle. Tests simulated 1,000 concurrent users interacting with a website supplied from the server. Each user queried the web application using secure SSL communications as many times as possible per minute, clearing caches in between queries. The workload was sustained for 10 minutes to demonstrate a continuous workload rather than a peak performance capability. This load test was not intended to push the upper limits of the server but rather to demonstrate the overhead of cryptography at a reasonable load and the effects of using hardware-assisted cryptographic acceleration.

As shown in Figure 3, the results show minimal difference in CPU utilization due to SSL overhead between the completely unsecured application versus full end-to-end SSL encryption utilizing the on-chip cryptographic acceleration. Turning on SPARC encryption yields tangible, immediate, and cost-efficient results in the form of faster, secure transactions and fast response times—all without adding any additional security equipment costs, performance penalties, or changes in power usage profiles, and without elaborate system configurations.

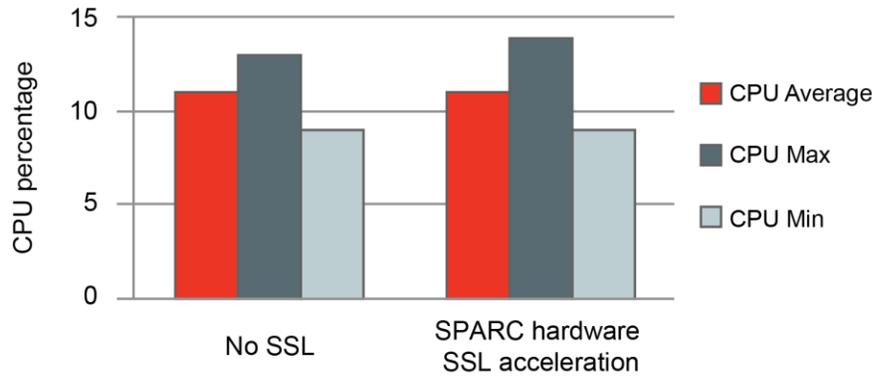


Figure 3. SPARC-based accelerated encryption results in only a minor CPU load difference from a system with no SSL encryption whatsoever, leaving CPU resources available for running the application.

Configuring Security

The following sections provide information on configuring security for applications running on Oracle Solaris 11 and SPARC servers.

Oracle WebLogic Server Security Acceleration Using Oracle Ucrypto Provider

By default, when deployed on SPARC servers, the Oracle WebLogic Server relies on the Java Development Kit (JDK) and its Oracle Ucrypto provider environment for handling cryptographic operations. The Oracle Ucrypto provider in Java Cryptography Extension (JCE) contains a dedicated integration that leverages Oracle Solaris 11 Ucrypto APIs for offloading and delegating cryptographic operations supported by Oracle's SPARC-based on-core cryptographic instructions. In addition, the Oracle WebLogic Server SSL must be configured to use the Java Secure Socket Extension (JSSE) provider as the default SSL provider. The JSSE provider uses the underlying JCE provider exclusively for all of its cryptographic operations and, hence, the Oracle WebLogic Server SSL configuration is able to automatically take advantage of hardware-assisted cryptographic acceleration capabilities through the Oracle Ucrypto provider.

To leverage the Oracle Ucrypto provider JDK 7 update 4 or later must be installed and used as the Java runtime environment (JRE) on Oracle Solaris 11. After installation, make sure that the Oracle Ucrypto provider is identified as the default provider in the Java security properties file `java.security` located in the `$JAVA_HOME/jre/lib/security/` directory.

```
security.provider.1=com.oracle.security.ucrypto.UcryptoProvider
    ${java.home}/lib/security/ucrypto-solaris.cfg
```

With the release of JDK 7 update 4, Oracle introduced the Oracle Ucrypto provider, which provides a specialized interface that bypasses PKCS#11 and automatically leverages the hardware-assisted cryptographic acceleration capabilities of Oracle's SPARC T4 (or newer) processors. In a typical JDK 7 installation (JDK 7 update 4 or later) on Oracle Solaris 11 and SPARC servers, the JRE is preconfigured to make use of the Oracle Ucrypto provider by default. This enables the Java and Oracle WebLogic Server–hosted applications and XML web services to automatically delegate their cryptographic-intensive operations processed via the Cryptographic Framework using Oracle's SPARC on-core cryptographic instructions (see Figure 4).

In addition to the Oracle Ucrypto provider, the JDK provides a PKCS#11 provider implementation (SunPKCS11) that enables Java applications to access PKCS#11-based cryptographic provider implementations provided by the Oracle Solaris Cryptographic Framework.



