Oracle® Solaris 11.2의 암호화 및 인증서 관리

부품 번호: E53960-02
2014년 9월

ORACLE®

목차

이 설명서 사용

*Oracle® Solaris 11.2*의 암호화 및 인증서 중앙 관리에서는 암호화 관리 및 사용 방법, 개인/공개 키 인증서 생성 및 관리 방법에 대해 설명합니다.

- **개요** - 암호화 프레임워크 및 키 관리 프레임워크 위주의 개념과 이 기술을 사용하여 파일을 보호하기 위한 작업에 대해 설명합니다.
- **대상** - 기업에서 보안을 구현해야 하는 시스템 관리자
- **필요한 지식** - 보안 개념과 용어에 익숙해야 합니다.

제품 설명서 라이브러리

이 제품에 대한 최신 정보 및 알려진 문제는 설명서 라이브러리(http://www.oracle.com/pls/topic/lookup?ctx=E56343)에서 확인할 수 있습니다.

Oracle 지원 액세스

Oracle 고객은 My Oracle Support를 통해 온라인 지원에 액세스할 수 있습니다. 자세한 내용은 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info를 참조하거나, 청각 장애가 있는 경우 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs를 방문하십시오.

피드백

http://www.oracle.com/goto/docfeedback에서 이 설명서에 대한 피드백을 보낼 수 있습니다.

♦ ♦ ♦ 1 장

1 암호화 프레임워크

이 장에서는 Oracle Solaris의 암호화 프레임워크 기능에 대해 설명하고 다음 항목을 다룹니다.

암호화 프레임워크를 관리하고 사용하려면 3장. 암호화 프레임워크를 참조하십시오.

Oracle Solaris 11.2를 위한 암호화의 새로운 기능

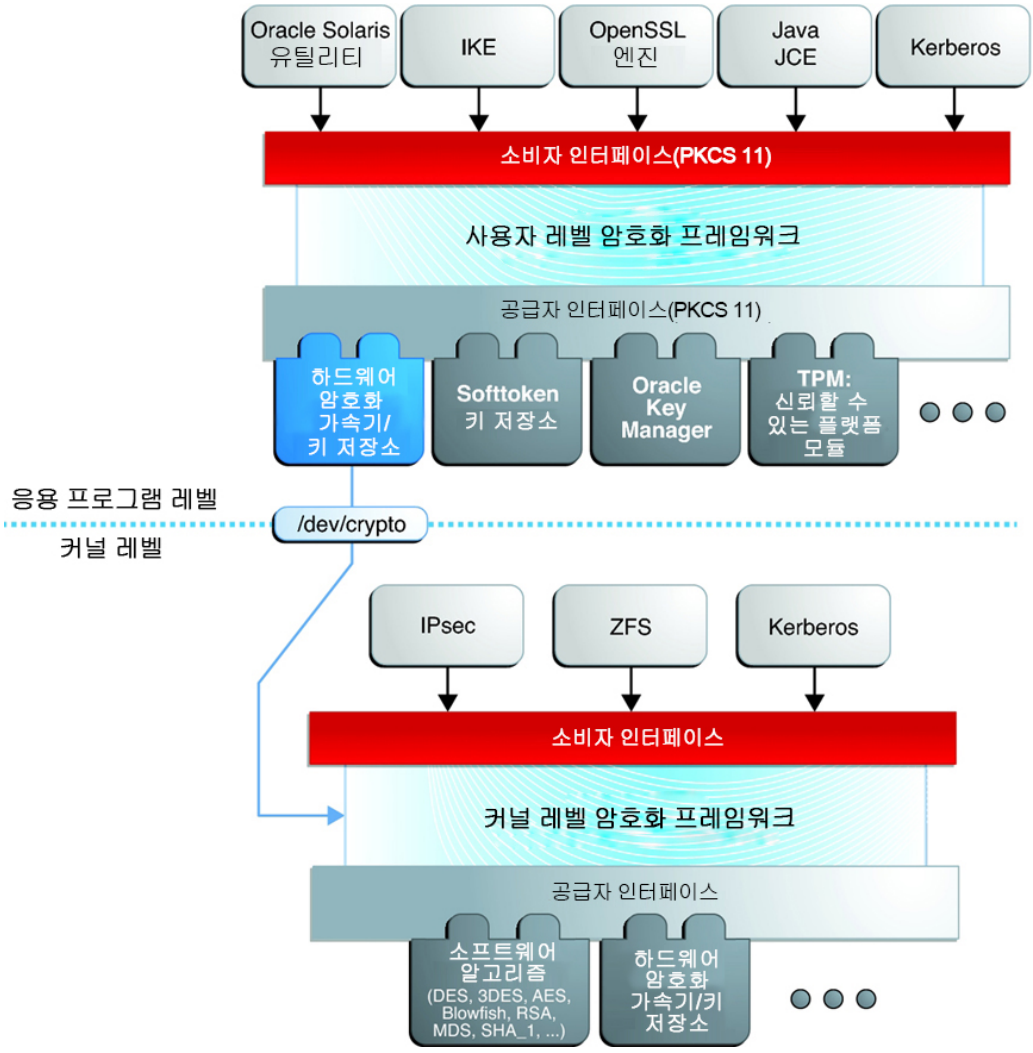
이 절에서는 기존 고객을 위해 이 릴리스에 포함된 암호화 지원의 새로운 기능에 대해 중점적으로 설명합니다.

- Oracle Solaris에서는 FIPS 가능 버전과 FIPS 불가능 버전의 OpenSSL이 둘 다 지원됩니다.
- 암호화 최적화가 포함된 SPARC T4 시스템에서는 하드웨어에서 직접 암호화 지침을 사용할 수 있어 암호화 작업을 더 신속하게 실행할 수 있습니다.
- 암호화 프레임워크는 AES와 비슷하며 일본 시장에서 주로 사용되는 128비트 블록 암호화인 Camellia를 지원합니다.

암호화 프레임워크 소개

암호화 프레임워크는 암호화 요구 사항을 처리하기 위한 알고리즘 및 PKCS #11 라이브러리의 공통 저장소를 제공합니다. PKCS #11 라이브러리는 RSA Security Inc. PKCS #11 암호화 토큰 인터페이스(Cryptoki) 표준에 따라 구현됩니다.

그림 1-1 암호화 프레임워크 레벨



커널 레벨에서 프레임워크는 현재 ZFS, Kerberos, IPsec 및 하드웨어에 대한 암호화 요구 사항을 처리합니다. 사용자 레벨 소비자에는 OpenSSL 엔진, JCE(Java Cryptographic Extensions), `libsasl` 및 IKE(Internet Key Protocol)가 포함됩니다. 커널 SSL(`kssl`) 프록시가 암호화 프레임워크를 사용합니다. 자세한 내용은 "Oracle Solaris 11.2의 네트워크 보안 "의 "SSL 커널 프록시로 웹 서버 통신 암호화" 및 `ksslcfg`(1M) 매뉴얼 페이지를 참조하십시오.

미국 수출법에 따라 개방형 암호화 인터페이스의 사용이 허가되어야 합니다. 암호화 프레임워크는 현행 법규에 맞게 커널 암호화 공급자와 PKCS #11 암호화 공급자의 서명을 요구합니다. 자세한 내용은 "암호화 프레임워크의 사용자 레벨 명령" [11]에서 `elfsign` 명령에 대한 내용을 참조하십시오.

프레임워크에서 암호화 서비스 공급자의 서비스를 Oracle Solaris의 많은 소비자가 사용할 수 있습니다. 공급자의 다른 이름은 플러그인입니다. 프레임워크에서는 다음과 같은 세 가지 유형의 플러그인을 지원합니다.

- 사용자 레벨 플러그인 - PKCS #11 라이브러리(예: `/var/user/$USER/pkcs11_softtoken.so.1`)를 사용하여 서비스를 제공하는 공유 객체입니다.
- 커널 레벨 플러그인 - 소프트웨어의 암호화 알고리즘(예: AES)을 구현하는 커널 모듈입니다.

  프레임워크에서 대부분의 알고리즘은 x86(SSE2 명령 세트 포함) 및 SPARC 하드웨어에 맞게 최적화되어 있습니다. T-Series 최적화는 "암호화 프레임워크 및 SPARC T-Series 서버" [17]를 참조하십시오.
- 하드웨어 플러그인 - 장치 드라이버 및 연관된 하드웨어 가속기입니다. 그 예로 Niagara 칩과 Oracle의 `ncp` 및 `n2cp` 장치 드라이버가 있습니다. 하드웨어 가속기는 운영 체제에서 값비싼 암호화 작업 부담을 덜어줍니다. Sun Crypto Accelerator 6000 보드를 예로 들 수 있습니다.

프레임워크는 사용자 레벨 공급자에 대한 표준 인터페이스인 PKCS #11, v2.20 개정판 3 라이브러리를 구현합니다. 타사 응용 프로그램에서 라이브러리를 사용하여 공급자에 연결할 수 있습니다. 또한 서명된 라이브러리, 서명된 커널 알고리즘 모듈, 서명된 장치 드라이버를 프레임워크에 추가할 수 있습니다. 이러한 플러그인은 IPS(이미지 패키징 시스템)를 통해 타사 소프트웨어가 설치될 때 추가됩니다. 프레임워크의 주요 구성 요소에 대한 다이어그램은 그림 1-1. "암호화 프레임워크 레벨"을 참조하십시오.

암호화 프레임워크의 개념

다음은 암호화 프레임워크로 작업할 때 유용한 예와 개념에 대한 설명입니다.

- **알고리즘** - 암호화 알고리즘은 입력을 암호화하거나 해시하는 확립된 순환적 계산 프로시저입니다. 암호화 알고리즘은 대칭 또는 비대칭일 수 있습니다. 대칭 알고리즘은 암호화 및 해독에 동일한 키를 사용합니다. 비대칭 알고리즘은 공개 키 암호화에 사용되며 두 개의 키가 필요합니다. 해시 함수 역시 알고리즘입니다.

  알고리즘의 예는 다음과 같습니다.

  - 대칭 알고리즘 - AES, ECC
  - 비대칭 알고리즘 - Diffie-Hellman, RSA
  - 해시 함수 - SHA256
- **소비자** - 공급자로부터 전달된 암호화 서비스의 사용자입니다. 소비자는 응용 프로그램, 최종 사용자 또는 커널 작업일 수 있습니다.

  소비자의 예는 다음과 같습니다.

  - 응용 프로그램 - IKE
  - 최종 사용자 - `encrypt` 명령을 실행하는 일반 사용자
  - 커널 작업 - IPsec
- **키 저장소** - 암호화 프레임워크에서 키 저장소는 토큰 객체에 대한 지속 저장소를 의미하며 종종 **토큰**과 혼용되기도 합니다. 예약된 키 저장소에 대한 자세한 내용은 이 정의 목록에서 **Metaslot**을 참조하십시오.
- **방식** - 특정 목적을 위한 알고리즘 모드의 적용입니다.

  예를 들어, 인증에 적용된 DES 방식(예: CKM_DES_MAC)은 암호화에 적용된 DES 방식(예: CKM_DES_CBC_PAD)과 별도의 방식입니다.
- **Metaslot** - 프레임워크에서 로드된 다른 여러 슬롯의 기능을 결합한 단일 슬롯입니다. metaslot을 사용하면 프레임워크를 통해 사용 가능한 공급자의 기능을 쉽게 다룰 수 있습니다. metaslot을 사용하는 응용 프로그램이 작업을 요청하면 metaslot이 작업을 수행할 실제 슬롯을 결정합니다. metaslot 기능은 구성 가능하지만, 구성이 필요하지 않습니다. metaslot은 기본적으로 켜져 있습니다. 자세한 내용은 `cryptoadm`(1M) 매뉴얼 페이지를 참조하십시오.

  metaslot은 고유한 키 저장소가 없습니다. metaslot은 암호화 프레임워크의 실제 슬롯 중 하나를 키 저장소 용도로 예약합니다. 기본적으로 metaslot은 `Sun Crypto Softtoken` 키 저장소를 예약합니다. metaslot에 사용되는 키 저장소는 사용 가능한 슬롯 중 하나로 표시되지 않습니다.

  사용자는 환경 변수 `${METASLOT_OBJECTSTORE_SLOT}` 및 `${METASLOT_OBJECTSTORE_TOKEN}`을 설정하거나 `cryptoadm` 명령을 실행하여 metaslot에 대한 대체 키 저장소를 지정할 수 있습니다. 자세한 내용은 `libpkcs11`(3LIB), `pkcs11_softtoken`(5) 및 `cryptoadm`(1M) 매뉴얼 페이지를 참조하십시오.
- **모드** - 암호화 알고리즘의 버전입니다. 예를 들어, CBC(Cipher Block Chaining)는 ECB(Electronic Code Book)와 다른 모드입니다. AES 알고리즘에는 CKM_AES_ECB 및 CKM_AES_CBC의 두 가지 모드가 있습니다.
- **정책** - 사용할 방식을 관리자가 선택합니다. 기본적으로 모든 공급자와 모든 방식을 사용할 수 있습니다. 어떤 방식을 사용 또는 사용 안함으로 설정하면 정책에 적용됩니다. 정책 설정 및 적용 예는 "암호화 프레임워크 관리" [33]를 참조하십시오.
- **공급자** - 소비자가 사용하는 암호화 서비스입니다. 공급자는 프레임워크에 플러그인되므로 플러그인이라고도 합니다.

  공급자의 예는 다음과 같습니다.
  - PKCS #11 라이브러리 - `/var/user/$USER/pkcs11_softtoken.so`
  - 암호화 알고리즘의 모듈 - `aes`, `arcfour`
  - 장치 드라이버 및 연관된 하드웨어 가속기 - Sun Crypto Accelerator 6000용 `mca` 드라이버
- **슬롯** - 하나 이상의 암호화 장치에 대한 인터페이스입니다. 물리적 판독기나 기타 장치 인터페이스에 해당하는 각 슬롯은 토큰을 포함할 수 있습니다. 토큰은 프레임워크의 암호화 장치에 대한 논리적 뷰를 제공합니다.
- **토큰** - 슬롯에서 토큰은 프레임워크의 암호화 장치에 대한 논리적 뷰를 제공합니다.

암호화 프레임워크 명령 및 플러그인

프레임워크는 공급자를 제공하는 관리자용, 사용자용, 개발자용 명령을 제공합니다.

- 관리 명령 - `cryptoadm` 명령은 사용 가능한 공급자와 관련 기능을 나열하는 `list` 하위 명령을 제공합니다. 일반 사용자가 `cryptoadm list` 및 `cryptoadm --help` 명령을 실행할 수 있습니다.

  기타 모든 `cryptoadm` 하위 명령을 실행하려면 Crypto Management 권한 프로파일이 포함된 역할을 맡거나 수퍼 유저로 로그인해야 합니다. `disable`, `install`, `uninstall`과 같은 하위 명령을 프레임워크 관리에 사용할 수 있습니다. 자세한 내용은 `cryptoadm`(1M) 매뉴얼 페이지를 참조하십시오.

  `svcadm` 명령을 사용하여 `kcfd` 데몬을 관리하고 커널에서 암호화 정책을 새로 고칠 수 있습니다. 자세한 내용은 `svcadm`(1M) 매뉴얼 페이지를 참조하십시오.
- 사용자 레벨 명령 - `digest` 및 `mac` 명령은 파일 무결성 서비스를 제공합니다. `encrypt` 및 `decrypt` 명령은 도청으로부터 파일을 보호합니다. 이 명령을 사용하려면 표 3-1. "암호화 프레임워크 작업 맵으로 파일 보호"을 참조하십시오.

암호화 프레임워크의 관리 명령

`cryptoadm` 명령은 실행 중인 암호화 프레임워크를 관리합니다. 명령은 Crypto Management 권한 프로파일의 일부입니다. 이 프로파일은 암호화 프레임워크의 보안 관리를 위해 역할에 지정할 수 있습니다. `cryptoadm` 명령을 사용하여 다음을 수행합니다.

- 공급자 방식 사용 또는 사용 안함
- metaslot 사용 또는 사용 안함

`svcadm` 명령을 사용하여 암호화 서비스 데몬 `kcfd`를 새로 고치고 사용 또는 사용 안함으로 설정할 수 있습니다. 이 명령은 Oracle Solaris SMF(서비스 관리 기능) 기능의 일부입니다. `svc:/system/cryptosvcs`는 암호화 프레임워크에 대한 서비스 인스턴스입니다. 자세한 내용은 `smf`(5) 및 `svcadm`(1M) 매뉴얼 페이지를 참조하십시오.

암호화 프레임워크의 사용자 레벨 명령

암호화 프레임워크는 파일 무결성을 검사하고 파일을 암호화하고 파일을 해독하기 위한 사용자 레벨 명령을 제공합니다.

- `digest` 명령 - 하나 이상의 파일 또는 stdin에 대한 메시지 다이제스트를 계산합니다. 다이제스트는 파일 무결성 확인에 유용합니다. 다이제스트 함수의 예로 SHA1 및 MD5가 있습니다.