Figure 4. Oracle WebLogic Server security using Oracle's SPARC servers.

Accelerating SSL Using Oracle Ucrypto Provider

The following steps explain how to configure Oracle WebLogic Server for SSL acceleration using the on-chip cryptographic acceleration capabilities of Oracle's SPARC processor–based servers.

- » Configure Oracle WebLogic Server to listen for SSL. Before configuration, obtain the necessary private keys, the server certificate (including the public key), and the trust Certificate Authority (CA) certificates from a CA, and then store them in the Java keystore (identity and trust keystores) configured within the Oracle WebLogic Server environment. For development and testing, users may choose to use a self-signed certificate, private key, and trusted CA certificate created using the Java `keytool` utility. Use the Oracle WebLogic Server Administration Console to configure the identity and trust keystores. Follow the SSL configuration guidelines specified in the [Oracle Fusion Middleware—Securing Oracle WebLogic Server 12c Release 1 \(12.1.1\)](#) documentation.
- » Check that Oracle WebLogic Server is listening and responds over the SSL port. This can be verified using the Oracle WebLogic Server console logs for the managed server.
- » Confirm that the SSL configuration relies on the JSSE provider to enable automatic delegation of SSL-based cryptography to automatically take advantage of the Oracle Ucrypto provider. This can be accomplished by adding the Java runtime option `-Dweblogic.ssl.JSSEEnabled=true` in the JRE settings of the managed server instance of Oracle WebLogic Server.
- » To enforce the hardware-facilitated cryptographic capabilities, it is critical that the Oracle WebLogic Server SSL provider configuration defines SSL cipher suites that include cryptographic algorithms supported by the hardware. This configuration also helps to disable weak SSL cipher suites. This can be accomplished by editing the Oracle WebLogic Server domain's `config.xml` file:

```

<ssl>
  <enabled>true</enabled>
  <ciphersuite>TLS_RSA_WITH_AES_128_CBC_SHA</ciphersuite>
  <ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
  . . .
</ssl>

```

- » Restart the managed server instance of Oracle WebLogic Server. This can be accomplished using the Oracle WebLogic Server console.

Accelerating Web Services Security Using the Oracle Ucrypto Provider

Web Services Security (WS-Security) plays a critical role in providing message-level security for XML web services by ensuring confidentiality, integrity, and access control for SOAP messages. Oracle WebLogic Server relies on the JCE provider for supporting cryptographic operations involved with message-level security of XML web services and the Oracle Ucrypto provider for facilitating cryptographic acceleration associated with encryption, signature, and message digest operations.

Oracle WebLogic Server strongly recommends the use of the WS-SecurityPolicy standard, and it provides predefined WS-SecurityPolicy files to specify message-level security requirements. The WS-SecurityPolicy standard describes the message-level security requirements of SOAP messages and how the requested operation should be digitally signed or encrypted using relevant cryptographic algorithm suites defined in the policy. The exposed web service makes the associated security policy available via the Web Services Definition Language (WSDL).

The following steps explain how to configure Oracle WebLogic Server for XML web services security operations and acceleration using the on-chip cryptographic acceleration capabilities of SPARC servers. It is assumed that the web service created is deployed as a JSON Web Signature (JWS) file that is implemented using Oracle WebLogic Server web service and/or Java API for XML Web Services (JAX-WS) APIs.

Update the JWS file, including the Oracle WebLogic Server-specific `@Policy` and `@Policies` JWS annotations, to specify the predefined policy files that represent WS-Policy and WS-SecurityPolicy definitions for performing the required WS-Security mechanisms.

```

@WebService(name="Simple", targetNamespace="http://oracle.org")
@WLHttpTransport(contextPath="/wsspl2/wss10",
serviceUri="UsernameTokenPlainX509SignAndEncrypt")
@Policy(uri="policy:Wsspl.2-2007-Wss1.0-UsernameToken-Plain-X509-Basic256.xml")
public class UsernameTokenPlainX509SignAndEncrypt {
  @WebMethod
  @Policies({
    @Policy(uri="policy:Wsspl.2-2007-SignBody.xml"),
    @Policy(uri="policy:Wsspl.2-2007-EncryptBody.xml")})
  public String echo(String s)
  {
    return s;
  }
}

```

The above WS-Policy annotation identifies the WS-SecurityPolicy, specifying that the service authenticates the client using a username token and that both the request and response messages are signed and encrypted with X.509 certificates.



After including the policies, recompile and deploy the web service. For the web services configuration and deployment steps, refer to the [Oracle Fusion Middleware—Securing Oracle WebLogic Server 12c Release 1 \(12.1.1\)](#) documentation.

It is also critical that the client application that invokes a deployed web service be associated to a client-side security policy file. Typically, the security policy files are the same as those configured for the server-side web service invoked. But because the server-side files are not exposed to the client Java runtime, the client application must load its own local copies. Users may also choose to use the `weblogic.jws.jaxws.ClientPolicyFeature` class in the client application to override the effective policy defined for a service.

Make sure the cryptographic mechanisms specified in the security policy file identify the WS-SecurityPolicy algorithm suite supported by the Oracle Ucrypto provider or the SunPKCS11 provider. Refer to the [Java PKCS#11 Reference Guide](#) for the supported list of cryptographic algorithms. If the specified algorithm suite is Basic256, it represents the AES-256 algorithm for bulk encryption, the Sha256 algorithm to represent SHA-256–based message digests, and Rsa-oaep-mgf1p to represent RSA for key wrap. The Oracle Ucrypto provider supports both the AES-256 and SHA256withRSA algorithms, and it leverages the use of Oracle’s SPARC hardware-assisted cryptographic acceleration.

Update the Java client application to make sure it loads the client-side policy files, and rebuild/redeploy the client application. If the client is a web application, restart the managed server instances of Oracle WebLogic Server where it is deployed.

Using the Latest Version of SSL/TLS

There are several ways to take advantage of SPARC cryptographic hardware acceleration features, but not all are recommended. Many older papers refer to the Kernel SSL (KSSL) proxy approach. KSSL essentially acts as a two-way proxy for intercepting SSL workloads and executing the encryption and decryption using the cryptographic capabilities of the SPARC processor, rather than placing that burden on the overall system. KSSL was introduced in the early days of Oracle’s SPARC T2, T3, and T4 processors as a means to quickly take advantage of cryptographic acceleration for applications. However, its use is no longer recommended because more-secure and higher-performance options exist to take advantage of the features of Oracle’s newer SPARC T5, M6, and M7 processors.

In many cases, the use of low-grade SSL (versions 2.0 and 3.0) is also now discouraged as a web encryption mechanism due to recent discoveries regarding its susceptibility to hacking (see CVE-2014-2566). The current recommendation is to disable SSL 2.0 and 3.0 and instead use TLS version 1.1 or 1.2. The use of TLS 1.1 and 1.2 requires JDK 7 update 1 (or later) and that Java Secure Socket Extensions (JSSE) be enabled.

In Oracle WebLogic Server versions 10.3.6 and 12c, JSSE and JDK 7 are certified, and using Java startup options enables TLS functionality and allows the minimum acceptable version of the TLS protocol to be specified. The required `JAVA_OPTIONS` parameter can be configured via environment variables for the Oracle Solaris shell where the Oracle WebLogic Server startup scripts will be run.

```
export JAVA_OPTIONS=-Dweblogic.security.SSL.protocolVersion=TLS1 \  
-Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.1
```

Oracle HTTP Server should also be configured to properly take advantage of TLS. This can be achieved by ensuring that Oracle HTTP Server version 12.1.3 is the minimum version in use and that the master copy of the `ssl.conf` file (located at `DOMAIN_HOME/config/fmwconfig/components/OHS/componentName`) for Oracle HTTP Server contains the following line:

```
SSLProtocol nzos_Version_1_1 nzos_Version_1_2
```

Once the TLS version is specified, as shown above, the hardware accelerators will automatically be used. For more information on enabling TLS security on Oracle WebLogic Server, refer to My Oracle Support document number 1936300.1).

Verifying Hardware-Assisted Security for Oracle Application Server

To ensure hardware-assisted cryptographic acceleration is configured for use and is working with the security scenarios, it is recommended that the following Oracle Solaris DTrace script be used. DTrace is a feature of Oracle Solaris.

```
#!/usr/sbin/dtrace -s
pid$1:libsoftcrypto:yf*:entry,
pid$1:libmd:yf*:entry
{
    @[probefunc] = count();
}
tick-10sec
{
    printa(@);
    clear(@);
    trunc(@,0);
}
tick-100sec
{exit(0);}
```

Save the script as the file `cryptoverify.d` and then run the script (including the Oracle Containers for J2EE server's Java process ID as a command-line argument).

```
# dtrace -s cryptoverify.d <Server Process ID>
```

For example, in an SSL/TLS encryption scenario using the `TLS_RSA_WITH_AES_128_CBC_SHA` cipher suite, a positive and growing value of AES jobs indicates that cryptographic acceleration is operational on the target AES bulk encryption payloads. Refer to the following sample output.

```
# dtrace -s cryptoverify.d 5774

dtrace: script 'crypto-t4.d' matched 51 probes
CPU   ID           FUNCTION:NAME
65 83719         :tick-10sec
yf_aes128_ecb_decrypt           39922
yf_aes128_load_keys_for_decrypt 39922
```

```

65 83719          :tick-10sec
yf_aes128_ecb_decrypt          44108
yf_aes128_load_keys_for_decrypt 44108
65 83719          :tick-10sec
yf_aes128_ecb_decrypt          44534
yf_aes128_load_keys_for_decrypt 44534
..