- `mac` 명령 - 하나 이상의 파일이나 stdin에 대한 MAC(메시지 인증 코드)를 계산합니다. MAC은 데이터를 인증된 메시지와 연관시킵니다. MAC을 사용하여 수신자는 메시지가 발신자로부터 왔는지, 메시지가 변조되지 않았는지 확인할 수 있습니다. `sha1_mac` 및 `md5_hmac` 방식은 MAC을 계산할 수 있습니다.
- `encrypt` 명령 - 파일 또는 stdin을 대칭 암호화합니다. `encrypt -l` 명령은 사용 가능한 알고리즘을 나열합니다. 사용자 레벨 라이브러리 아래에 나열된 방식을 `encrypt` 명령에 사용할 수 있습니다. 프레임워크는 사용자 암호화를 위해 AES, DES, 3DES(3중 DES), ARCFOUR 방식을 제공합니다.
- `decrypt` 명령 - `encrypt` 명령으로 암호화된 파일 또는 stdin을 해독합니다. `decrypt` 명령은 원본 파일을 암호화하는 데 사용된 것과 동일한 키 및 방식을 사용합니다.
- `elfsign` 명령 - 암호화 프레임워크와 함께 사용할 공급자를 서명할 수 있습니다. 일반적으로, 이 명령은 공급자의 개발자가 실행합니다. `elfsign` 명령에는 인증서를 요청하고 이진을 서명하고 이진에서 서명을 확인하는 하위 명령이 있습니다. 서명되지 않은 이진은 암호화 프레임워크에서 사용할 수 없습니다. 검증 가능한 서명된 이진이 있는 공급자는 프레임워크를 사용할 수 있습니다.

암호화 프레임워크의 플러그인

타사 공급자를 암호화 프레임워크에 플러그인할 수 있습니다. 타사 공급자는 다음 객체 중 하나일 수 있습니다.

- PKCS #11 공유 라이브러리
- 암호화 알고리즘, MAC 함수, 다이제스트 함수와 같은 로드 가능한 커널 소프트웨어 모듈
- 하드웨어 가속기용 커널 장치 드라이버

공급자의 객체를 Oracle의 인증서로 서명해야 합니다. 인증서 요청은 타사가 선택한 개인 키와 Oracle이 제공한 인증서를 기반으로 합니다. 인증서 요청을 Oracle로 보내면 타사를 등록한 후 인증서를 발행합니다. 그런 다음 타사 공급자 객체를 Oracle의 인증서로 서명합니다.

로드 가능한 커널 소프트웨어 모듈과 하드웨어 가속기용 커널 장치 드라이버도 커널에 등록해야 합니다. 등록은 암호화 프레임워크 SPI(서비스 공급자 인터페이스)를 통해 이루어집니다.

암호화 서비스 및 영역

전역 영역과 각 비전역 영역에는 고유의 `/system/cryptosvc` 서비스가 있습니다. 전역 영역에서 암호화 서비스를 사용으로 설정하거나 새로 고치면 `kcfd` 데몬이 전역 영역에서 시작되고, 전역 영역에 대한 사용자 레벨 정책이 설정되고, 시스템에 대한 커널 정책이 설정됩니다. 비전역 영역에서 서비스를 사용으로 설정하거나 새로 고치면 `kcfd` 데몬이 영역에서 시작되

고, 영역에 대한 사용자 레벨 정책이 설정됩니다. 커널 정책은 전역 영역에서 설정되었습니다.

영역에 대한 자세한 내용은 "Oracle Solaris 영역 소개 "를 참조하십시오. SMF를 사용하여 지속성 응용 프로그램을 관리하는 데 대한 자세한 내용은 "Oracle Solaris 11.2의 시스템 서비스 관리 "의 1 장, "서비스 관리 기능 소개" 및 `smf`(5) 매뉴얼 페이지를 참조하십시오.

암호화 프레임워크 및 FIPS 140

FIPS 140은 암호화 모듈에 대한 미국 정부 컴퓨터 보안 표준입니다. Oracle Solaris 시스템은 FIPS 140-2 레벨 1에서 승인된 암호화 알고리즘에 두 가지 공급자를 제공합니다.

해당 공급자는 다음과 같습니다.

- Oracle Solaris 암호화 프레임워크는 두 개의 FIPS 140 승인 모듈을 제공합니다. *userland* 모듈은 사용자 공간에서 실행되는 응용 프로그램에 대한 암호화를 제공합니다. 커널 모듈은 커널 레벨 프로세스에 대한 암호화를 제공합니다.
- OpenSSL 객체 모듈은 SSH 및 웹 응용 프로그램에 대한 FIPS 140 승인 암호화를 제공합니다.

다음 주요 고려 사항에 유의하십시오.

- FIPS 140-2 공급자 모듈은 CPU를 많이 사용하므로 기본적으로 사용으로 설정되지 않습니다. 시스템 관리자는 FIPS 140 모드에서 공급자를 사용으로 설정하고 FIPS 승인 알고리즘을 사용할 응용 프로그램을 구성할 책임이 있습니다.
- FIPS 140-2에서 검증한 암호화만 사용하도록 엄격한 요구 사항을 적용할 경우 Oracle Solaris11.1 SRU5.5 릴리스나 Oracle Solaris11.1 SRU3 릴리스를 실행 중이어야 합니다. Oracle은 이러한 두 가지 릴리스에서 Solaris 암호화 프레임워크에 대해 FIPS 140-2 검증을 마쳤습니다. Oracle Solaris11.2는 이러한 검증된 기반 위에 구축되어 성능, 기능성, 안정성을 보장하는 향상된 소프트웨어 기능을 제공합니다. 이러한 향상된 기능을 활용하려면 가능한 Oracle Solaris11.2를 FIPS 140-2 모드로 구성해야 합니다.

자세한 내용은 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "을 참조하십시오. 이 문서에서는 다음 항목을 다룹니다.

- Oracle Solaris의 FIPS 140-2 레벨 1 암호화 개요
- FIPS 140 공급자 사용으로 설정
- FIPS 140 소비자 사용으로 설정
- FIPS 140 모드로 두 가지 응용 프로그램을 사용으로 설정하는 예제
- FIPS 140 승인 알고리즘 및 인증서 참조

다음 추가 정보를 사용할 수 있습니다.

- FIPS 140 가능 OpenSSL 구현으로 전환하는 방법 [14]

■ “FIPS 140이 사용으로 설정된 부트 환경 만들기 ” [41]

Oracle Solaris의 OpenSSL 지원

Oracle Solaris에서는 OpenSSL의 두 가지 구현을 지원합니다.

- FIPS 140 가능 OpenSSL
- FIPS 140 불가능 OpenSSL

두 구현 모두 OpenSSL 1.0.1인 OpenSSL 프로젝트에서 최신 OpenSSL 버전과 호환되도록 업그레이드되었습니다. 버전 라이브러리와 관련하여 둘 다 API/ABI와 호환됩니다.

OS에 두 구현이 모두 있지만 한 번에 한 가지 구현만 활성 상태일 수 있습니다. 시스템에서 활성 상태인 OpenSSL 구현을 확인하려면 `pkg mediator openssl` 명령을 사용하십시오.

▼ FIPS 140 가능 OpenSSL 구현으로 전환하는 방법

기본적으로 FIPS 140 불가능 OpenSSL 구현이 Oracle Solaris에서 활성 상태입니다. 하지만 시스템의 보안을 선택하고 원하는 구현을 선택할 수 있습니다.

1. **관리자로 로그인합니다.**

2. **두 가지 구현이 모두 시스템에 있어야 합니다.**

```
$ pkg mediator -a openssl
```

주의 - 전환하려는 OpenSSL 구현이 시스템에 있어야 합니다. 그렇지 않으면 시스템에 없는 구현으로 전환할 경우 시스템을 사용할 수 없게 될 수 있습니다.

3. **다른 OpenSSL 구현으로 전환합니다.**

```
# pkg set-mediator [--be-name name] -I implementation openssl
```

여기서 *implementation*은 `default` 또는 `fips-140`이고 *name*은 현재 부트 환경의 새 복제본 이름입니다. 복제본에 지정된 구현이 활성화됩니다.

참고 - *--be-name*이 지정된 경우 이 명령은 현재 부트 환경의 백업을 만듭니다. 시스템을 재부트하면 새로 복제된 부트 환경이 새로운 구현으로 실행됩니다.

`pkg set-mediator` 명령에 대한 자세한 내용은 “Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 ”의 “기본 응용 프로그램 변경”을 참조하십시오.

4. **시스템을 재부트합니다.**

5. **(옵션) 전환이 성공적이고 원하는 OpenSSL 구현이 활성 상태인지 확인합니다.**

```
# pkg mediator openssl
```

예 1-1 FIPS 140 가능 OpenSSL 구현으로 전환

이 예에서는 시스템의 OpenSSL 구현을 FIPS 140 가능으로 변경합니다.

```
# pkg mediator -a openssl
MEDIATOR   VER. SRC.   VERSION IMPL.   SRC. IMPLEMENTATION
openssl       vendor           vendor               default
openssl       system          system              fips-140

# pkg set-mediator --be-name BE2 -I fips-140 openssl
# reboot

# pkg mediator openssl
MEDIATOR   VER. SRC.   VERSION IMPL.   SRC. IMPLEMENTATION
openssl      vendor           vendor              default
```

♦ ♦ ♦ 2 장

2 SPARC T-Series 시스템 및 암호화 프레임워크 정보

이 장에서는 SPARC T-series 서버의 암호화 프레임워크 및 암호화 기능의 성능을 향상시키는 Oracle Solaris 11의 최적화에 대해 설명합니다.

암호화 프레임워크 및 SPARC T-Series 서버

암호화 프레임워크는 SPARC T-Series 시스템에 암호화 방식을 제공하며 이러한 서버의 일부 방식을 최적화합니다. 세 가지 암호화 방식 `AES-CBC`, `AES-CFB128` 및 `ARCFOUR`이 저장된 데이터 및 이동되는 데이터에 맞게 최적화됩니다. DES 암호화 방식은 OpenSSL용으로 최적화되며 모든 정밀도 연산(bignum)과 RSA 및 DSA를 최적화하여 제공됩니다. 다른 최적화에는 핸드셰이크 및 이동 중인 데이터에 대한 작은 패킷 성능이 포함됩니다.

이 릴리스에서는 다음과 같은 암호화 방식을 사용할 수 있습니다.

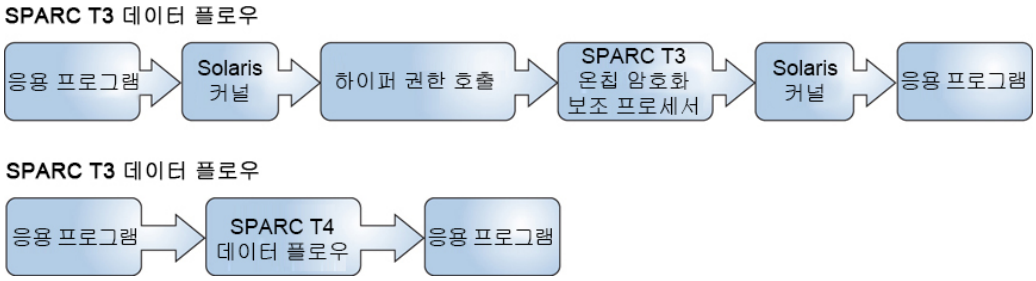
- `AES-XTS` - 보관 중인 데이터에 사용됩니다.
- `SHA-224` - `SHA2` 방식
- `AES-XCBC-MAC` - IPsec에 사용됨

SPARC T-4 시스템의 암호화 최적화

SPARC T4 마이크로 프로세서부터는 암호화 기능을 수행하기 위한 새로운 지침을 하드웨어에서 직접 사용할 수 있게 되었습니다. 지침은 권한이 없으므로 어느 프로그램에서나 커널 환경, 루트 권한 또는 그 밖의 특별한 설정 없이 지침을 사용할 수 있습니다. 여러 하위 레벨 지침을 사용하지 않고 하드웨어에서 직접 암호화가 수행되므로 이전의 SPARC 프로세서에 별도의 암호화용 처리 장치가 있는 시스템의 작업에 비해 암호화 작업 속도가 빨라졌습니다.

다음 비교에서는 SPARC T-3 시스템 및 암호화 최적화가 포함된 SPARC T-4 시스템의 데이터 흐름 차이를 보여줍니다.

그림 2-1 SPARC T 시스템 사이의 데이터 흐름 비교



다음 표에서는 측정 Oracle Solaris 릴리스와 결합된 SPARC T 마이크로 프로세서 장치의 암호화 기능을 자세히 비교합니다.

표 2-1 SPARC T-Series 서버의 암호화 성능

기능/소프트웨어 소비자	T-3 및 이전 시스템	Oracle Solaris 10을 실행하는 T-4 시스템	Oracle Solaris 11을 실행하는 T-4 시스템
SSH	Solaris 10 5/09 이상에서는 사용으로 자동 설정됩니다. `/etc/ssh/sshd_config`에서 `UseOpenSSLEngine` 절을 통해 사용 안함/사용으로 설정합니다.	패치 147707-01이 필요합니다. `/etc/ssh/sshd_config`에서 `UseOpenSSLEngine` 절을 통해 사용 안함/사용으로 설정합니다.	사용으로 자동 설정됩니다. `/etc/ssh/sshd_config`에서 `UseOpenSSLEngine` 절을 통해 사용 안함/사용으로 설정합니다.
Java/JCE	사용으로 자동 설정됩니다. `$JAVA_HOME/jre/lib/security/java.security`에서 구성합니다.	사용으로 자동 설정됩니다. `$JAVA_HOME/jre/lib/security/java.security`에서 구성합니다.	사용으로 자동 설정됩니다. `$JAVA_HOME/jre/lib/ security/java.security`에서 구성합니다.
ZFS 암호화	사용할 수 없습니다.	사용할 수 없습니다.	데이터 세트가 암호화되면 HW 암호화가 사용으로 자동 설정됩니다.
IPsec	사용으로 자동 설정됩니다.	사용으로 자동 설정됩니다.	사용으로 자동 설정됩니다.
OpenSSL	`-engine pkcs11`을 사용합니다.	패치 147707-01이 필요합니다. `-engine pkcs11`을 사용합니다.	T4 최적화가 자동으로 사용됩니다. 선택적으로 `-engine pkcs11`을 사용할 수도 있습니다. 여기서는 RSA/DSA에 권장되는 `pkcs11`을 사용합니다.
KSSL(커널 SSL 프록시)	사용으로 자동 설정됩니다.	사용으로 자동 설정됩니다.	사용으로 자동 설정됩니다.
Oracle TDE	지원되지 않습니다.	패치 보류 중입니다.	Oracle DB 11.2.0.3 및 ASO에서 사용으로 자동 설정됩니다.

기능/소프트웨어 소비자	T-3 및 이전 시스템	Oracle Solaris 10을 실행하는 T-4 시스템	Oracle Solaris 11을 실행하는 T-4 시스템
Apache SSL	`SSLCryptoDevice pkcs11`로 구성합니다.	`SSLCryptoDevice pkcs11`로 구성합니다.	`SSLCryptoDevice pkcs11`로 구성합니다.
논리적 도메인	도메인에 암호화 단위를 지정합니다.	항상 기능을 사용할 수 있으며 구성은 필요 없습니다.	항상 기능을 사용할 수 있으며 구성은 필요 없습니다.

T4 암호화 지침에는 다음 항목이 포함됩니다.

- `aes_kexpand0`, `aes_kexpand1`, `aes_kexpand2`

  이 지침이 키 확장을 수행하며 128비트, 192비트 또는 256비트 사용자 제공 키를 암호화 및 해독 중에 내부적으로 사용되는 키 일정으로 확장합니다. `aes_kexpand2` 지침은 AES-256에만 사용됩니다. 나머지 두 `aes_kexpand` 지침은 AES-128, AES-192, AES-256 등의 모든 키 길이에 사용됩니다.
- `aes_eround01`, `aes_eround23`, `aes_eround01_l`, `aes_eround_23_l`

  이 지침은 AES 암호화 라운드 또는 변환에 사용됩니다. FIPS 197의 AES 표준에 의거하여 더 많은 계산 비용으로 강력한 암호화를 원할수록 큰 키를 사용하므로 사용되는 라운드 수(예: 10, 12 또는 14)는 AES 키 길이에 따라 달라집니다.
- `aes_dround01`, `aes_dround23`, `aes_dround01_l`, `aes_dround_23_l`

  이 지침은 암호화와 비슷한 방식으로 AES 해독 라운드에 사용됩니다.
- DES/DES-3, Kasumi, Camellia, Montgomery 곱셈/제곱근(RSA Bignum용) 및 CRC32c 체크섬을 위한 지침
- MD5, SHA1 및 SHA2 다이제스트 지침

SPARC T4 하드웨어 암호화 지침이 제공되며, Oracle Solaris 11을 실행하는 SPARC T4 시스템에서 시스템의 T4 마이크로 프로세서의 내장 t4 엔진으로 자동 사용됩니다. Oracle Solaris 11.2부터는 OpenSSL 업스트림 코드에 이 지침이 내장되므로 이번 릴리스에서는 해당 지침을 사용할 수 있도록 패치와 함께 OpenSSL 1.0.1e가 제공됩니다.

T4 지침에 대한 자세한 내용은 다음 문서를 참조하십시오.