```

Database Tier Security

Enterprise applications rely on Oracle Database security for ensuring the confidentiality and integrity of data in transit and at rest by using encryption at all levels. Transparent Data Encryption (TDE), a feature of Oracle Advanced Security, encrypts and decrypts data stored in the database and in transit, providing support for all operations related to network communication, tablespace and column-level encryption, and encrypted backups. Since Oracle Database 11g (release 11.2.0.3), TDE extended support for hardware-assisted cryptographic acceleration using SPARC processors and Oracle Solaris 11 to support offloading cryptographic processing associated with tablespace encryption and master key-based operations (see Figure 5).

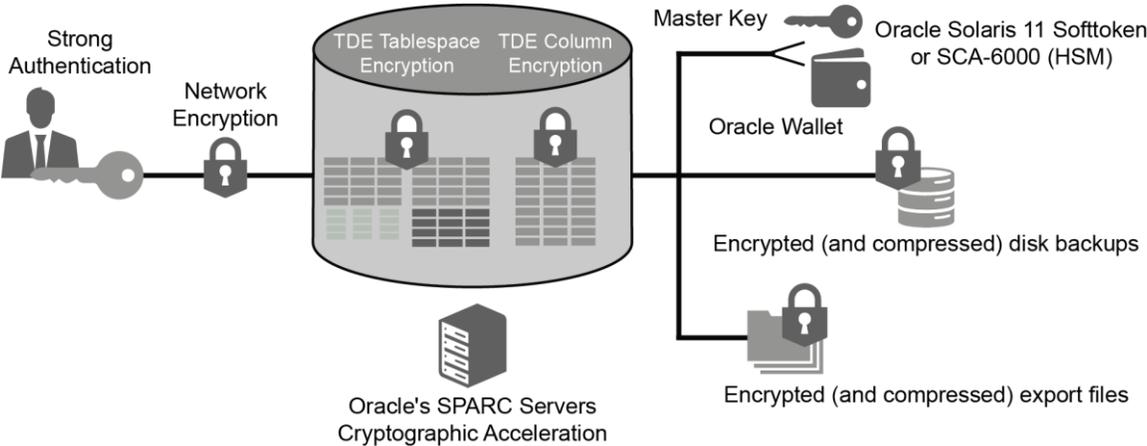


Figure 5. Oracle Database security: applied security scenarios.

Oracle Database Security: Applied Scenarios

TDE has been tested and verified to use SPARC processor hardware-assisted cryptographic acceleration for most encryption operations. The applied security scenarios are as follows:

- » Enabling TDE with Oracle Database 12c
- » Master key management using Oracle Solaris PKCS#11 Softtoken
- » Master key backup and recovery
- » Tablespace and column encryption
- » Encryption and decryption of database backup and restore operations
- » Network data encryption

Enabling TDE with Oracle Database 12c

The following section describes how to configure TDE with Oracle Database 12c using a software keystore. For more detailed information, see [Database Advanced Security Guide](#) in the Oracle Database Online Documentation 12c Release 1 (12.1).

1. Create a wallet directory.

```
% mkdir /export/home/oracle/wallets/<keystore_location>
```

2. Set the software keystore location in the `sqlnet.ora` file. By default, this file is located in the `$ORACLE_HOME/network/admin/directory`.

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE=
(METHOD=FILE)
(METHOD_DATA=
(DIRECTORY=/export/home/oracle/wallets/<keystore_location>)))
```

3. Create the keystore. Log in to the database instance as a user who has been granted the `ADMINISTER KEY MANAGEMENT` or `SYSKM` privilege. Then run the `ADMINISTER KEY MANAGEMENT SQL` statement to create the keystore. This example creates a password-based software keystore.

```
$ sqlplus c##sec_admin as syskm
Enter password: password
Connected.

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location'
IDENTIFIED BY software_keystore_password;

keystore altered.
```

4. You can confirm the process was successful by looking at the status of the encryption wallet. As this example shows, the wallet is open but does not yet have a master key to use.