- "SPARC T4 OpenSSL Engine" (https://blogs.oracle.com/DanX/entry/sparc_t4_openssl_engine)
- "How to tell if SPARC T4 crypto is being used?" (https://blogs.oracle.com/DanX/entry/how_to_tell_if_sparc)
- "Exciting Crypto Advances with the T4 processor and Oracle Solaris 11" (http://bubbva.blogspot.com/2011/11/exciting-crypto-advances-with-t4.html)
- "SPARC T4 Digest and Crypto Optimizations in Solaris 11.1" (https://blogs.oracle.com/DanX/entry/sparc_t4_digest_and_crypto)

시스템의 SPARC T4 최적화 지원 여부 확인

T4 최적화가 사용 중인지 확인하려면 `isainfo` 명령을 사용하십시오. 출력에 `sparcv9` 및 `aes`가 포함되어 있으면 시스템에서 최적화를 사용하고 있는 것입니다.

```
$ isainfo -v
64-bit sparcv9 applications
        crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5 camellia kasumi
        des aes ima hpc vis3 fmaf asi_blk_init vis2 vis popc
```

시스템의 OpenSSL 버전 확인

시스템에서 실행 중인 OpenSSL의 버전을 확인하려면 `openssl version`을 입력하십시오. 결과는 다음과 유사합니다.

```
OpenSSL 1.0.0j 10 May 2012
```

시스템에 SPARC T4 최적화가 포함된 OpenSSL이 있는지 확인

시스템에서 SPARC T4 최적화가 포함된 OpenSSL을 지원하는지 여부를 확인하려면 다음과 같이 `libcrypto.so` 라이브러리를 확인하십시오.

```
# nm /lib/libcrypto.so.1.0.0 | grep des_t4
[5239]  |    504192|       300|FUNC |GLOB |3    |12     |des_t4_cbc_decrypt
[5653]  |    503872|       300|FUNC |GLOB |3    |12     |des_t4_cbc_encrypt
[4384]  |    505024|       508|FUNC |GLOB |3    |12     |des_t4_ede3_cbc_decrypt
[2963]  |    504512|       508|FUNC |GLOB |3    |12     |des_t4_ede3_cbc_encrypt
[4111]  |    503712|       156|FUNC |GLOB |3    |12     |des_t4_key_expand
```

명령이 출력을 생성하지 않을 경우 시스템에서 OpenSSL에 대해 SPARC T4 최적화를 지원하지 않는 것입니다.

♦ ♦ ♦ 3 장

3 암호화 프레임워크

이 장에서는 암호화 프레임워크를 사용하는 방법에 대해 설명하고 다음 항목을 다룹니다.

암호화 프레임워크로 파일 보호

이 절에서는 대칭 키를 생성하는 방법, 파일 무결성을 위한 체크섬을 만드는 방법, 도청으로부터 파일을 보호하는 방법을 설명합니다. 이 절의 명령은 일반 사용자가 실행할 수 있습니다. 개발자는 이러한 명령을 사용하는 스크립트를 작성할 수 있습니다.

FIPS 140 모드로 시스템을 설정하려면 FIPS에서 검증한 알고리즘, 모드, 키 길이를 사용해야 합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithm Lists and Certificate References for Oracle Solaris Systems"를 참조하십시오.

암호화 프레임워크는 파일을 보호하도록 도울 수 있습니다. 다음 작업 맵은 사용 가능한 알고리즘을 나열하고 암호화 기법으로 파일을 보호하기 위한 절차를 가리킵니다.

표 3-1 암호화 프레임워크 작업 맵으로 파일 보호

작업	설명	지침
대칭 키를 생성합니다.	사용자가 지정한 길이의 키를 생성합니다. 선택적으로 파일, PKCS #11 키 저장소 또는 NSS 키 저장소에 키를 저장합니다. FIPS 140 승인 모드의 경우 FIPS에서 검증한 키 유형, 모드, 키 길이를 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.	`pktool` 명령을 사용하여 대칭 키를 생성하는 방법 [22]
파일 무결성을 보장하는 체크섬을 제공합니다.	수신자의 파일 복사본이 보낸 파일과 같은지 확인합니다.	파일의 다이제스트를 계산하는 방법 [27]
MAC(메시지 인증 코드)으로 파일을 보호합니다.	본인이 발신자임을 메시지 수신자에게 확인합니다.	파일의 MAC을 계산하는 방법 [28]

작업	설명	지침
파일을 암호화하고, 암호화된 파일을 해독합니다.	파일을 암호화하여 파일 내용을 보호합니다. 파일을 해독하려면 암호화 매개변수를 제공합니다.	파일을 암호화 및 해독하는 방법 [30]

▼ pktool 명령을 사용하여 대칭 키를 생성하는 방법

일부 응용 프로그램에서 통신을 암호화 및 해독하려면 대칭 키가 필요합니다. 이 절차에서 대칭 키를 만들고 저장합니다.

사이트에 난수 생성기가 있는 경우 생성기를 사용하여 키에 대한 난수를 생성할 수 있습니다. 이 절차에서는 사이트의 난수 생성기를 사용하지 않습니다.

1. **(옵션) 키 저장소를 사용하려면 하나 만듭니다.**

   - **PKCS #11 키 저장소를 만들고 초기화하려면 pktool setpin 명령을 사용하여 암호문을 생성하는 방법 [58]을 참조하십시오.**
   - **NSS 데이터베이스를 만들고 초기화하려면 예 4-5. “키 저장소를 문장암호로 보호”의 샘플 명령을 참조하십시오.**

2. **대칭 키로 사용할 난수를 생성합니다.**

   다음 방법 중 하나를 사용합니다.

   - **키를 생성하고 파일에 저장합니다.**

     파일에 저장된 키의 이점은, 응용 프로그램의 키 파일(예: /etc/inet/secret/ipseckeys 파일 또는 IPsec)에 사용할 키를 이 파일에서 추출할 수 있다는 것입니다. 사용법 명령문은 인수를 보여줍니다.

```
% pktool genkey keystore=file
...genkey keystore=file
outkey=key-fn
[ keytype=aes|arcfour|des|3des|generic ]
[ keylen=key-size (AES, ARCFOUR or GENERIC only)]
[ print=y|n ]
```

     outkey=*key-fn*

     키가 저장되는 파일 이름입니다.

     keytype=*specific-symmetric-algorithm*

     모든 길이의 대칭 키의 경우 값이 generic입니다. 특정 알고리즘에 대해 aes, arcfour, des 또는 3des를 지정합니다.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 유형을 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`keylen=`*size-in-bits*

비트 단위의 키 길이입니다. 숫자는 8로 나눌 수 있어야 합니다. `des` 또는 `3des`에 대해서는 지정하지 마십시오.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 길이를 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`print=n`

터미널 창에 키를 인쇄합니다. 기본적으로 `print`의 값은 `n`입니다.

■ **키를 생성하고 PKCS #11 키 저장소에 저장합니다.**

PKCS #11 키 저장소의 이점은, 레이블로 키를 검색할 수 있다는 것입니다. 이 방법은 파일을 암호화하고 해독하는 키에 유용합니다. 이 방법을 사용하기 전에 1단계을 완료해야 합니다. 사용법 명령문은 인수를 보여줍니다. 키 저장소 인수 주위의 대괄호는 키 저장소 인수를 지정하지 않을 경우 키가 PKCS #11 키 저장소에 저장됨을 나타냅니다.

```
$ pktool genkey keystore=pkcs11
...genkey [ keystore=pkcs11 ]
label=key-label
[ keytype=aes|arcfour|des|3des|generic ]
[ keylen=key-size (AES, ARCFOUR or GENERIC only)]
[ token=token[:manuf[:serial]]]
[ sensitive=y|n ]
[ extractable=y|n ]
[ print=y|n ]
```

`label=`*key-label*

사용자가 지정한 키의 레이블입니다. 레이블로 키 저장소에서 키를 검색할 수 있습니다.

`keytype=`*specific-symmetric-algorithm*

모든 길이의 대칭 키의 경우 값이 `generic`입니다. 특정 알고리즘에 대해 `aes`, `arcfour`, `des` 또는 `3des`를 지정합니다.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 유형을 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`keylen=`*size-in-bits*

비트 단위의 키 길이입니다. 숫자는 8로 나눌 수 있어야 합니다. `des` 또는 `3des`에 대해서는 지정하지 마십시오.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 길이를 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`token=`*token*

토큰 이름입니다. 기본적으로 토큰은 `Sun Software PKCS#11 softtoken`입니다.

`sensitive=n`

키의 민감도를 지정합니다. 값이 `y`이면 `print=y` 인수를 사용하여 키를 인쇄할 수 없습니다. 기본적으로 `sensitive`의 값은 `n`입니다.

`extractable=y`

키 저장소에서 키를 추출할 수 있는지 지정합니다. 키가 추출되지 않도록 하려면 `n`을 지정합니다.

`print=n`

터미널 창에 키를 인쇄합니다. 기본적으로 `print`의 값은 `n`입니다.

■ **키를 생성하고 NSS 키 저장소에 저장합니다.**

이 방법을 사용하기 전에 1단계을 완료해야 합니다. 사용법 명령문은 인수를 보여줍니다.

```
$ pktool genkey keystore=nss
...genkey keystore=nss
label=key-label
[ keytype=aes|arcfour|des|3des|generic ]
[ keylen=key-size (AES, ARCFOUR or GENERIC only)]
[ token=token[:manuf[:serial]]]
[ dir=directory-path ]
[ prefix=DBprefix ]
```

`label=`*key-label*

사용자가 지정한 키의 레이블입니다. 레이블로 키 저장소에서 키를 검색할 수 있습니다.

`keytype=`*specific-symmetric-algorithm*

모든 길이의 대칭 키의 경우 값이 `generic`입니다. 특정 알고리즘에 대해 `aes`, `arcfour`, `des` 또는 `3des`를 지정합니다.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 유형을 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`keylen=`*size-in-bits*

비트 단위의 키 길이입니다. 숫자는 8로 나눌 수 있어야 합니다. `des` 또는 `3des`에 대해서는 지정하지 마십시오.

FIPS 140 승인 알고리즘의 경우 FIPS에서 검증한 키 길이를 선택합니다. "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"을 참조하십시오.

`token=`*token*

토큰 이름입니다. 기본적으로 토큰은 NSS 내부 토큰입니다.

`dir=`*directory*

NSS 데이터베이스에 대한 디렉토리 경로입니다. 기본적으로 *directory*가 현재 디렉토리입니다.

`prefix=`*directory*

NSS 데이터베이스의 접두어입니다. 기본값은 접두어 없음입니다.

3. **(옵션) 키가 존재하는지 확인합니다.**

키를 저장한 위치에 따라 다음 명령 중 하나를 사용합니다.

- ***key-fn* 파일에서 키를 확인합니다.**

```
% pktool list keystore=file objtype=key [infile=key-fn]
Found n keys.
Key #1 - keytype:location (keylen)
```

- **PKCS #11 또는 NSS 키 저장소에서 키를 확인합니다.**

```
For PKCS #11, use the following command:

$ pktool list keystore=pkcs11 objtype=key
Enter PIN for keystore:
Found n keys.
Key #1 - keytype:location (keylen)
```

또는 명령에서 `keystore=pkcs11`을 `keystore=nss`로 바꿉니다.

예 3-1 `pktool` 명령을 사용하여 대칭 키 만들기

다음 예에서 사용자가 처음으로 PKCS #11 키 저장소를 만들고, 응용 프로그램에 대한 대형 대칭 키를 생성합니다. 마지막으로, 키가 키 저장소에 있는지 확인합니다.

PKCS #11 키 저장소의 초기 암호는 `changeme`입니다. NSS 키 저장소의 초기 암호는 비어 있는 암호입니다.

```
# pktool setpin
Create new passphrase:      Type password
Re-enter new passphrase:    Retype password
Passphrase changed.
% pktool genkey label=specialappkey keytype=generic keylen=1024
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
```

```
% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-------------------------------------------------
Symmetric keys:
1        Symmetric     1024          specialappkey
```

예 3-2 pktool 명령을 사용하여 FIPS에서 승인한 AES 키 만들기

다음 예에서 FIPS 승인 알고리즘과 키 유형을 사용하여 AES 알고리즘의 보안 키를 만듭니다. 나중에 해독을 위해 로컬 파일에 키를 저장합니다. 명령이 400 사용 권한으로 파일을 보호합니다. 키를 만들 때 print=y 옵션이 터미널 창에 생성된 키를 표시합니다.

키 파일을 소유한 사용자가 od 명령을 사용하여 키를 검색합니다.

```
% pktool genkey keystore=file outkey=256bit.file1 keytype=aes keylen=256 print=y
Key Value ="aaa2df1d10f02eaee2595d48964847757a6a49cf86c4339cd5205c24ac8c8873"
% od -x 256bit.file1

0000000 aaa2 df1d 10f0 2eae e259 5d48 9648 4775
0000020 7a6a 49cf 86c4 339c d520 5c24 ac8c 8873
0000040
```

예 3-3 IPsec 보안 연관에 대한 대칭 키 만들기

다음 예에서 관리자가 수동으로 IPsec SA에 대한 키 관련 자료를 만들고 파일에 저장합니다. 그런 다음, 관리자가 /etc/inet/secret/ipseckeys 파일로 키를 복사하고 원본 파일을 삭제합니다.

먼저, 관리자가 IPsec 정책에 필요한 키를 만들고 표시합니다.

```
# pktool genkey keystore=file outkey=ipencrin1 keytype=generic keylen=192 print=y
Key Value ="294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049"
# pktool genkey keystore=file outkey=ipencrout1 keytype=generic keylen=192 print=y
Key Value ="9678f80e33406c86e3d1686e50406bd0434819c20d09d204"
# pktool genkey keystore=file outkey=ipspi1 keytype=generic keylen=32 print=y
Key Value ="acbeaa20"
# pktool genkey keystore=file outkey=ipspi2 keytype=generic keylen=32 print=y
Key Value ="19174215"
# pktool genkey keystore=file outkey=ipsha21 keytype=generic keylen=256 print=y
Key Value ="659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e"
# pktool genkey keystore=file outkey=ipsha22 keytype=generic keylen=256 print=y
Key Value ="b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810"
```

그런 후 관리자가 다음 /etc/inet/secret/ipseckeys 파일을 만듭니다.

```
##   SPI values require a leading 0x.
##   Backslashes indicate command continuation.
##
## for outbound packets on this system
add esp spi 0xacbeaa20 \
```

```
src 192.168.1.1 dst 192.168.2.1 \
encr_alg aes auth_alg sha256  \
encrkey  294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049 \
authkey  659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e
##
## for inbound packets
add esp spi 0x19174215 \
src 192.168.2.1 dst 192.168.1.1 \
encr_alg aes auth_alg sha256  \
encrkey 9678f80e33406c86e3d1686e50406bd0434819c20d09d204 \
authkey b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810
```

ipseckeys 파일의 구문이 유효한지 확인한 후에 관리자가 원본 키 파일을 삭제합니다.

```
# ipseckey -c /etc/inet/secret/ipseckeys
# rm ipencrin1 ipencrout1 ipspi1 ipspi2 ipsha21 ipsha22
```

관리자가 ssh 명령 또는 다른 보안 방식을 사용하여 ipseckeys 파일을 통신 시스템에 복사합니다. 통신 시스템에서 보호 사항이 반전됩니다. ipseckeys 파일의 첫번째 항목이 인바운드 패킷을 보호하고, 두번째 항목이 아웃바운드 패킷을 보호합니다. 통신 시스템에는 키가 생성되지 않습니다.

다음 순서 계속 키를 사용하여 파일의 MAC(메시지 인증 코드)를 만들려면 파일의 MAC을 계산하는 방법 [28]을 참조하십시오.

▼ 파일의 다이제스트를 계산하는 방법

파일의 다이제스트를 계산할 때 다이제스트 출력을 비교하여 파일이 변조되지 않았는지 확인할 수 있습니다. 다이제스트는 원본 파일을 고치지 않습니다.

1. **사용 가능한 다이제스트 알고리즘을 나열합니다.**

```
% digest -l
md5
sha1
sha224
sha256
sha384
sha512
```

참고 - 가능하면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에서 목록별 FIPS 승인 알고리즘을 선택하십시오.

2. **파일의 다이제스트를 계산하고 다이제스트 목록을 저장합니다.**

digest 명령으로 알고리즘을 제공합니다.

```
% digest -v -a algorithm input-file > digest-listing
```

-v — 다음 형식으로 출력을 표시합니다.

algorithm (*input-file*) = *digest*

-a *algorithm* — 파일의 다이제스트를 계산하는 데 사용할 알고리즘입니다. 1단계의 출력에 나타난 대로 알고리즘을 입력합니다.

참고 - 가능하면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에 나열된 FIPS 승인 알고리즘을 선택하십시오.

input-file — `digest` 명령에 대한 입력 파일입니다.

digest-listing — `digest` 명령에 대한 출력 파일입니다.

예 3-4 SHA1 방식으로 다이제스트 계산

다음 예에서 `digest` 명령이 SHA1 방식을 사용하여 디렉토리 목록을 제공합니다. 결과가 파일에 배치됩니다.

```
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd0cfa2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564fdf28027d10e1d983
```

▼ 파일의 MAC을 계산하는 방법

메시지 인증 코드(또는 MAC)는 파일의 다이제스트를 계산하고 보안 키를 사용하여 다이제스트를 한층 더 보호합니다. MAC은 원본 파일을 고치지 않습니다.

1. **사용 가능한 방식을 나열합니다.**

```
% mac -l
Algorithm       Keysize:  Min   Max
---------------------------------
des_mac                   64    64
sha1_hmac                  8   512
md5_hmac                   8   512
sha224_hmac                8   512
sha256_hmac                8   512
sha384_hmac                8  1024
```

```
sha512_hmac                8  1024
```

참고 - 각 지원되는 알고리즘은 특정 알고리즘 유형의 가장 흔히 사용되고 가장 제한이 적은 버전의 별칭입니다. 위의 출력은 사용 가능한 알고리즘 이름과 각 알고리즘의 키 크기를 보여줍니다. 가능하면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에 나열된, FIPS 승인 알고리즘과 FIPS 승인 키 길이를 대응한 지원되는 알고리즘을 사용하십시오.

2. **적절한 길이의 대칭 키를 생성합니다.**

   키 생성에 사용하거나 키를 제공할 수 있는 문장암호를 제공할 수 있습니다.

   - 문장암호를 제공하는 경우 문장암호를 저장하거나 기억해야 합니다. 문장암호를 온라인으로 저장할 경우 본인만 문장암호 파일을 읽을 수 있어야 합니다.
   - 키를 제공하는 경우 방식에 맞는 올바른 크기여야 합니다. `pktool` 명령을 사용할 수 있습니다. 절차 및 일부 예제는 `pktool` 명령을 사용하여 대칭 키를 생성하는 방법 [22]을 참조하십시오.

3. **파일에 대한 MAC을 만듭니다.**

   `mac` 명령으로 키를 제공하고 대칭 키 알고리즘을 사용합니다.

   % **mac [-v] -a** *algorithm* **[-k** *keyfile* **| -K** *key-label* **[-T** *token***]]** *input-file*

-v	다음 형식으로 출력을 표시합니다. *algorithm* (*input-file*) = *mac*
-a *algorithm*	MAC을 계산하는 데 사용할 알고리즘입니다. `mac -l` 명령의 출력에 나타난 대로 알고리즘을 입력합니다.
-k *keyfile*	알고리즘이 지정된 길이의 키를 포함하는 파일입니다.
-K *key-label*	PKCS #11 키 저장소에서 키의 레이블입니다.
-T *token*	토큰 이름입니다. 기본적으로 토큰은 `Sun Software PKCS#11 softtoken`입니다. -K *key-label* 옵션을 사용할 때만 사용됩니다.
input-file	MAC에 대한 입력 파일입니다.

예 3-5 SHA1_HMAC 및 문장암호로 MAC 계산

다음 예에서는 SHA1_HMAC 방식과 문장암호에서 파생된 키를 사용하여 전자 메일 첨부 파일이 인증됩니다. MAC 목록이 파일에 저장됩니다. 문장암호가 파일에 저장된 경우 사용자 이외의 다른 사람이 파일을 읽을 수 없어야 합니다.

```
% mac -v -a sha1_hmac email.attach
Enter passphrase:      Type passphrase
```

```
sha1_hmac (email.attach) = 2b31536d3b3c0c6b25d653418db8e765e17fe07b
% echo "sha1_hmac (email.attach) = 2b31536d3b3c0c6b25d653418db8e765e17fe07b" \
>> ~/sha1hmac.daily.05.12
```

예 3-6 SHA1_HMAC 및 키 파일로 MAC 계산

다음 예에서 SHA1_HMAC 방식과 보안 키를 사용하여 디렉토리 매니페스트가 인증됩니다. 결과가 파일에 배치됩니다.

```
% mac -v -a sha1_hmac \
-k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

예 3-7 SHA1_HMAC 및 키 레이블로 MAC 계산

다음 예에서 SHA1_HMAC 방식과 보안 키를 사용하여 디렉토리 매니페스트가 인증됩니다. 사용자의 PKCS #11 키 저장소에 결과가 놓입니다. 사용자가 초기에 `pktool setpin` 명령을 사용하여 키 저장소와 키 저장소의 암호를 만들었습니다.

```
% mac -a sha1_hmac -K legaldocs0507 docs/*
Enter pin for Sun Software PKCS#11 softtoken:     Type password
```

키 저장소에서 MAC을 검색하려면 사용자가 상세 정보 표시 옵션을 사용하고 키 레이블과 인증된 디렉토리의 이름을 제공합니다.

```
% mac -v -a sha1_hmac -K legaldocs0507  docs/*
Enter pin for Sun Software PKCS#11 softtoken:     Type password
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

▼ 파일을 암호화 및 해독하는 방법

파일을 암호화할 때 원본 파일은 제거되거나 변경되지 않습니다. 출력 파일이 암호화됩니다.

`encrypt` 명령과 관련된 공통적인 오류 해결 방법은 예 다음에 오는 절을 참조하십시오.

참고 - 파일을 암호화하고 해독할 때 가능한 FIPS 승인 알고리즘을 승인된 키 길이와 함께 사용하십시오. “Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ”의 “FIPS 140 Algorithms in the Cryptographic Framework”에서 목록을 참조하십시오. `encrypt -l` 명령을 실행하여 사용 가능한 알고리즘과 해당 키 길이를 확인합니다.

---

1. **적절한 길이의 대칭 키를 만듭니다.**

   키 생성에 사용하거나 키를 제공할 수 있는 문장암호를 제공할 수 있습니다.

   - 문장암호를 제공하는 경우 문장암호를 저장하거나 기억해야 합니다. 문장암호를 온라인으로 저장할 경우 본인만 문장암호 파일을 읽을 수 있어야 합니다.
   - 키를 제공하는 경우 방식에 맞는 올바른 크기여야 합니다. `pktool` 명령을 사용할 수 있습니다. 절차 및 일부 예제는 `pktool` 명령을 사용하여 대칭 키를 생성하는 방법 [22]을 참조하십시오.

2. **파일을 암호화합니다.**

   `encrypt` 명령으로 키를 제공하고 대칭 키 알고리즘을 사용합니다.

   ```
   % encrypt -a algorithm [-v] \
   [-k keyfile | -K key-label [-T token]] [-i input-file] [-o output-file]
   ```

`-a` *algorithm*	파일을 암호화하는 데 사용할 알고리즘입니다. `encrypt -l` 명령의 출력에 나타난 대로 알고리즘을 입력합니다. 가능하면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에서 목록별 FIPS 승인 알고리즘을 선택하십시오.
`-k` *keyfile*	알고리즘이 지정된 길이의 키를 포함하는 파일입니다. `encrypt -l` 명령의 출력에 각 알고리즘의 키 길이가 비트 단위로 나열됩니다.
`-K` *key-label*	PKCS #11 키 저장소에서 키의 레이블입니다.
`-T` *token*	토큰 이름입니다. 기본적으로 토큰은 `Sun Software PKCS#11 softtoken`입니다. `-K` *key-label* 옵션을 사용할 때만 사용됩니다.
`-i` *input-file*	암호화하려는 입력 파일입니다. 이 파일은 명령에 의해 바뀌지 않습니다.
`-o` *output-file*	입력 파일의 암호화된 형태인 출력 파일입니다.

예 3-8 파일을 암호화하기 위한 AES 키 만들기

다음 예에서 사용자가 암호화 및 해독에 사용할 AES 키를 만들고 기존 PKCS #11 키 저장소에 저장합니다. 키가 존재하는지 확인하고 키를 사용할 수 있지만, 키 자체를 볼 수는 없습니다.

```
% pktool genkey label=MyAESkeynumber1 keytype=aes keylen=256
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-----------------------------------------------------
```

```
Symmetric keys:
1        AES           256           MyAESkeynumber1
```

키를 사용하여 파일을 암호화하려면 레이블로 키를 검색합니다.

```
% encrypt -a aes -K MyAESkeynumber1 -i encryptthisfile -o encryptedthisfile
```

encryptedthisfile 파일을 해독하려면 레이블로 키를 검색합니다.

```
% decrypt -a aes -K MyAESkeynumber1 -i encryptedthisfile -o sameasencryptthisfile
```

예 3-9 AES 및 문장암호로 암호화 및 해독

다음 예에서는 AES 알고리즘으로 파일이 암호화됩니다. 키가 문장암호에서 생성됩니다. 문장암호가 파일에 저장된 경우 사용자 이외의 다른 사람이 파일을 읽을 수 없어야 합니다.

```
% encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
Enter passphrase:         Type passphrase
Re-enter passphrase:      Type passphrase again
```

입력 파일 ticket.to.ride가 여전히 원본 형태로 존재합니다.

출력 파일을 해독하려면 파일을 암호화한 것과 동일한 문장암호 및 암호화 방식을 사용합니다.

```
% decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
Enter passphrase:      Type passphrase
```

예 3-10 AES 및 키 파일로 암호화 및 해독

다음 예에서 AES 알고리즘으로 파일이 암호화됩니다. AES 방식은 128비트 또는 16바이트 키를 사용합니다.

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \
-i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

입력 파일 ticket.to.ride가 여전히 원본 형태로 존재합니다.

출력 파일을 해독하려면 파일을 암호화한 것과 동일한 키 및 암호화 방식을 사용합니다.

```
% decrypt -a aes -k ~/keyf/05.07.aes16  \
-i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

일반 오류 다음 메시지는 encrypt 명령에 제공한 키가 사용 중인 알고리즘에서 허가되지 않음을 나타냅니다.

- encrypt: unable to create key for crypto operation:(암호화 작업에 대한 키를 작성할 수 없습니다.) CKR_ATTRIBUTE_VALUE_INVALID

- `encrypt: failed to initialize crypto operation:`(암호화 작업을 초기화하지 못했습니다.) `CKR_KEY_SIZE_RANGE`

알고리즘에 요구 사항에 맞지 않는 키를 전달할 경우 다음 방법 중 하나를 사용하여 더 적합한 키를 제공해야 합니다.

- 문장암호를 사용합니다. 그러면 프레임워크가 요구 사항을 충족하는 키를 제공합니다.
- 알고리즘에 적합한 키 크기를 전달합니다. 예를 들어, DES 알고리즘에 64비트 키가 필요합니다. 3DES 알고리즘에 192비트 키가 필요합니다.

암호화 프레임워크 관리

이 절에서는 암호화 프레임워크에서 소프트웨어 공급자 및 하드웨어 공급자를 관리하는 방법을 설명합니다. 원하는 경우 소프트웨어 공급자 및 하드웨어 공급자를 사용에서 제할 수 있습니다. 예를 들어, 한 소프트웨어 공급자에서 알고리즘 구현을 사용 안함으로 설정할 수 있습니다. 그런 다음, 다른 소프트웨어 공급자에서 알고리즘을 사용하도록 시스템에 강제 적용할 수 있습니다.

참고 - 암호화 프레임워크 관리의 중요한 구성 요소는, 암호화 모듈에 대한 미국 정부 컴퓨터 보안 표준인 FIPS 140에 관한 정책을 계획하고 구현하는 것입니다.

FIPS 140-2에서 검증한 암호화만 사용하도록 엄격한 요구 사항을 적용할 경우 Oracle Solaris11.1 SRU5.5 릴리스나 Oracle Solaris11.1 SRU3 릴리스를 실행 중이어야 합니다. Oracle은 이러한 두 가지 릴리스에서 Solaris 암호화 프레임워크에 대해 FIPS 140-2 검증을 마쳤습니다. Oracle Solaris11.2는 이러한 검증된 기반 위에 구축되어 성능, 기능성, 안정성을 보장하는 향상된 소프트웨어 기능을 제공합니다. 이러한 향상된 기능을 활용하려면 가능한 Oracle Solaris11.2를 FIPS 140-2 모드로 구성해야 합니다.

"Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "을 검토하고 시스템의 전반적인 FIPS 정책을 계획합니다.

다음 작업 맵은 암호화 프레임워크에서 소프트웨어 및 하드웨어 공급자를 관리하기 위한 절차를 가리킵니다.

표 3-2 암호화 프레임워크 작업 맵 관리

작업	설명	지침
시스템의 FIPS 정책을 계획합니다.	FIPS 승인 공급자 및 소비자를 사용으로 설정하기 위한 계획을 결정하고 해당 계획을 구현합니다.	"Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "
암호화 프레임워크에서 공급자를 나열합니다.	암호화 프레임워크에서 사용할 수 있는 알고리즘, 라이브러리 및 하드웨어 장치를 나열합니다.	"사용 가능한 공급자 나열" [34]
FIPS 140 모드를 사용으로 설정합니다.	암호화 모듈에 대한 미국 정보 표준에 따라 암호화 프레임워크를 실행합니다.	FIPS 140이 사용으로 설정된 부트 환경을 만드는 방법 [41]

작업	설명	지침
소프트웨어 공급자를 추가합니다.	PKCS #11 라이브러리 또는 커널 모듈을 암호화 프레임워크에 추가합니다. 공급자에 서명해야 합니다.	소프트웨어 공급자를 추가하는 방법 [39]
사용자 레벨 방식의 사용을 금지합니다.	소프트웨어 방식을 사용에서 제합니다. 방식을 다시 사용으로 설정할 수 있습니다.	사용자 레벨 방식의 사용을 금지하는 방법 [43]
커널 모듈에서 방식을 일시적으로 사용 안함으로 설정합니다.	방식을 일시적으로 사용에서 제합니다. 대개 테스트에 사용됩니다.	커널 소프트웨어 방식의 사용을 금지하는 방법 [45]
라이브러리를 제거합니다.	사용자 레벨 소프트웨어 공급자를 사용에서 제합니다.	예 3-17. “영구적으로 사용자 레벨 라이브러리 제거”
커널 공급자를 제거합니다.	커널 소프트웨어 공급자를 사용에서 제합니다.	예 3-19. “커널 소프트웨어 공급자 가용성을 일시적으로 제거”
하드웨어 공급자에서 방식을 사용 안함으로 설정합니다.	하드웨어 가속기에서 선택한 방식이 사용되지 않는지 확인합니다.	하드웨어 공급자 방식 및 기능을 사용 안함으로 설정하는 방법 [47]
암호화 서비스를 다시 시작하거나 새로 고칩니다.	암호화 서비스가 사용 가능한지 확인합니다.	모든 암호화 서비스를 새로 고치거나 다시 시작하는 방법 [49]

사용 가능한 공급자 나열

하드웨어 공급자는 자동으로 검색되어 로드됩니다. 자세한 내용은 `driver.conf`(4) 매뉴얼 페이지를 참조하십시오.

암호화 프레임워크에 연결해야 할 하드웨어가 있을 경우 하드웨어가 커널에서 SPI에 등록됩니다. 프레임워크가 하드웨어 드라이버를 서명했는지 확인합니다. 특히, Oracle이 발행한 인증서로 드라이버의 객체 파일이 서명되었는지 검사됩니다.

예를 들어, Sun Crypto Accelerator 6000 보드(`mca`), UltraSPARC T1 및 T2 프로세서의 암호화 가속기용 `ncp` 드라이버(`ncp`), UltraSPARC T2 프로세서의 n2cp 드라이버(`n2cp`), T-Series 시스템용 `/dev/crypto` 드라이버가 하드웨어 방식을 프레임워크에 연결합니다.

공급자 서명 얻기에 대한 자세한 내용은 “암호화 프레임워크의 사용자 레벨 명령” [11]에서 `elfsign` 명령에 대한 내용을 참조하십시오.

사용 가능한 공급자를 나열하려면 원하는 특정 정보에 따라 여러 옵션과 함께 `cryptoadm list` 명령을 사용합니다.

- 시스템의 모든 공급자 나열

  공급자 목록의 내용 및 형식은 여러 Oracle Solaris 릴리스 및 플랫폼마다 다릅니다. 시스템에서 `cryptoadm list` 명령을 실행하여 시스템이 지원하는 공급자를 확인하십시오. 일반 사용자는 사용자 레벨의 방식만 직접 사용할 수 있습니다.

  ```
  % cryptoadm list
  ```

```
User-level providers:       /* for applications */
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Provider: /usr/lib/security/$ISA/pkcs11_tpm.so

Kernel software providers:       /* for IPsec, kssl, Kerberos */
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0       /* for hardware */
ncp/0
n2cp/0
```

- 암호화 프레임워크에서 공급자 및 해당 방식 나열

  사용 가능한 방식의 강도 및 모드(예: ECB 및 CBC)를 볼 수 있습니다. 그러나 나열된 방식 중 일부는 사용하지 못할 수 있습니다. 사용할 수 있는 방식을 나열하는 방법에 대한 지침은 다음 항목을 참조하십시오.

  표시 목적상 다음 출력이 잘립니다.

```
% cryptoadm list -m [provider=provider]
User-level providers:
=====================

Provider: /usr/lib/security/$ISA/pkcs11_kernel.so

Mechanisms:
CKM_DSA
CKM_RSA_X_509
CKM_RSA_PKCS
...
CKM_SHA256_HMAC_GENERAL
CKM_SSL3_MD5_MAC

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC
CKM_DES_CBC_PAD
```

```
CKM_DES_ECB
CKM_DES_KEY_GEN
CKM_DES_MAC_GENERAL
...
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE

Provider: /usr/lib/security/$ISA/pkcs11_tpm.so
/usr/lib/security/$ISA/pkcs11_tpm.so: no slots presented.