```
SQL> select STATUS from V$ENCRYPTION_WALLET;

STATUS
-----
OPEN_NO_MASTER_KEY

SQL> select WALLET_TYPE from V$ENCRYPTION_WALLET;

WALLET_TYPE
-----
PASSWORD
```

5. Close the keystore, and then reopen it.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE;

keystore altered.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
      IDENTIFIED BY software_keystore_password;
```

```
keystore altered.
```

6. Set the software TDE master encryption key and create a backup of the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY software_keystore_password
      WITH BACKUP USING 'backup_key';
```

```
keystore altered.
```

7. You can confirm the process was successful by looking at the status of the encryption wallet.

```
SQL> select STATUS from V$ENCRYPTION_WALLET;
```

```
STATUS
```

```
-----
```

```
OPEN
```

```
SQL exit
```

8. Now, you can create an encrypted tablespace using a desired cipher. In this example, the `ENCRYPTION USING 'AES256'` statement specifies the encryption algorithm and the key length for the encryption.

```
$ sqlplus "/as sysdba" << !
CREATE BIGFILE TABLESPACE O_cust_space
      DATAFILE '$DB_DIR/O_cust' SIZE 100000M REUSE
      ENCRYPTION USING 'AES256'
      DEFAULT STORAGE (ENCRYPT);
!
```

9. Note: If you shut down the database, you will need to open the wallet after startup.

```
% sqlplus "/as sysdba" << !
      startup
      ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
      " software_keystore_password "
      exit;
!
```

Master Key Management Using Oracle Solaris PKCS#11 Softtoken

Oracle Wallet Manager is a centralized keystore for securing the master key used for encryption and decryption. Starting with Oracle Database 11g, the database supports the use of a PKCS#11-based hardware security module (HSM) keystore as an Oracle wallet. Using an Oracle Solaris PKCS#11 softtoken-based Oracle wallet secures the master key from duplication and copying during database and file system backups. This can be done using the Oracle Solaris PKCS#11 softtoken referred to as “Sun Software PKCS#11 Softtoken.”

1. Configure a Sun Software PKCS#11 Softtoken keystore using the Oracle Solaris Cryptographic Framework `pktool` utility. Set the PIN/passphrase for accessing the softtoken keystore.

```
# pktool setpin keystore=pkcs11
Create new passphrase:
Re-enter new passphrase:
```

2. Enable the Sun Software PKCS#11 Softtoken token as `metaslot`:

```
# cryptoadm enable metaslot
token="Sun Software PKCS#11 Softtoken"
```

3. Copy the PKCS#11 library to the Oracle-suggested directory structure.

» In an Oracle Solaris SPARC environment, first create a directory for the PKCS#11 library:

```
# mkdir -p /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

» Then copy the Oracle Solaris `libpkcs11.so` file to the PKCS#11 library directory:

```
# cp /usr/lib/sparcv9/libpkcs11.so /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

4. Make sure the user and group (`oracle:install`) are set correctly for the PKCS#11 library directory, and make sure the directory is assigned with read and write privileges.

```
# chown -R oracle:oinstall <directory>
```

» In an Oracle Solaris environment, you may choose to set a `SOFTTOKEN_DIR` environment variable (in the Oracle default user shell).

```
# export SOFTTOKEN_DIR=/export/home/oracle/.sunw
```

Securing the Master Key Management for Transparent Data Encryption

To configure TDE to use the Oracle Solaris PKCS#11 softtoken, initially you must set up an HSM-based Oracle wallet identifying the source of the master key as HSM.

1. Edit the `$TNS_ADMIN/sqlnet.ora` file and add an `ENCRYPTION_WALLET_LOCATION` parameter.

```
ENCRYPTION_WALLET_LOCATION =
    (SOURCE=(METHOD=HSM) (METHOD_DATA=
    (DIRECTORY = /export/home/oracle/11g/network/admin/)))
```

2. Log in to SQLPlus as `system` or `sysdba` and create an HSM wallet.

```
$ sqlplus "/ as sysdba"

SQL> alter system set encryption key
    identified by "HSM Username:Password";
```

Note: `Username:Password` are the credentials of the dedicated user account for TDE for support performing master key management operations with the Oracle Solaris PKCS#11 softtoken keystore.

If the database was previously using an Oracle software wallet, users can migrate the master key to the configured Oracle Solaris PKCS#11 softtoken. The migration process automatically decrypts existing data objects and re-encrypts them using the newly created master encryption key on the Oracle Solaris PKCS#11 softtoken.

If TDE was previously configured using a “software wallet,” the master key must be migrated from the software wallet to HSM by adding the clause `MIGRATE USING "software_wallet_password"` to the preceding `sqlplus` command. The `software_wallet_password` is the original password for the software wallet.