Kernel providers:
==========================
des: CKM_DES_ECB,CKM_DES_CBC,CKM_DES3_ECB,CKM_DES3_CBC
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM, \
     CKM_AES_GCM,CKM_AES_GMAC,
CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
arcfour: CKM_RC4
blowfish: CKM_BLOWFISH_ECB,CKM_BLOWFISH_CBC
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA, \
     CKM_ECDSA_SHA1
sha1: CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL
sha2: CKM_SHA224,CKM_SHA224_HMAC,...CKM_SHA512_256_HMAC_GENERAL

md4: CKM_MD4
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS,CKM_RSA_X_509,CKM_MD5_RSA_PKCS, \
     CKM_SHA1_RSA_PKCS,CKM_SHA224_RSA_PKCS,
CKM_SHA256_RSA_PKCS,CKM_SHA384_RSA_PKCS,CKM_SHA512_RSA_PKCS
swrand: No mechanisms presented.
n2rng/0: No mechanisms presented.
ncp/0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS,CKM_RSA_PKCS_KEY_PAIR_GEN,
CKM_DH_PKCS_KEY_PAIR_GEN,CKM_DH_PKCS_DERIVE,CKM_EC_KEY_PAIR_GEN,
CKM_ECDH1_DERIVE,CKM_ECDSA
n2cp/0: CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES3_CBC, \
     ...CKM_SSL3_SHA1_MAC
```

- 사용 가능한 암호화 방식 나열

  정책에 따라 사용 가능한 방식이 결정됩니다. 관리자가 정책을 설정합니다. 관리자는 특정 공급자의 방식을 사용 안함으로 설정하도록 선택할 수 있습니다. `-p` 옵션은 관리자가 설정한 정책에 의해 허가된 방식 목록을 표시합니다.

  ```
  % cryptoadm list -p [provider=provider]
  User-level providers:
  =====================
  /usr/lib/security/$ISA/pkcs11_kernel.so: \
       all mechanisms are enabled.random is enabled.
  /usr/lib/security/$ISA/pkcs11_softtoken.so: \
  ```

```
     all mechanisms are enabled, random is enabled.
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.

Kernel providers:
=========================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
ecc: all mechanisms are enabled.
sha1: all mechanisms are enabled.
sha2: all mechanisms are enabled.
md4: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.
n2rng/0: all mechanisms are enabled. random is enabled.
ncp/0: all mechanisms are enabled.
n2cp/0: all mechanisms are enabled.
```

다음 예에서는 `cryptoadm list` 명령의 구체적인 사용을 추가로 보여줍니다.

예 3-11 특정 공급자의 암호화 정보 나열

`cryptoadm` *options* 명령에 공급자를 지정하면 공급자에게 해당되는 정보로만 출력이 제한됩니다.

```
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5, CKM_MD5_HMAC,...
random is enabled.
```

다음 출력에서는 방식만 사용으로 설정되었음을 보여줍니다. 난수 생성기는 계속 사용 안함으로 설정됩니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....

# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

다음 출력에서는 보드의 모든 기능과 방식이 사용으로 설정되었음을 보여줍니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms ar enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.

# cryptoadm enable provider=dca/0 all
```

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

예 3-12 사용자 레벨 암호화 방식만 찾기

다음 예에서 사용자 레벨 라이브러리 pkcs11_softtoken이 제공하는 모든 방식이 나열됩니다.

```
% cryptoadm list -m provider=/usr/lib/security/\
   $ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC
CKM_DES_CBC_PAD
CKM_DES_ECB
CKM_DES_KEY_GEN
CKM_DES_MAC_GENERAL
CKM_DES_MAC
…
CKM_ECDSA
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE
```

예 3-13 어떤 암호화 방식이 어떤 기능을 수행하는지 확인

방식은 서명 또는 키 생성과 같은 특정 암호화 기능을 수행합니다. -v -m 옵션은 모든 방식과 해당 기능을 표시합니다.

이 경우 관리자가 CKM_ECDSA* 방식이 사용될 수 있는 기능이 무엇인지 확인할 수 있습니다.

```
% cryptoadm list -vm
User-level providers:
=====================
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Number of slots: 3
Slot #2
Description: ncp/0 Crypto Accel Asym 1.0
...
CKM_ECDSA                   163  571  X  .  .  .  X  .  X  .  .  .  .  .  .  .
...

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
...
CKM_ECDSA       112 571  .  .  .  .  X  .  X  .  .  .  .  .  .  .  .
CKM_ECDSA_SHA1  112 571  .  .  .  .  X  .  X  .  .  .  .  .  .  .  .
...
Kernel providers:
=================
...
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA,CKM_ECDSA_SHA1
...
```

목록에는 다음 사용자 레벨 공급자에서 제공되는 방식이 표시됩니다.

- CKM_ECDSA 및 CKM_ECDSA_SHA1 - /usr/lib/security/$ISA/pkcs11_softtoken.so 라이브러리의 소프트웨어 구현
- CKM_ECDSA - /usr/lib/security/$ISA/pkcs11_kernel.so 라이브러리의 ncp/0 Crypto Accel Asym 1.0으로 가속화됨

각 항목은 방식에 대한 정보 조각을 나타냅니다. 이러한 ECC 방식에 나타나는 목록은 다음과 같습니다.

- 최소 길이 - 112바이트
- 최대 길이 - 571바이트
- 하드웨어 - 하드웨어에 사용하거나 사용할 수 없습니다.
- 암호화 - 데이터를 암호화하는 데 사용되지 않습니다.
- 해독 - 데이터를 해독하는 데 사용되지 않습니다.
- 다이제스트 - 메시지 다이제스트를 만드는 데 사용되지 않습니다.
- 서명 - 데이터를 서명하는 데 사용됩니다.
- 서명 + 복구 - 데이터를 서명으로부터 복구할 수 있는 경우 데이터를 서명하는 데 사용되지 않습니다.
- 확인 - 서명된 데이터를 확인하는 데 사용됩니다.
- 확인 + 복구 - 서명으로부터 복구할 수 있는 데이터를 확인하는 데 사용되지 않습니다.
- 키 생성 - 개인 키를 생성하는 데 사용되지 않습니다.
- 쌍 생성 - 키 쌍을 생성하는 데 사용되지 않습니다.
- 래핑 - 래핑하는 데 사용되지 않습니다. 즉, 기존 키를 암호화합니다.
- 언래핑 - 래핑된 키를 언래핑하는 데 사용되지 않습니다.
- 파생 - 기본 키로부터 새 키를 파생하는 데 사용되지 않습니다.
- EC Caps - 이전 항목에 포함되지 않은 Absent EC 기능

소프트웨어 공급자 추가

다음 절차에서는 시스템에 공급자를 추가하는 방법을 설명합니다. Crypto Management 권한 프로파일이 지정된 관리자여야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안 ”의 “지정된 관리 권한 사용”을 참조하십시오.

▼ 소프트웨어 공급자를 추가하는 방법

1. **시스템에 사용 가능한 소프트웨어 공급자를 나열합니다.**

```
% cryptoadm list
User-level providers:
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.
```

```
Kernel software providers:
des
aes
arcfour
blowfish
camellia
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

2. **저장소에서 공급자를 추가합니다.**
   기존 공급자 소프트웨어는 Oracle에서 인증서를 발행했습니다.

3. **공급자를 새로 고칩니다.**
   소프트웨어 공급자를 추가한 경우 또는 하드웨어와 지정된 정책을 추가한 경우 공급자를 새로 고쳐야 합니다.

   ```
   # svcadm refresh svc:/system/cryptosvc
   ```

4. **목록에서 새 공급자를 찾습니다.**
   이 경우 새 커널 소프트웨어 공급자가 설치되었습니다.

   ```
   # cryptoadm list
   …
   Kernel software providers:
   des
   aes
   arcfour
   blowfish
   camellia
   ecc
   sha1
   sha2
   md4
   md5
   rsa
   swrand
   sha3       <-- added provider
   …
   ```

예 3-14 사용자 레벨 소프트웨어 공급자 추가

다음 예에서 서명된 PKCS #11 라이브러리가 설치됩니다.

```
# pkgadd -d /cdrom/cdrom0/PKCSNew
```

Answer the prompts

```
# svcadm refresh system/cryptosvc
# cryptoadm list
user-level providers:
=========================
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so
/opt/lib/$ISA/libpkcs11.so.1      <-- added provider
```

암호화 프레임워크로 라이브러리를 테스트 중인 개발자가 수동으로 라이브러리를 설치할 수 있습니다.

```
# cryptoadm install provider=/opt/lib/\$ISA/libpkcs11.so.1
```

FIPS 140이 사용으로 설정된 부트 환경 만들기

기본적으로 FIPS 140 모드는 Oracle Solaris에서 사용 안함으로 설정됩니다. 이 절차에서는 FIPS 140 모드에 대해 새로운 BE(부트 환경)를 만든 후 FIPS 140을 사용으로 설정하고 새 BE로 부트합니다. 백업 BE가 주어진 경우 이 방식을 사용하면 FIPS 140 준수 테스트로부터 발생할 수 있는 시스템 패닉을 빠르게 복구할 수 있습니다.

FIPS에 대한 개요는 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "을 참조하십시오. 또한 `cryptoadm`(1M) 매뉴얼 페이지와 "암호화 프레임워크 및 FIPS 140" [13]을 참조하십시오.

▼ FIPS 140이 사용으로 설정된 부트 환경을 만드는 방법

시작하기 전에 `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "의 "지정된 관리 권한 사용"을 참조하십시오.

1. 시스템이 FIPS 140 모드인지 확인합니다.

```
% cryptoadm list fips-140
User-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_softtoken: FIPS-140 mode is disabled.

Kernel software providers:
==========================
des: FIPS-140 mode is disabled.
aes: FIPS-140 mode is disabled.
ecc: FIPS-140 mode is disabled.
sha1: FIPS-140 mode is disabled.
sha2: FIPS-140 mode is disabled.
rsa: FIPS-140 mode is disabled.
swrand: FIPS-140 mode is disabled.
```

```
Kernel hardware providers:
========================:
```

2. **암호화 프레임워크의 FIPS 140 버전에 대한 새 BE를 만듭니다.**

   FIPS 140 모드를 사용으로 설정하기 전에 `beadm` 명령을 사용하여 새 BE를 만들고, 활성화하고, 부트해야 합니다. FIPS 140 지원 시스템은 실패할 경우 패닉을 일으킬 수 있는 준수 테스트를 실행합니다. 따라서 FIPS 140 경계 문제를 디버그할 때 시스템을 작동 및 실행하기 위해 부트할 수 있는 사용 가능한 BE가 있어야 합니다.

   a. **현재 BE를 기반으로 BE를 만듭니다.**

      이 예에서는 `S11.1-FIPS`라는 이름의 BE를 만듭니다.

      ```
      # beadm create S11.1-FIPS-140
      ```

   b. **해당 BE를 활성화합니다.**

      ```
      # beadm activate S11.1-FIPS-140
      ```

   c. **시스템을 재부트합니다.**

   d. **새 BE에서 FIPS 140 모드를 사용으로 설정합니다.**

      ```
      # cryptoadm enable fips-140
      ```

참고 - 이 하위 명령은 사용자 레벨의 `pkcs11_softtoken` 라이브러리 및 커널 소프트웨어 공급자로부터의 비FIPS 140 승인 알고리즘을 사용 안함으로 설정하지 않습니다. 프레임워크 소비자는 FIPS 140 승인 알고리즘만 사용해야 합니다.

FIPS 140 모드의 영향에 대한 자세한 내용은 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "을 참조하십시오. 또한 `cryptoadm`(1M) 매뉴얼 페이지를 참조하십시오.

---

3. **FIPS 140을 사용으로 설정하지 않고 실행하려는 경우 FIPS 140 모드를 사용 안함으로 설정합니다.**

   원래 BE로 재부트하거나 현재 BE에서 FIPS 140을 사용 안함으로 설정합니다.

   ■ **원래 BE로 부트합니다.**

   ```
   # beadm list
   BE              Active Mountpoint Space   Policy Created
   --              ------ ---------- -----   ------ -------
   S11.1           -      -          48.22G   static 2012-10-10 10:10
   S11.1-FIPS-140  NR     /          287.01M  static 2012-11-18 18:18
   # beadm activate S11.1
   # beadm list
   BE              Active Mountpoint Space   Policy Created
   --              ------ ---------- -----   ------ -------
   ```

```
S11.1            R      -          48.22G   static 2012-10-10 10:10
S11.1-FIPS-140   N      /          287.01M  static 2012-11-18 18:18
# reboot
```

■ **현재 BE에서 FIPS 140 모드를 사용 안함으로 설정하고 재부트합니다.**

```
# cryptoadm disable fips-140
```

FIPS 140 모드는 시스템이 재부트될 때까지 작동 상태로 유지됩니다.

```
# reboot
```

방식 사용 금지

라이브러리 공급자의 암호화 방식 중 일부를 사용하면 안 되는 경우 선택한 방식을 제거할 수 있습니다. 예를 들어, 다른 라이브러리에 있는 동일한 방식의 성능이 더 낫거나 보안 취약성을 조사 중인 경우 방식을 사용하지 않도록 설정할 수 있습니다.

암호화 프레임워크가 AES와 같은 여러 모드의 공급자를 제공하는 경우 느리거나 손상된 방식이 사용되지 않도록 제거할 수 있습니다. 이 절차에 따라 보안 취약성이 입증된 알고리즘을 제거할 수도 있습니다.

하드웨어 공급자에서 방식과 난수 기능을 선택적으로 사용 안함으로 설정할 수 있습니다. 다시 사용으로 설정하려면 예 3-22. “하드웨어 공급자에서 방식 및 기능을 사용으로 설정”를 참조하십시오. 이 예제의 하드웨어인 Sun Crypto Accelerator 1000 보드는 난수 생성기를 제공합니다.

▼ 사용자 레벨 방식의 사용을 금지하는 방법

시작하기 전에 Crypto Management 권한 프로파일이 지정된 관리자여야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안 ”의 “지정된 관리 권한 사용”을 참조하십시오.

1. **특정 사용자 레벨 소프트웨어 공급자가 제공한 방식을 나열합니다.**

```
% cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
…
```

2. **사용할 수 있는 방식을 나열합니다.**

```
$ cryptoadm list -p
user-level providers:
=====================
…
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
```

```
random is enabled.
…
```

3. **사용하면 안되는 방식을 사용 안함으로 설정합니다.**

```
$ cryptoadm disable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB
```

4. **사용할 수 있는 방식을 나열합니다.**

```
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

예 3-15 사용자 레벨 소프트웨어 공급자 방식을 사용으로 설정

다음 예에서 사용 안함으로 설정된 DES 방식을 다시 사용할 수 있도록 만듭니다.

```
$ cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
…
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_ECB
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

예 3-16 모든 사용자 레벨 소프트웨어 공급자 방식을 사용으로 설정

다음 예에서 사용자 레벨 라이브러리의 모든 방식이 사용으로 설정됩니다.

```
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so all
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
```

예 3-17 영구적으로 사용자 레벨 라이브러리 제거

다음 예에서는 `libpkcs11.so.1` 라이브러리가 `/opt` 디렉토리에서 제거됩니다.

```
$ cryptoadm uninstall provider=/opt/lib/\$ISA/libpkcs11.so.1
$ cryptoadm list
user-level providers:
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so
```

```
kernel providers:
…
```

▼ 커널 소프트웨어 방식의 사용을 금지하는 방법

시작하기 전에 Crypto Management 권한 프로파일이 지정된 관리자여야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "의 "지정된 관리 권한 사용"을 참조하십시오.

1. **특정 커널 소프트웨어 공급자가 제공한 방식을 나열합니다.**

```
$ cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,CKM_AES_GCM,
CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
```

2. **사용할 수 있는 방식을 나열합니다.**

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

3. **사용하면 안 되는 방식을 사용 안함으로 설정합니다.**

```
$ cryptoadm disable provider=aes mechanism=CKM_AES_ECB
```

4. **사용할 수 있는 방식을 나열합니다.**

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
```

예 3-18 커널 소프트웨어 공급자 방식을 사용으로 설정

다음 예에서 사용 안함으로 설정된 AES 방식을 다시 사용할 수 있도록 만듭니다.

```
cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,
CKM_AES_GCM,CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
$ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

예 3-19 커널 소프트웨어 공급자 가용성을 일시적으로 제거

다음 예에서는 AES 공급자가 사용되지 않도록 일시적으로 제거합니다. 설치를 제거하는 동안 공급자가 자동으로 로드되지 못하게 하려면 `unload` 하위 명령이 유용합니다. 예를 들어, `unload` 하위 명령은 이 공급자의 방식을 수정할 때 사용할 수 있습니다.

```
$ cryptoadm unload provider=aes

$ cryptoadm list
```

```
…
Kernel software providers:
des
aes (inactive)
arcfour
blowfish
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

암호화 프레임워크를 새로 고칠 때까지 AES 공급자를 사용할 수 없습니다.

```
$ svcadm refresh system/cryptosvc

$ cryptoadm list
…
Kernel software providers:
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

커널 소비자가 커널 소프트웨어 공급자를 사용 중인 경우 소프트웨어가 언로드되지 않습니다. 오류 메시지가 표시되고 공급자를 계속 사용할 수 있습니다.

예 3-20 소프트웨어 공급자 가용성을 영구적으로 제거

다음 예에서는 AES 공급자가 사용되지 않도록 제거합니다. 일단 제거된 AES 공급자는 커널 소프트웨어 공급자의 정책 목록에 나타나지 않습니다.

```
$ cryptoadm uninstall provider=aes

$ cryptoadm list
…
Kernel software providers:
des
```

```
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

커널 소비자가 커널 소프트웨어 공급자를 사용 중인 경우 오류 메시지가 표시되고 공급자를 계속 사용할 수 있습니다.