```
SQL> alter system set encryption key identified by
      "HSM Password" migrate using "software_wallet_password";
```

Tablespace and Column Encryption

Oracle Database 11.2.0.3 introduced support for Oracle’s SPARC T4 and higher hardware-assisted cryptographic acceleration for TDE. Because the installation process automatically identifies the processor of the host machine, technically no setup is required for the use of hardware-accelerated cryptography on SPARC servers.

Once it is deployed, Oracle Database will use hardware-accelerated cryptography for both the encryption and decryption operations involved with tablespace encryption, network encryption, encrypted backups and restore files, and encrypted dump files.

To test and verify that TDE is using the master encryption key stored in the Oracle Solaris PKCS#11 softtoken, use the following recommended SQL examples. These examples demonstrate TDE operations that rely on the Oracle Solaris PKCS#11 softtoken resident master key.

1. Make sure the database is up and running. Log in and connect to `sqlplus` as `system`.

```
$ sqlplus "/ as sysdba"
SQL> startup;
SQL> connect as system/password;
```

2. Verify opening and closing the Oracle Solaris PKCS#11 softtoken-based HSM wallet. Make sure users use the username and password created for accessing TDE.

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "tdepassword";
System altered.

SQL> select WRL_TYPE, STATUS from v$encryption_wallet;
WRL_TYPE          STATUS
-----
HSM                OPEN

SQL> ALTER SYSTEM SET WALLET CLOSE IDENTIFIED BY "tdepassword";
System altered.
```

3. Create an encrypted tablespace using the HSM wallet.
 - » Make sure the HSM wallet is open.

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

» Now, create the encrypted tablespace.

```
SQL> CREATE TABLESPACE SCASecuredTablespace
 2 DATAFILE '/export/home/oracle/11g/oradata/scasecuretbls1.dbf'
 3 SIZE 50M
 4 ENCRYPTION
 5 DEFAULT STORAGE (ENCRYPT);
```

4. Create a table on the encrypted tablespace, which automatically encrypts all data objects stored.

```
SQL> CREATE TABLE PERSON
 2 (first_name VARCHAR2(11),
 3 last_name VARCHAR2(10),
 4 social_security_number NUMBER(9),
 5 address VARCHAR2(25),
 6 city VARCHAR2(25),
 7 state VARCHAR2(2)) TABLESPACE SecuredTablespace;
```

Encrypting and Decrypting Database Backup and Restore

Encrypted export/import files that use the Oracle Database Data Pump utility can use an HSM-resident master key for encrypting and decrypting dump files.

» By default (without specifying encryption) all export dump files are stored in an unencrypted file. In the following example, the export file is dumped without encryption.

```
$ expdp system/oracle@sid tables=employee
```

» To enforce export dump file encryption using a master key, first make sure the HSM wallet remains open. Users need to use `ENCRYPTION_MODE=TRANSPARENT` to enable encryption of the dump file using the master key stored in the HSM wallet. Specifying the option `ENCRYPTION_MODE=DUAL` encrypts the dump set using the master key stored in the wallet and additionally using the password for encryption.

```
$ expdp system/oracle@sid tables=employee encryption=all
encryption_password=pwd4encrypt encryption_algorithm=AES256 encryption_mode=DUAL
```

» To import the dump file encrypted using the master key, make sure the HSM wallet remains open and set the option for specifying the password used for encryption. Here is an example :

```
$ impdp system/oracle@Ssid encryption_password=pwd4encrypt tables=employee
table_exists_action=replace
```

A backup and restore of the database using Oracle Recovery Manager (Oracle RMAN) can use an HSM-resident master key.

» Make sure the HSM wallet is open before performing the backup, restore, or recover database commands, and also ensure the database is in `archive log` mode.

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
Database mounted.
```

```
SQL> alter database archivelog;
Database altered.
SQL> alter database open;
Database altered.
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "oracle:password";
System altered.
SQL> exit
```

» Use the `rman` utility to turn encryption on before executing the `backup` command.