예 3-21 제거된 커널 소프트웨어 공급자 재설치

다음 예에서 AES 커널 소프트웨어 공급자가 재설치됩니다. 제거된 커널 공급자를 재설치하려면 설치할 방식을 열거해야 합니다.

```
$ cryptoadm install provider=aes \
mechanism=CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,
CKM_AES_GCM,CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC

$ cryptoadm list
…
Kernel software providers:
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

▼ 하드웨어 공급자 방식 및 기능을 사용 안함으로 설정하는 방법

시작하기 전에 Crypto Management 권한 프로파일이 지정된 관리자여야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안 ”의 “지정된 관리 권한 사용”을 참조하십시오.

- **사용하지 않을 방식 또는 기능을 선택합니다.**

하드웨어 공급자를 나열합니다.

```
# cryptoadm list
...
Kernel hardware providers:
dca/0
```

- **선택된 방식을 사용 안함으로 설정합니다.**

```
# cryptoadm list -m provider=dca/0
dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
random is enabled.
# cryptoadm disable provider=dca/0 mechanism=CKM_DES_CBC,CKM_DES3_CBC
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_CBC,CKM_DES3_CBC.
random is enabled.
```

- **난수 생성기를 사용 안함으로 설정합니다.**

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

- **모든 방식을 사용 안함으로 설정합니다. 난수 생성기는 사용 안함으로 설정하지 마십시오.**

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is enabled.
```

- **하드웨어에서 모든 기능 및 방식을 사용 안함으로 설정합니다.**

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is disabled.
```

예 3-22 하드웨어 공급자에서 방식 및 기능을 사용으로 설정

다음 예에서는 하드웨어에서 사용 안함으로 설정된 방식이 선택적으로 사용으로 설정됩니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB,CKM_DES3_ECB
```

.

```
random is enabled.
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB.
random is enabled.
```

다음 예에서 난수 생성기만 사용으로 설정됩니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is disabled.
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is enabled.
```

다음 예에서 방식만 사용으로 설정됩니다. 난수 생성기는 계속 사용 안함으로 설정됩니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is disabled.
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

다음 예에서 보드의 모든 기능 및 방식이 사용으로 설정됩니다.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

모든 암호화 서비스 새로 고침 또는 다시 시작

기본적으로 암호화 프레임워크는 사용으로 설정됩니다. 어떤 이유로 `kcfd` 데몬을 실패할 때 SMF(서비스 관리 기능)를 사용하여 암호화 서비스를 다시 시작할 수 있습니다. 자세한 내용은 `smf`(5) 및 `svcadm`(1M) 매뉴얼 페이지를 참조하십시오. 암호화 서비스 다시 시작이 영역에 미치는 영향은 “암호화 서비스 및 영역” [12]을 참조하십시오.

▼ 모든 암호화 서비스를 새로 고치거나 다시 시작하는 방법

시작하기 전에 Crypto Management 권한 프로파일이 지정된 관리자여야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안 ”의 “지정된 관리 권한 사용”을 참조하십시오.

1. **암호화 서비스의 상태를 확인합니다.**

```
% svcs cryptosvc
STATE          STIME    FMRI
offline         Dec_09   svc:/system/cryptosvc:default
```

2. **암호화 서비스를 사용으로 설정합니다.**

```
# svcadm enable svc:/system/cryptosvc
```

예 3-23 암호화 서비스 새로 고침

다음 예에서 암호화 서비스를 전역 영역에서 새로 고칩니다. 따라서 모든 비전역 영역의 커널 레벨 암호화 정책도 새로 고쳐집니다.

```
# svcadm refresh system/cryptosvc
```

♦ ♦ ♦ 4 장

4 키 관리 프레임워크

Oracle Solaris의 키 관리 프레임워크(Key Management Framework, KMF) 기능은 공개 키 객체 관리용 도구 및 프로그래밍 인터페이스를 제공합니다. 공개 키 객체는 X.509 인증서 및 공개/개인 키 쌍을 포함합니다. 이러한 객체의 저장 형식은 다양할 수 있습니다. 또한 KMF는 응용 프로그램의 X.509 인증서 사용을 정의하는 정책 관리용 도구를 제공합니다. KMF는 타사 플러그인을 지원합니다.

이 장에서는 다음 내용을 다룹니다.

공개 키 기술 관리

KMF는 PKI(공개 키 기술)를 중앙 집중적인 방식으로 관리합니다. Oracle Solaris에는 PKI 기술을 이용하는 다양한 응용 프로그램이 있습니다. 각 응용 프로그램은 고유의 프로그래밍 인터페이스, 키 저장소 방식, 관리 유틸리티를 제공합니다. 응용 프로그램이 정책 시행 방식을 제공하는 경우 방식이 해당 응용 프로그램에만 적용됩니다. KMF와 함께 응용 프로그램은 통일된 관리 도구 세트, 단일 프로그래밍 인터페이스 세트, 단일 정책 시행 방식을 사용합니다. 이러한 기능은 해당 인터페이스를 채택한 모든 응용 프로그램의 PKI 요구를 관리합니다.

KMF는 다음 인터페이스를 사용하여 공개 키 기술 관리를 통일합니다.

- `pktool` 명령 - 인증서와 같은 PKI 객체를 다양한 키 저장소에서 관리합니다.
- `kmfcfg` 명령 - PKI 정책 데이터베이스 및 타사 플러그인을 관리합니다.

  PKI 정책 결정은 검증 방법과 같은 작업을 포함합니다. 또한 PKI 정책은 인증서의 범위를 제한할 수 있습니다. 예를 들어, PKI 정책이 인증서를 특정 목적으로만 사용할 수 있는지 검증할 수 있습니다. 이러한 정책은 해당 인증서가 다른 요청에 사용되는 것을 금지합니다.
- KMF 라이브러리 - 기본 키 저장소 방식을 끌어내는 프로그래밍 인터페이스를 포함합니다.

  응용 프로그램이 하나의 특정 키 저장소 방식을 선택할 필요는 없지만, 한 방식에서 다른 방식으로 마이그레이션할 수 있습니다. 지원되는 키 저장소는 PKCS #11, NSS,

OpenSSL입니다. 라이브러리가 플러그인 가능한 프레임워크를 포함하므로 새 키 저장소 방식을 추가할 수 있습니다. 따라서 새 방식을 사용하는 응용 프로그램을 조금만 바꾸면 새 키 저장소를 사용할 수 있습니다.

키 관리 프레임워크 유틸리티

KMF는 키 저장소 관리 방법을 제공하고 이러한 키 사용에 대한 전체 정책을 제공합니다. KMF는 세 가지 공개 키 기술에 대한 정책, 키 및 인증서를 관리할 수 있습니다.

- PKCS #11 공급자(암호화 프레임워크)의 토큰
- NSS(Network Security Services)
- OpenSSL, 파일 기반 키 저장소

`kmfcfg` 도구는 KMF 정책 항목을 만들고 수정, 삭제할 수 있습니다. 또한 프레임워크에 대한 플러그인을 관리합니다. KMF는 `pktool` 명령을 통해 키 저장소를 관리합니다. 자세한 내용은 `kmfcfg`(1) 및 `pktool`(1) 매뉴얼 페이지와 다음 절을 참조하십시오.

KMF 정책 관리

KMF 정책은 데이터베이스에 저장됩니다. 이 정책 데이터베이스는 KMF 프로그래밍 인터페이스를 사용하는 모든 응용 프로그램에서 내부적으로 액세스할 수 있습니다. 데이터베이스는 KMF 라이브러리에서 관리되는 키 및 인증서의 사용을 제약할 수 있습니다. 응용 프로그램이 인증서를 확인하려고 시도할 때 정책 데이터베이스를 검사합니다. `kmfcfg` 명령은 정책 데이터베이스를 수정합니다.

KMF 플러그인 관리

`kmfcfg` 명령은 다음과 같은 플러그인의 하위 명령을 제공합니다.

- `list plugin` - KMF에서 관리되는 플러그인을 나열합니다.
- `install` *plugin* - 모듈의 경로 이름으로 플러그인을 설치하고 플러그인에 대한 키 저장소를 만듭니다. KMF에서 플러그인을 제거하려면 키 저장소를 제거합니다.
- `uninstall` *plugin* - 키 저장소를 제거하여 KMF에서 플러그인을 제거합니다.
- `modify` *plugin* - `debug`와 같은 플러그인 코드에 저장된 옵션과 함께 플러그인이 실행되도록 합니다.

자세한 내용은 `kmfcfg`(1) 매뉴얼 페이지를 참조하십시오. 절차는 KMF에서 타사 플러그인을 관리하는 방법 [65]을 참조하십시오.

KMF 키 저장소 관리

KMF는 세 가지 공개 키 기술인 PKCS #11 토큰, NSS 및 OpenSSL에 대한 키 저장소를 관리합니다. 이러한 기술은 모두 `pktool` 명령으로 다음 작업을 수행할 수 있습니다.

- 자체 서명된 인증서를 생성합니다
- 인증서 요청을 생성합니다
- 대칭 키를 생성합니다
- 공개/개인 키 쌍을 생성합니다
- 서명할 외부 인증 기관(CA)에 보낼 PKCS #10 인증서 서명 요청(CSR)을 생성합니다
- PKCS #10 CSR을 서명합니다
- 객체를 키 저장소로 가져옵니다
- 키 저장소의 객체를 나열합니다
- 키 저장소에서 객체를 삭제합니다
- CRL을 다운로드합니다.

PKCS #11 및 NSS 기술의 경우 `pktool` 명령으로 키 저장소나 키 저장소의 객체에 대한 문장암호를 생성하여 PIN을 설정할 수도 있습니다.

`pktool` 유틸리티 사용의 예는 `pktool`(1) 매뉴얼 페이지 및 표 4-1. “키 관리 프레임워크 작업 맵 사용”을 참조하십시오.

키 관리 프레임워크 사용

이 절은 `pktool` 명령을 사용하여 암호, 문장암호, 파일, 키 저장소, 인증서, CRL과 같은 공개 키 객체를 관리하는 방법을 설명합니다.

키 관리 프레임워크(KMF)를 통해 중앙에서 공개 키 기술을 관리할 수 있습니다.

표 4-1 키 관리 프레임워크 작업 맵 사용

작업	설명	지침
인증서를 만듭니다.	PKCS #11, NSS 또는 OpenSSL에서 사용할 인증서를 만듭니다.	`pktool gencert` 명령을 사용하여 인증서를 만드는 방법 [54]
인증서를 내보냅니다.	인증서와 해당 지원 키가 포함된 파일을 만듭니다. 파일은 암호로 보호할 수 있습니다.	PKCS #12 형식의 인증서 및 개인 키를 내보내는 방법 [57]
인증서를 가져옵니다.	다른 시스템에서 인증서를 가져옵니다.	인증서를 키 저장소로 가져오는 방법 [56]
	다른 시스템에서 PKCS #12 형식의 인증서를 가져옵니다.	예 4-2. “PKCS #12 파일을 키 저장소로 가져오기”
문장암호를 생성합니다.	PKCS #11 키 저장소나 NSS 키 저장소에 액세스하기 위한 문장암호를 생성합니다.	`pktool setpin` 명령을 사용하여 암호문을 생성하는 방법 [58]

작업	설명	지침
대칭 키를 생성합니다.	파일 암호화, 파일의 MAC 만들기 및 응용 프로그램에 사용할 대칭 키를 생성합니다.	pktool 명령을 사용하여 대칭 키를 생성하는 방법 [22]
키 쌍을 생성합니다.	응용 프로그램에 사용할 공개/개인 키 쌍을 생성합니다.	pktool genkeypair 명령을 사용하여 키 쌍을 생성하는 방법 [60]
PKCS #10 CSR을 생성합니다.	서명할 외부 인증 기관(CA)을 위한 PKCS #10 인증서 서명 요청(CSR)을 생성합니다.	pktool(1) 매뉴얼 페이지
PKCS #10 CSR을 서명합니다.	PKCS #10 CSR을 서명합니다.	pktool signcsr 명령을 사용하여 인증서 요청을 서명하는 방법 [64]
KMF에 플러그인을 추가합니다.	플러그인을 설치, 수정, 나열합니다. 또한 KMF에서 플러그인을 제거합니다.	KMF에서 타사 플러그인을 관리하는 방법 [65]

▼ pktool gencert 명령을 사용하여 인증서를 만드는 방법

이 절차는 자체 서명된 인증서를 만들고 PKCS #11 키 저장소에 인증서를 저장합니다. 이 작업의 일부로 RSA 공개/개인 키 쌍도 생성됩니다. 개인 키는 인증서와 함께 키 저장소에 저장됩니다.

1. **자체 서명된 인증서를 생성합니다.**

```
% pktool gencert [keystore=keystore] label=label-name \
subject=subject-DN serial=hex-serial-number keytype=rsa/dsa keylen=key-size
```

`keystore=`*keystore* 공개 키 객체의 유형별로 키 저장소를 지정합니다. 값은 `nss`, `pkcs11` 또는 `file`일 수 있습니다. 이 키워드는 선택 사항입니다.

`label=`*label-name* 발행자가 인증서에 제공하는 고유한 이름을 지정합니다.

`subject=`*subject-DN* 인증서에 대한 식별 이름을 지정합니다.

`serial=`*hex-serial-number* 16진수 형식의 일련 번호를 지정합니다. 인증서의 발행자가 `0x0102030405`와 같은 숫자를 선택합니다.

`keytype=`*key type* 인증서와 연관된 개인 키의 유형을 지정하는 선택적 변수입니다. 선택한 키 저장소에 사용 가능한 키 유형을 찾으려면 `pktool`(1) 매뉴얼 페이지를 확인하십시오.

FIPS 140 승인 키를 사용하려면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에서 승인된 키 유형을 확인하십시오.

`keylen=`*key size* 인증서와 연관된 개인 키의 길이를 지정하는 선택적 변수입니다.

FIPS 140 승인 키를 사용하려면 "Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "의 "FIPS 140 Algorithms in the Cryptographic Framework"에서 선택한 키 유형에 대해 승인된 키 길이를 확인하십시오.

2. **키 저장소의 내용을 확인합니다.**

```
% pktool list
Found number certificates.
1. (X.509 certificate)
Label:  label-name
ID: fingerprint that binds certificate to private key
Subject: subject-DN
Issuer:  distinguished-name
Serial:  hex-serial-number
n. ...
```

이 명령은 키 저장소의 모든 인증서를 나열합니다. 다음 예에서 키 저장소는 하나의 인증서만 포함합니다.

예 4-1 pktool을 사용하여 자체 서명된 인증서 만들기

다음 예에서 My Company의 사용자가 자체 서명된 인증서를 만들고 PKCS #11 객체용 키 저장소에 인증서를 저장합니다. 키 저장소는 처음에 비어 있습니다. 키 저장소가 초기화되지 않은 경우 softtoken의 PIN이 changeme이고 pktool setpin 명령을 사용하여 PIN을 재설정할 수 있습니다. FIPS에서 승인한 키 유형 및 키 길이로 RSA 2048이 명령 옵션에 지정됩니다.

```
% pktool gencert keystore=pkcs11 label="My Cert" \
subject="C=US, O=My Company, OU=Security Engineering Group, CN=MyCA" \
serial=0x000000001 keytype=rsa keylen=2048
Enter pin for Sun Software PKCS#11 softtoken:     Type PIN for token

% pktool list
No.  Key Type  Key Len.  Key Label
-----------------------------------------------------
Asymmetric public keys:
1    RSA                My Cert
Certificates:
1    X.509 certificate
Label: My Cert
ID: d2:7e:20:04:a5:66:e6:31:90:d8:53:28:bc:ef:55:55:dc:a3:69:93
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
...
...
Serial: 0x00000010
...
```

▼ 인증서를 키 저장소로 가져오는 방법

이 절차는 PEM 또는 원시 DER로 인코딩된 PKI 정보가 담긴 파일을 키 저장소로 가져오는 방법을 설명합니다. 내보내기 절차는 예 4-4. “PKCS #12 형식의 인증서 및 개인 키 내보내기”를 참조하십시오.

1. **인증서를 가져옵니다.**

   ```
   % pktool import keystore=keystore infile=infile-name label=label-name
   ```

2. **개인 PKI 객체를 가져오는 경우 프롬프트가 표시되면 암호를 제공합니다.**

   a. **프롬프트에 파일의 암호를 입력합니다.**

   PKCS #12 형식의 내보내기 파일과 같은 개인 PKI 정보를 가져오는 경우 파일에 암호가 필요합니다. 가져오기 중인 파일의 작성자가 사용자에게 PKCS #12 암호를 알려줍니다.

   ```
   Enter password to use for accessing the PKCS12 file:      Type PKCS #12 password
   ```

   b. **프롬프트에 키 저장소의 암호를 입력합니다.**

   ```
   Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
   ```

3. **키 저장소의 내용을 확인합니다.**

   ```
   % pktool list
   Found number certificates.
   1. (X.509 certificate)
   Label:  label-name
   ID:  fingerprint that binds certificate to private key
   Subject:  subject-DN
   Issuer:  distinguished-name
   Serial:  hex-serial-number

   2. ...
   ```

예 4-2 PKCS #12 파일을 키 저장소로 가져오기

다음 예에서 사용자가 PKCS #12 파일을 타사로부터 가져옵니다. `pktool import` 명령은 `gracedata.p12` 파일에서 개인 키 및 인증서를 추출하여 사용자의 선호 키 저장소에 저장합니다.

```
% pktool import keystore=pkcs11 infile=gracedata.p12 label=GraceCert
Enter password to use for accessing the PKCS12 file:      Type PKCS #12 password
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Found 1 certificate(s) and 1 key(s) in gracedata.p12
% pktool list
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
```

```
Asymmetric public keys:
1    RSA                GraceCert
Certificates:
1    X.509 certificate
Label: GraceCert
ID: 71:8f:11:f5:62:10:35:c2:5d:b4:31:38:96:04:80:25:2e:ad:71:b3
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Serial: 0x00000010
```

예 4-3 X.509 인증서를 키 저장소로 가져오기

다음 예에서 사용자가 PEM 형식의 X.509 인증서를 사용자의 선호 키 저장소로 가져옵니다. 이 공개 인증서는 암호로 보호되지 않습니다. 사용자의 공개 키 저장소도 암호로 보호되지 않습니다.

```
% pktool import keystore=pkcs11 infile=somecert.pem label="TheirCompany Root Cert"
% pktool list
No.  Key Type  Key Len.  Key Label
Certificates:
1    X.509 certificate
Label: TheirCompany Root Cert
ID: ec:a2:58:af:83:b9:30:9d:de:b2:06:62:46:a7:34:49:f1:39:00:0e
Subject: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
Issuer: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
Serial: 0x00000001
```

▼ PKCS #12 형식의 인증서 및 개인 키를 내보내는 방법

PKCS #12 형식의 파일을 만들어서 개인 키 및 연관된 X.509 인증서를 다른 시스템으로 내보낼 수 있습니다. 파일 액세스는 암호로 보호됩니다.

1. **내보낼 인증서를 찾습니다.**

```
% pktool list
Found number certificates.
1. (X.509 certificate)
Label:  label-name
ID: fingerprint that binds certificate to private key
Subject: subject-DN
Issuer:  distinguished-name
Serial:  hex-serial-number

2. ...
```

2. **키 및 인증서를 내보냅니다.**

`pktool list` 명령에서 키 저장소 및 레이블을 사용합니다. 내보내기 파일에 대한 파일 이름을 제공합니다. 이름에 공백이 포함된 경우 이름을 큰따옴표로 묶습니다.

```
% pktool export keystore=keystore outfile=outfile-name label=label-name
```

3. **내보내기 파일을 암호로 보호합니다.**

   프롬프트에 키 저장소의 현재 암호를 입력합니다. 이 시점에 내보내기 파일의 암호를 만듭니다. 파일을 가져올 때 수신자가 이 암호를 제공해야 합니다.

```
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:      Create PKCS #12 password
```

작은 정보 - 내보내기 파일에서 별도로 암호를 보냅니다. 최적의 사용법으로 대역 외에서(예: 전화 통화 중) 암호를 알려줄 것을 제안합니다.

예 4-4 PKCS #12 형식의 인증서 및 개인 키 내보내기

다음 예에서는 사용자가 개인 키를 연관된 X.509 인증서와 함께 표준 PKCS #12 파일로 내보냅니다. 이 파일을 다른 키 저장소로 가져올 수 있습니다. PKCS #11 암호는 소스 키 저장소를 보호합니다. PKCS #12 암호는 PKCS #12 파일의 개인 데이터를 보호하는 데 사용됩니다. 파일을 가져오려면 이 암호가 필요합니다.

```
% pktool list
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
Asymmetric public keys:
1    RSA                My Cert
Certificates:
1    X.509 certificate
Label: My Cert
ID: d2:7e:20:04:a5:66:e6:31:90:d8:53:28:bc:ef:55:55:dc:a3:69:93
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Serial: 0x000001

% pktool export keystore=pkcs11 outfile=mydata.p12 label="My Cert"
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:      Create PKCS #12 password
```

그런 다음 사용자가 수신자에게 전화를 걸어서 PKCS #12 암호를 알려줍니다.

▼ pktool setpin 명령을 사용하여 암호문을 생성하는 방법

키 저장소의 객체에 대한, 그리고 키 저장소 자체에 대한 문장암호를 생성할 수 있습니다. 객체나 키 저장소에 액세스하려면 문장암호가 필요합니다. 키 저장소의 객체에 대한 문장암호 생성의 예는 예 4-4. “PKCS #12 형식의 인증서 및 개인 키 내보내기”를 참조하십시오.

1. **키 저장소 액세스를 위한 문장암호를 생성합니다.**

```
% pktool setpin keystore=nss|pkcs11 [dir=directory]
```

키 저장소에 대한 기본 디렉토리는 /var/*username*입니다.

PKCS #11 키 저장소의 초기 암호는 changeme입니다. NSS 키 저장소의 초기 암호는 비어 있는 암호입니다.

2. **프롬프트에 응답합니다.**

   현재 토큰 문장암호를 묻는 프롬프트가 표시되면 PKCS #11 키 저장소에 대한 토큰 PIN을 입력하고, NSS 키 저장소의 경우 Return 키를 누릅니다.

```
Enter current token passphrase:      Type PIN or press the Return key
Create new passphrase:               Type the passphrase that you want to use
Re-enter new passphrase:             Retype the passphrase
Passphrase changed.
```

   이제 키 저장소가 *passphrase*로 보호됩니다. 문장암호를 잊어버린 경우 키 저장소의 객체에 액세스할 수 없습니다.

3. **(옵션) 토큰 목록을 표시합니다.**

```
# pktool tokens
```

   metaslot 사용 여부에 따라 출력이 달라집니다. metaslot에 대한 자세한 내용은 "암호화 프레임워크의 개념" [9]을 참조하십시오.

   - metaslot이 사용으로 설정된 경우 pktools token 명령이 다음과 같은 출력을 생성합니다.

```
ID Slot   Name                      Token Name                     Flags
--        ---------                 ----------                     -----
0         Sun Metaslot              Sun Metaslot
1         Sun Crypto Softtoken      Sun Software PKCS#11 softtoken LIX
2         PKCS#11 Interface for TPM TPM                            LXS
```

   - metaslot이 사용 안함으로 설정된 경우 pktools token 명령이 다음과 같은 출력을 생성합니다.

```
ID Slot   Name                      Token Name                     Flags
--        ---------                 ----------                     -----
1         Sun Crypto Softtoken      Sun Software PKCS#11 softtoken LIX
2         PKCS#11 Interface for TPM TPM                            LXS
```

   두 출력 버전에서 플래그는 다음 조합일 수 있습니다.

   - L - 로그인 필요
   - I - 초기화됨
   - X - 사용자 PIN 만료됨
   - S - SO PIN 만료됨

예 4-5 키 저장소를 문장암호로 보호

다음 예는 NSS 데이터베이스에 대한 문장암호를 설정하는 방법을 보여줍니다. 문장암호를 만든 적이 없으므로 사용자가 첫번째 프롬프트에 Return 키를 누릅니다.

```
% pktool setpin keystore=nss dir=/var/nss
Enter current token passphrase:      Press the Return key
Create new passphrase: has8n0NdaH
Re-enter new passphrase: has8n0NdaH
Passphrase changed.
```

▼ pktool genkeypair 명령을 사용하여 키 쌍을 생성하는 방법

일부 응용 프로그램은 공개/개인 키 쌍이 필요합니다. 이 절차에서 이러한 키 쌍을 만들어서 저장합니다.

1. **(옵션) 키 저장소를 사용하려면 키 저장소를 만듭니다.**

   - **PKCS #11 키 저장소를 만들고 초기화하려면 pktool setpin 명령을 사용하여 암호문을 생성하는 방법 [58]을 참조하십시오.**
   - **NSS 키 저장소를 만들고 초기화하려면 예 4-5. “키 저장소를 문장암호로 보호”를 참조하십시오.**

2. **키 쌍을 만듭니다.**
   다음 방법 중 하나를 사용합니다.

   - **키 쌍을 만들어 파일에 저장합니다.**
     디스크의 파일에서 직접 키를 읽는 응용 프로그램의 경우 파일 기반 키가 생성됩니다. 일반적으로, OpenSSL 암호화 라이브러리를 직접 사용하는 응용 프로그램은 응용 프로그램의 키 및 인증서를 파일에 저장해야 합니다.

참고 - `file` 키 저장소는 타원 곡선(ec) 키 및 인증서를 지원하지 않습니다.

```
% pktool genkeypair keystore=file outkey=key-filename \
[format=der|pem] [keytype=rsa|dsa] [keylen=key-size]
```

`keystore=file`
`file` 값은 키에 대해 파일 유형의 저장소 위치를 지정합니다.

`outkey=`*key-filename*
키 쌍이 저장된 파일의 이름을 지정합니다.

`format=der|pem`

키 쌍의 인코딩 형식을 지정합니다. `der` 출력은 이진이고 `pem` 출력은 ASCII입니다.

`keytype=rsa|dsa`

`file` 키 저장소에 저장할 수 있는 키 쌍의 유형을 지정합니다. 정의는 DSA 및 RSA를 참조하십시오.

`keylen=`*key-size*

키의 길이를 비트 단위로 지정합니다. 숫자는 8로 나눌 수 있어야 합니다. 가능한 키 크기를 결정하려면 `cryptoadm list -vm` 명령을 사용합니다.

■ **키 쌍을 만들어 PKCS #11 키 저장소에 저장합니다.**

이 방법을 사용하기 전에 1단계을 완료해야 합니다.

PKCS #11 키 저장소는 하드웨어 장치에 객체를 저장하는 데 사용됩니다. 이 장치에는 암호화 프레임워크로 플러그인할 수 있는 Sun Crypto Accelerator 6000 카드, 신뢰된 플랫폼 모듈(TPM) 장치, 스마트 카드 등이 있습니다. 또한 PKCS #11을 사용하여 `softtoken`(디스크의 개인 하위 디렉토리에 객체를 저장하는 소프트웨어 기반 토큰)에 객체를 저장할 수 있습니다. 자세한 내용은 `pkcs11_softtoken`(5) 매뉴얼 페이지를 참조하십시오.

지정한 레이블로 키 저장소에서 키 쌍을 검색할 수 있습니다.

```
% pktool genkeypair label=key-label \
[token=token[:manuf[:serial]]] \
[keytype=rsa|dsa|ec]  [curve=ECC-Curve-Name]]\
[keylen=key-size] [listcurves]
```

`label=`*key-label*

키 쌍의 레이블을 지정합니다. 키 쌍은 자체 레이블로 키 저장소에서 검색할 수 있습니다.

`token=`*token*[:*manuf*[:*serial*]]

토큰 이름을 지정합니다. 기본적으로 토큰 이름은 `Sun Software PKCS#11 softtoken`입니다.

`keytype=rsa|dsa|ec [curve=`*ECC-Curve-Name*`]`

키 쌍 유형을 지정합니다. 타원 곡선(`ec`) 유형의 경우 선택적으로 곡선 이름을 지정합니다. 곡선 이름은 `listcurves` 옵션에 출력으로 나열됩니다.

`keylen=`*key-size*

키의 길이를 비트 단위로 지정합니다. 숫자는 8로 나눌 수 있어야 합니다.

listcurves

ec 키 유형에 대한 curve= 옵션에 값으로 사용할 수 있는 타원 곡선 이름을 나열합니다.

■ **키 쌍을 만들어 NSS 키 저장소에 저장합니다.**

NSS 키 저장소는 주 암호화 인터페이스로 NSS를 이용하는 서버에서 사용됩니다.

이 방법을 사용하기 전에 1단계을 완료해야 합니다.

```
% pktool keystore=nss genkeypair label=key-nickname \
[token=token[:manuf[:serial]]] \
[dir=directory-path] [prefix=database-prefix] \
[keytype=rsa|dsa|ec] [curve=ECC-Curve-Name]] \
[keylen=key-size] [listcurves]
```

keystore=nss

nss 값은 키에 대해 NSS 유형의 저장소 위치를 사용합니다.

label=*nickname*

키 쌍의 레이블을 지정합니다. 키 쌍은 자체 레이블로 키 저장소에서 검색할 수 있습니다.

token=*token*[:*manuf*[:*serial*]]

토큰 이름을 지정합니다. 기본적으로 토큰은 Sun Software PKCS#11 softtoken입니다.

dir=*directory*

NSS 데이터베이스에 대한 디렉토리 경로를 지정합니다. 기본적으로 *directory*가 현재 디렉토리입니다.

prefix=*database-prefix*

NSS 데이터베이스에 대한 접두어를 지정합니다. 기본값은 접두어 없음입니다.

keytype=rsa|dsa|ec [curve=*ECC-Curve-Name*]

키 쌍 유형을 지정합니다. 타원 곡선 유형의 경우 선택적으로 곡선 이름을 지정합니다. 곡선 이름은 listcurves 옵션에 출력으로 나열됩니다.

keylen=*key-size*

키의 길이를 비트 단위로 지정합니다. 숫자는 8로 나눌 수 있어야 합니다.

listcurves

ec 키 유형에 대한 curve= 옵션에 값으로 사용할 수 있는 타원 곡선 이름을 나열합니다.

3. **(옵션) 키가 존재하는지 확인합니다.**

키를 저장한 위치에 따라 다음 명령 중 하나를 사용합니다.

■ ***key-filename* 파일에서 키를 확인합니다.**

```
% pktool list keystore=file objtype=key infile=key-filename
Found n keys.
Key #1 - keytype:location (keylen)
```

■ **PKCS #11 키 저장소에서 키를 확인합니다.**

```
$ pktool list objtype=key
Enter PIN for keystore:
Found n keys.
Key #1 - keytype:location (keylen)
```

■ **NSS 키 저장소에서 키를 확인합니다.**

```
% pktool list keystore=nss dir=directory objtype=key
```

예 4-6 pktool 명령을 사용하여 키 쌍 만들기

다음 예에서 사용자가 처음으로 PKCS #11 키 저장소를 만듭니다. RSA 키 쌍에 대한 키 크기를 결정한 후에 응용 프로그램에 대한 키 쌍을 생성합니다. 마지막으로, 키 쌍이 키 저장소에 있는지 확인합니다. RSA 키 쌍의 두번째 인스턴스를 하드웨어에 저장할 수 있습니다. 사용자가 token 인수를 지정하지 않았으므로 키 쌍이 Sun Software PKCS#11 softtoken으로 저장됩니다.

```
# pktool setpin
Create new passphrase:
Re-enter new passphrase:     Retype password
Passphrase changed.
% cryptoadm list -vm | grep PAIR
...
CKM_DSA_KEY_PAIR_GEN       512  3072 .  .  .  .  .  .  .  .  .  X  .  .  .  .
CKM_RSA_PKCS_KEY_PAIR_GEN  256  8192 .  .  .  .  .  .  .  .  .  X  .  .  .  .
...
CKM_RSA_PKCS_KEY_PAIR_GEN  256  2048 X  .  .  .  .  .  .  .  .  X  .  .  .  .
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA,CKM_ECDSA_SHA1
% pktool genkeypair label=specialappkeypair keytype=rsa keylen=2048
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

% pktool list
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-----------------------------------------------------
Asymmetric public keys:
1        RSA                         specialappkeypair
```

예 4-7 타원 곡선 알고리즘을 사용하는 키 쌍 만들기

다음 예에서는 사용자가 타원 곡선(ec) 키 쌍을 키 저장소에 추가하고, 곡선 이름을 지정하고, 키 쌍이 키 저장소에 존재하는지 확인합니다.

```
% pktool genkeypair listcurves
secp112r1, secp112r2, secp128r1, secp128r2, secp160k1
.
.
.
c2pnb304w1, c2tnb359v1, c2pnb368w1, c2tnb431r1, prime192v2
prime192v3
% pktool genkeypair label=eckeypair keytype=ec curves=c2tnb431r1
% pktool list
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
Asymmetric public keys:
1    ECDSA              eckeypair
```

▼ pktool signcsr 명령을 사용하여 인증서 요청을 서명하는 방법

이 절차는 PKCS #10 인증서 서명 요청(CSR)을 서명하는 데 사용됩니다. CSR은 PEM 또는 DER 형식일 수 있습니다. 서명 프로세스가 X.509 v3 인증서를 발행합니다. PKCS #10 CSR을 생성하려면 pktool(1) 매뉴얼 페이지를 참조하십시오.

시작하기 전에 이 절차에서는 사용자가 인증 기관(CA)이며 CSR을 받아 파일에 저장했다고 가정합니다.

1. **pktool signcsr 명령의 필수 인수를 위해 다음 정보를 수집합니다.**

signkey	서명자의 키를 PKCS #11 키 저장소에 저장한 경우 signkey는 이 개인 키를 검색하는 레이블입니다. 서명자의 키를 NSS 키 저장소나 file 키 저장소에 저장한 경우 signkey는 이 개인 키를 보유하는 파일 이름입니다.
csr	CSR의 파일 이름을 지정합니다.
serial	서명된 인증서의 일련 번호를 지정합니다.
outcer	서명된 인증서에 대한 파일 이름을 지정합니다.
issuer	CA 발행자 이름을 식별 이름(DN) 형식으로 지정합니다.

signcsr 하위 명령의 선택적 인수에 대한 자세한 내용은 pktool(1) 매뉴얼 페이지를 참조하십시오.

2. **요청을 서명하고 인증서를 발행합니다.**
   예를 들어, 다음 명령은 PKCS #11 저장소에서 서명자의 키로 인증서를 서명합니다.

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
CN=rootsign Oracle"
```

   다음 명령은 파일에서 서명자의 키로 인증서를 서명합니다.