```
RMAN> connect target sysoper/oracle;
connected to target database: sid (DBID=1555558107)

RMAN> set encryption on;

RMAN> backup as compressed backupset database;
```

Network Data Encryption

Network data encryption enables encryption of data in transit over the network between the Oracle Database server and the Oracle Database clients. Oracle Database supports the use of an Oracle Solaris PKCS#11 keystore that leverages the SPARC processor's hardware-assisted cryptographic acceleration for performing select cryptographic operations. To enable this support, the Oracle wallet must be configured to wallet type PKCS#11, which allows the use of an Oracle Solaris PKCS#11 softtoken instead of a file system-based wallet. The Oracle Wallet Manager application enables configuring the PKCS#11-based wallet to support storing and managing PKI certificate credentials including private keys, certificates, and trusted certificates needed by the SSL and TLS protocols for securing communication and client/server authentication.

As a prerequisite, you must configure a "Sun Software PKCS#11 Softtoken" keystore. Perform Step 1 through Step 3 in the previous section "Master Key Management Using Oracle Solaris PKCS#11 Softtoken" Once the keystore has been configured, use the Oracle Wallet Manager utility to set up the PKCS#11-based Oracle wallet. To configure SSL/TLS, refer to the steps described in the section "[Configuring Secure Sockets Layer Authentication](#)" in the *Oracle Database Advanced Security Administrator's Guide*. It is critical that the server choose SSL cipher suites, including algorithms supported by Oracle's SPARC T4 and SPARC T3 processor. For example, `SSL_RSA_WITH_AES_128_CBC_SHA` is supported because it uses RSA (handshake and authentication) and AES-128 for bulk encryption. The Oracle default SSL cipher suite `SSL_RSA_WITH_RC4_128_SHA` requires the use of RC4 for bulk encryption, which is not supported.

Securing Data at Rest Using ZFS Encryption

By default, ZFS uses the Oracle Solaris 11 cryptographic services APIs, which automatically benefit from the hardware acceleration of the AES algorithm available on SPARC processors. The policy for encryption is set at the data set level when data sets (file systems or ZFS volumes) are created. Each ZFS disk block (smallest size is 512 bytes, largest is 128 k) is encrypted using the AES algorithm in either CCM or GCM mode. The wrapping keys need to be provided by the Oracle Solaris administrator who creates the file system. These keys can be changed at any time without taking the file system offline. The data encryption keys are randomly generated at data set-creation time. The easiest way to create the wrapping keys is to use the existing Oracle Solaris `pktool` command.

```
$ pktool genkey keystore=file keytype=aes keylen=128
outkey=/export/home/user/mykey
```

Using ZFS encryption support can be as easy as this:

```
# zfs create -o encryption=on -o keysource=raw,file:///export/home/user/mykey
myfilesystem/cryptofs
```

Alternatively, for ensuring secure storage and retrieval of wrapping keys, it is recommended that the Oracle Solaris PKCS#11 softtoken be used as the keystore for storing wrapping keys. Using an Oracle Solaris PKCS#11 softtoken as the keystore ensures that the wrapping key is encrypted in storage and the keystore is protected by a PIN. The steps for creating and storing the wrapping key in an Oracle Solaris PKCS#11 softtoken keystore and using the key to create an encrypted ZFS data set are as follows.

```
# pktool genkey keystore=pkcs11 keytype=aes keylen=128 label=mykey
Enter PIN for Sun Software PKCS#11 softtoken:

# zfs create -o encryption=on
-o keysource=raw,pkcs11:object=mykey myfilesystem/cryptofs
Enter PKCS#11 token PIN for 'myfilesystem/cryptofs'
```

In the example above, an AES key is created in the default softtoken keystore for the user. This keystore requires authentication in order to create and use the keys stored in it, so the user is prompted for the keystore PIN (it is really a passphrase, but PKCS#11 terminology uses the word *PIN* for legacy reasons). The syntax of the PKCS#11 URI that is used with the `keysource` property allows for specifying a path to the PIN file. Using this method ensures that the actual wrapping key is encrypted and protected in the PKCS#11 keystore.

For more details on ZFS encryption, refer to the [Oracle Solaris ZFS Administration Guide](#).

Summary

This white paper presented various strategies for securing applications using Oracle Solaris 11 security and the hardware-assisted cryptographic acceleration features of Oracle's SPARC processors. The paper unveiled the core mechanisms, configuration, and deployment strategies, as well as the role and relevance of using Oracle Solaris Cryptographic Framework and Java Cryptography Extension–based techniques for delivering a high-performance, end-to-end security solution. Adopting SSL/TLS encryption for data in transit and encrypted data at rest has become critical for delivering end-to-end security for multitier business applications and to meet regulatory compliance mandates.

The use of the hardware-assisted cryptographic acceleration of SPARC processors for end-to-end security deployments has yielded tangible, immediate, and cost-efficient results in the form of faster secure transactions and better response times—all without adding any additional security equipment costs, changes in power usage profiles, or elaborate system configurations. The derived performance characteristics also clarify the massive burden that unaccelerated cryptographic workloads can have on a server.

To summarize, Oracle's SPARC processor–based servers provide high-performance enterprise security with consistent scalability for applications, while also delivering reductions in space, power consumption, and cost.

References

For more information, see the references in Table 2.

TABLE 2. REFERENCES FOR MORE INFORMATION

Websites	
Oracle Optimized Solutions	oracle.com/optimizedsolutions
Oracle SuperCluster	oracle.com/supercluster
Oracle's SPARC servers	oracle.com/goto/tseries
Oracle Solaris	oracle.com/solaris
Oracle Solaris Cluster	oracle.com/us/products/servers-storage/solaris/cluster/overview/index.html
Oracle ZFS Storage Appliance	oracle.com/us/products/servers-storage/storage/unified-storage
Oracle's Exadata Storage Expansion Racks	oracle.com/us/products/database/exadata/expansion-storage-rack/overview/index.html
Oracle E-Business Suite	oracle.com/us/products/applications/ebusiness/overview/index.html
Oracle Optimized Solution for Secure Backup and Recovery	oracle.com/solutions/optimized-solutions/backup-and-recovery
Oracle Optimized Solution for Secure Disaster Recovery	oracle.com/solutions/optimized-solutions/disaster-recovery
Oracle Technology Network	oracle.com/technetwork/index.html
Oracle Consulting	oracle.com/us/products/consulting/overview/index.html
Oracle SuperCluster White Papers	
"A Technical Overview of Oracle SuperCluster"	oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/o13-045-sc-t5-8-arch-1982476.pdf
"Oracle SuperCluster T5-8: Servers, Storage, Networking, and Software—Optimized and Ready to Run"	oracle.com/us/products/servers-storage/servers/sparc/supercluster/supercluster-t5-8/ssc-t5-8-wp-1964621.pdf
Oracle's Exadata Database Machine White Paper	
"E-Business Suite on Exadata: Oracle Maximum Availability Architecture White Paper"	oracle.com/technetwork/database/features/availability/maa-eps-exadata-197298.pdf
Oracle Solaris White Paper	
"Oracle Solaris and Oracle SPARC T4 Servers—Engineered Together for Enterprise Cloud Deployments"	oracle.com/us/products/servers-storage/solaris/solaris-and-sparc-t4-497273.pdf
Oracle Database White Paper	
"Oracle Database 11g Release 2 High Availability"	oracle.com/technetwork/database/features/availability/twp-databaseha-11gr2-1-132255.pdf
Backup, Recovery, High Availability, and Disaster Recovery White Papers	
"Oracle Optimized Solution for Secure Backup and Recovery: Oracle SuperCluster Backup and Recovery"	oracle.com/technetwork/server-storage/hardware-solutions/oos-ocs-backup-recovery-1973464.pdf

"Oracle Data Guard 11g Data Protection and Availability for Oracle Database"	oracle.com/technetwork/database/features/availability/twp-dataguard-11gr2-1-131981.pdf
--	--

Oracle Optimized Solution Support Document

Oracle Support Document 1558827.1, "Oracle Optimized Solution for Oracle E-Business Suite"	support.oracle.com/epmos/faces/DocumentDisplay?id=1558827.1
--	--

Security Resources

<i>Oracle E-Business Suite System Administrator's Guide – Security (Release 12.1)</i>	docs.oracle.com/cd/E18727_01/doc.121/e12843/T156458T156461.htm
<i>Oracle E-Business Suite Security Guide (Release 12.2)</i>	docs.oracle.com/cd/E26401_01/doc.122/e22952/T156458T156461.htm
<i>Oracle Database Security Guide 11g Release 2 (11.2)</i>	docs.oracle.com/cd/E11882_01/network.112/e36292/title.htm
My Oracle Support Document 946372.1 – Secure Configuration of Oracle E-Business Suite Profiles	support.oracle.com/epmos/faces/DocumentDisplay?id=946372.1
<i>Oracle Database Security Guide 12c Release 1 (12.1)</i>	docs.oracle.com/database/121/DBSEG/
<i>Oracle Solaris 11 Security Guidelines</i>	docs.oracle.com/cd/E26502_01/pdf/E29014.pdf
<i>Architecture Matters: Increasing Oracle Database Security Without Application Performance Loss</i>	community.oracle.com/docs/DOC-999059



Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

-  blogs.oracle.com/oracle
-  facebook.com/oracle
-  twitter.com/oracle
-  oracle.com

Integrated Cloud Applications & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0615

The Fully Encrypted Data Center
June 2016