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
CN=rootsign Oracle"
```

3. **인증서를 요청자에게 보냅니다.**
   전자 메일, 웹 사이트 또는 다른 방식을 사용하여 인증서를 요청자에게 전달할 수 있습니다.
   예를 들어, 전자 메일을 사용하여 `ExampleCoCert2010` 파일을 요청자에게 보낼 수 있습니다.

▼ KMF에서 타사 플러그인을 관리하는 방법

키 저장소 이름을 제공하여 플러그인을 식별합니다. KMF에 플러그인을 추가할 때 소프트웨어가 키 저장소 이름으로 플러그인을 식별합니다. 플러그인이 옵션을 받아들이도록 정의할 수 있습니다. 이 절차는 KMF에서 플러그인을 제거하는 방법을 포함합니다.

1. **플러그인을 설치합니다.**

```
% /usr/bin/kmfcfg install keystore=keystore-name \
modulepath=path-to-plugin [option="option-string"]
```

   구문 설명

keystore-name	제공할 키 저장소의 고유한 이름을 지정합니다.
path-to-plugin	KMF 플러그인의 공유 라이브러리 객체에 대한 전체 경로를 지정합니다.
option-string	공유 라이브러리 객체에 대한 선택적 인수를 지정합니다.

2. **플러그인을 나열합니다.**

```
% kmfcfg list plugin
keystore-name:path-to-plugin [(built-in)] | [;option=option-string]
```

3. **플러그인을 제거하려면 설치를 해제하고 제거되었는지 확인합니다.**

```
% kmfcfg uninstall keystore=keystore-name
% kmfcfg plugin list
```

예 4-8 KMF 플러그인을 옵션과 함께 호출

다음 예에서 관리자가 KMF 플러그인을 사이트 특정 디렉토리에 저장합니다. 플러그인이 debug 옵션을 받아들이도록 정의됩니다. 관리자가 플러그인을 추가하고 플러그인이 설치되었는지 확인합니다.

```
# /usr/bin/kmfcfg install keystore=mykmfplug \
modulepath=/lib/security/site-modules/mykmfplug.so
# kmfcfg list plugin
KMF plugin information:
-----------------------
pkcs11:kmf_pkcs11.so.1 (built-in)
file:kmf_openssl.so.1 (built-in)
nss:kmf_nss.so.1 (built-in)
mykmfplug:/lib/security/site-modules/mykmfplug.so
# kmfcfg modify plugin keystore=mykmfplug option="debug"
# kmfcfg list plugin
KMF plugin information:
-----------------------
...
mykmfplug:/lib/security/site-modules/mykmfplug.so;option=debug
```

이제 플러그인이 디버깅 모드로 실행됩니다.

보안 용어

감사 정책 어떤 감사 이벤트를 기록할지 결정하는 전역 사용자별 설정입니다. 감사 서비스에 적용되는 전역 설정은 일반적으로 감사 추적에 포함할 선택적 정보 조각에 영향을 미칩니다. 두 설정 `cnt` 및 `ahlt`는 감사 대기열을 채울 때 시스템의 작업에 영향을 미칩니다. 예를 들어, 감사 정책에서 시퀀스 번호가 모든 감사 레코드에 속하도록 요구할 수 있습니다.

감사 추적 모든 호스트의 모든 감사 파일 모음입니다.

감사 파일 이진 감사 로그. 감사 파일은 감사 파일 시스템에 별도로 저장됩니다.

강화 호스트에 내재된 보안 취약성을 제거하도록 운영 체제의 기본 구성을 수정한 것입니다.

개인 키 각 사용자 주체에 제공되며 주체의 사용자와 KDC에만 알려진 키입니다. 사용자 주체의 경우 키는 사용자 암호를 기반으로 합니다. 키를 참조하십시오.

개인 키 암호화 개인 키 암호화에서 발신자와 수신자는 암호화에 동일한 키를 사용합니다. 공개 키 암호화도 참조하십시오.

갱신 가능 티켓 장시간 존재하는 티켓은 보안 위험이 있으므로 티켓을 `renewable`로 지정할 수 있습니다. 갱신 가능 티켓에는 두 개의 만료 시간 a) 티켓의 현재 인스턴스가 만료되는 시간 b) 티켓의 최대 수명이 있습니다. 클라이언트가 티켓을 계속 사용하려면 첫번째 만료가 발생하기 전에 티켓을 갱신합니다. 예를 들어, 1시간 동안 유효한 티켓이 있고 모든 티켓은 최대 10시간의 수명을 가질 수 있습니다. 티켓을 보유하는 클라이언트가 1시간보다 더 오래 티켓을 보관하려면 티켓을 갱신해야 합니다. 티켓이 최대 티켓 수명에 도달하면 자동으로 만료되어 갱신할 수 없습니다.

검사 엔진 파일에 알려진 바이러스가 있는지 조사하는, 외부 호스트에 상주하는 타사 응용 프로그램입니다.

공개 키 기술에 대한 정책 키 관리 프레임워크(KMF)에서 정책은 인증서 사용을 관리합니다. KMF 정책 데이터베이스는 KMF 라이브러리에서 관리되는 키 및 인증서 사용을 제약할 수 있습니다.

공개 키 암호화 각 사용자가 두 개의 키, 즉 하나의 공개 키와 하나의 개인 키를 사용하는 암호화 체계입니다. 공개 키 암호화에서 발신자는 수신자의 공개 키를 사용하여 메시지를 암호화하고, 수신자는 개인 키를 사용하여 암호를 해독합니다. Kerberos 서비스는 개인 키 시스템입니다. 개인 키 암호화도 참조하십시오.

공급자 Oracle Solaris의 암호화 프레임워크 기능에서 소비자에게 제공된 암호화 서비스입니다. 공급자의 예로 PKCS #11 라이브러리, 커널 암호화 모듈, 하드웨어 가속기가 있습니다. 공급

자는 암호화 프레임워크에 플러그인되므로 플러그인이라고도 합니다. 소비자의 예는 소비자를 참조하십시오.

공용 객체 `root` 사용자가 소유하고 어디서든 읽을 수 있는 파일입니다(예: `/etc` 디렉토리의 파일).

관계 `kdc.conf` 또는 `krb5.conf` 파일에 정의된 구성 변수 또는 관계입니다.

관리자 주체 *username*`/admin`(예: `jdoe/admin`) 형식의 이름을 가진 사용자 주체입니다. 관리자 주체는 일반 사용자 주체보다 더 많은 권한(예: 정책 변경)을 가질 수 있습니다. 주체 이름, 사용자 주체도 참조하십시오.

권한 1. 일반적인 의미는 컴퓨터 시스템에서 일반 사용자의 권한을 넘는 작업을 수행할 수 있는 능력입니다. 수퍼 유저 권한은 수퍼 유저에게 부여된 모든 rights(권한)입니다. 권한 있는 사용자 또는 권한 있는 응용 프로그램은 추가 권한이 부여된 사용자 또는 응용 프로그램입니다.

2. Oracle Solaris 시스템에서 프로세스에 대한 별개의 권한입니다. 권한은 `root`인 프로세스를 좀 더 세부적으로 제어합니다. 권한은 커널에서 정의되고 시행됩니다. 권한을 프로세스 권한 또는 커널 권한이라고도 합니다. 권한에 대한 자세한 설명은 `privileges`(5) 매뉴얼 페이지를 참조하십시오.

권한 모델 수퍼 유저 모델보다 더 엄격한 컴퓨터 시스템의 보안 모델입니다. 권한 모델에서 프로세스를 실행하려면 권한이 필요합니다. 시스템 운영은 관리자가 해당 프로세스에 보유한 권한으로 기반으로 별개의 부분으로 나눌 수 있습니다. 관리자의 로그인 프로세스에 권한을 지정할 수 있습니다. 또는 특정 명령에만 효력을 발휘하도록 권한을 지정할 수 있습니다.

권한 부여 1. Kerberos에서는 주체가 서비스를 사용할 수 있는지, 어떤 객체에 주체가 액세스할 수 있는지, 각 객체에 대해 허용된 액세스 유형 등을 결정하는 프로세스입니다.

2. 사용자 권한 관리에서는 보안 정책으로 금지된 일련의 작업을 수행하기 위해 역할 또는 사용자에게 지정할 수 있는(또는 권한 프로파일에 포함할 수 있는) 권한입니다. 권한 부여는 커널에서가 아니라 사용자 응용 프로그램 레벨에서 실행됩니다.

권한 세트 권한 모음입니다. 각 프로세스에는 프로세스가 특정 권한을 사용할 수 있는지 여부를 결정하는 4개의 권한 세트가 있습니다. 제한 세트, 유효 세트, 허가된 세트, 상속 가능한 세트를 참조하십시오.

또한 권한의 기본 세트는 로그인 시 사용자의 프로세스에 지정된 권한 모음입니다.

권한 에스컬레이션 지정된 권한(기본값 대체 권한 포함)이 허용하는 리소스 범위 밖에 있는 리소스에 대한 액세스 권한을 얻는 것입니다. 그 결과, 프로세스가 권한이 없는 작업을 수행할 수 있습니다.

권한 인식 코드를 통해 권한 사용을 켜고 끄는 프로그램, 스크립트, 명령입니다. 운용 환경에서 켜져 있는 권한을 프로세스에 제공해야 합니다. 프로그램의 사용자가 권한을 프로그램에 추가한 권한 프로파일을 사용하도록 하면 됩니다. 권한에 대한 자세한 설명은 `privileges`(5) 매뉴얼 페이지를 참조하십시오.

권한 있는 응용 프로그램 시스템 컨트롤을 대체할 수 있는 응용 프로그램입니다. 응용 프로그램이 특정 UID, GID, 권한 부여 또는 권한과 같은 보안 속성을 검사합니다.

권한 프로파일 프로파일이라고도 합니다. 역할 또는 사용자에 지정할 수 있는 보안 대체 모음입니다. 권한 프로파일에는 권한 부여, 권한, 보안 속성이 포함된 명령 및 보충 프로파일이라고 하는 기타 권한 프로파일이 포함될 수 있습니다.

기밀성 프라이버시를 참조하십시오.

기본 주체 이름의 첫번째 부분입니다. 인스턴스, 주체 이름, 영역도 참조하십시오.

기본 세트 로그인 시 사용자 프로세스에 지정되는 권한 세트입니다. 수정되지 않은 시스템에서 각 사용자의 초기 상속 가능한 세트는 로그인 시 기본 세트와 같습니다.

네트워크 애플리케이션 서버 `ftp`와 같은 네트워크 응용 프로그램을 제공하는 서버입니다. 영역에 여러 네트워크 애플리케이션 서버를 포함할 수 있습니다.

네트워크 정책 네트워크 트래픽을 보호하기 위해 네트워크 유틸리티가 구성하는 설정입니다. 네트워크 보안에 대한 자세한 내용은 "Oracle Solaris 11.2의 네트워크 보안 "을 참조하십시오.

다이제스트 메시지 다이제스트를 참조하십시오.

단일 시스템 이미지 단일 시스템 이미지는 Oracle Solaris 감사에 사용되어 동일한 이름 지정 서비스를 사용하는 감사 시스템 그룹을 설명합니다. 이러한 시스템은 해당 감사 레코드를 중앙 감사 서버로 보내고, 여기서 레코드가 한 시스템에서 나온 것처럼 레코드를 비교할 수 있습니다.

동기 감사 이벤트 감사 이벤트의 다수를 차지합니다. 이러한 이벤트는 시스템의 프로세스와 연관됩니다. 프로세스와 연관된 출처를 알 수 없는 이벤트는 실패한 로그인과 같은 동기 이벤트입니다.

마스터 KDC 각 영역의 주 KDC로, Kerberos 관리 서버인 `kadmind`와 인증 및 티켓 부여 데몬인 `krb5kdc`를 포함합니다. 각 영역에는 적어도 하나의 마스터 KDC가 있어야 하고, 클라이언트에 인증 서비스를 제공하는 많은 중복된 슬레이브 KDC를 포함할 수 있습니다.

메시지 다이제스트 메시지 다이제스트는 메시지에서 계산된 해시 값입니다. 해시 값은 메시지를 거의 고유하게 식별합니다. 다이제스트는 파일 무결성 확인에 유용합니다.

무결성 사용자 인증과 더불어, 암호화 체크섬을 통해 전송된 데이터의 유효성을 제공하는 보안 서비스입니다. 인증, 프라이버시도 참조하십시오.

문장암호 개인 키를 문장암호 사용자가 만들었는지 확인하는 데 사용되는 문구입니다. 좋은 문장암호는 10-30자 길이로, 영문자와 숫자를 섞어서 만들고 단순한 문구와 단순한 이름을 피합니다. 통신을 암호화 및 해독하기 위해 개인 키 사용을 인증하려면 문장암호를 묻는 메시지가 나타납니다.

방식 1. 데이터 인증 또는 기밀성을 이루기 위한 암호화 기법을 지정하는 소프트웨어 패키지입니다. 예: Kerberos V5, Diffie-Hellman 공개 키.

2. Oracle Solaris의 암호화 프레임워크 기능에서 특정 목적을 위한 알고리즘의 구현입니다. 예를 들어, 인증에 적용된 DES 방식(예: CKM_DES_MAC)은 암호화에 적용된 DES 방식(예: CKM_DES_CBC_PAD)과 별도의 방식입니다.

보안 방식 방식을 참조하십시오.

보안 서비스 서비스를 참조하십시오.

보안 속성 수퍼 유저가 아닌 일반 사용자가 관리 명령을 실행할 때 명령을 성공하게 해주는 보안 정책의 대체입니다. 수퍼 유저 모델에서 `setuid root` 및 `setgid` 프로그램이 보안 속성입니다. 이러한 속성을 명령에 적용할 때 누가 명령을 실행하든지 관계없이 명령을 성공합니다. 권한 모델에서 커널 권한과 기타 rights(권한)이 보안 속성으로서 `setuid root` 프로그램을 대체합니다. 권한 모델은 `setuid` 및 `setgid` 프로그램을 보안 속성으로 인식하므로 수퍼 유저 모델과 호환됩니다.

보안 정책 정책을 참조하십시오.

보안 종류 종류를 참조하십시오.

보안 키 개인 키를 참조하십시오.

비동기 감사 이벤트 비동기 이벤트는 시스템 이벤트의 소수를 차지합니다. 이러한 이벤트는 어떤 프로세스와 연관되지 않으므로 프로세스를 차단했다가 나중에 깨울 수 없습니다. 비동기 이벤트의 예로, 초기 시스템 부트 및 PROM 진입/종료 이벤트가 있습니다.

사용자 주체 특정 사용자에 기인한 주체입니다. 사용자 주체의 기본 이름은 사용자 이름이고, 선택적 인스턴스는 의도한 해당 자격 증명 용도를 설명하는 데 사용되는 이름입니다(예: `jdoe` 또는 `jdoe/admin`). 사용자 인스턴스라고도 합니다. 서비스 주체도 참조하십시오.

상속 가능한 세트 `exec`의 호출에서 프로세스가 상속할 수 있는 권한 세트입니다.

서버 네트워크 클라이언트에 리소스를 제공하는 주체입니다. 예를 들어, `central.example.com` 시스템에 `ssh`를 제공하면 해당 시스템은 `ssh` 서비스를 제공하는 서버입니다. 서비스 주체도 참조하십시오.

서버 주체 (RPCSEC_GSS API) 서비스를 제공하는 주체입니다. 서버 주체는 *service@host* 형식으로 ASCII 문자열로 저장됩니다. 클라이언트 주체도 참조하십시오.

서비스 1. 종종 여러 대의 서버에 의해 네트워크 클라이언트에 제공된 리소스입니다. 예를 들어, `central.example.com` 시스템에 `rlogin`을 제공하는 경우 해당 시스템은 `rlogin` 서비스를 제공하는 서버입니다.

2. 인증 외의 보호 레벨을 제공하는 보안 서비스(무결성 또는 프라이버시)입니다. 무결성 및 프라이버시를 참조하십시오.

서비스 주체 서비스에 Kerberos 인증을 제공하는 주체입니다. 서비스 주체의 경우 기본 이름은 `ftp`와 같은 서비스 이름이고, 해당 인스턴스는 서비스를 제공하는 시스템의 정규화된 호스트 이름입니다. 호스트 주체, 사용자 주체도 참조하십시오.

서비스 키 서비스 주체 및 KDC에서 공유되고 시스템 한도 밖에서 배포되는 암호화 키입니다. 키를 참조하십시오.

세션 키 인증 서비스 또는 TGS(티켓 부여 서비스)에서 생성된 키입니다. 세션 키는 클라이언트와 서비스 간에 보안 트랜잭션을 제공하기 위해 생성됩니다. 세션 키의 수명은 단일 로그인 세션으로 제한됩니다. 키를 참조하십시오.

소비자 Oracle Solaris의 암호화 프레임워크 기능에서 소비자는 공급자로부터 전달된 암호화 서비스의 사용자입니다. 소비자는 응용 프로그램, 최종 사용자 또는 커널 작업일 수 있습니다. 소비자의 예로 Kerberos, IKE, IPsec 등이 있습니다. 공급자의 예는 공급자를 참조하십시오.

소프트웨어 공급자 Oracle Solaris의 암호화 프레임워크 기능에서 암호화 서비스를 제공하는 커널 소프트웨어 모듈 또는 PKCS #11 라이브러리입니다. 공급자를 참조하십시오.

수퍼 유저 모델 컴퓨터 시스템의 전형적인 UNIX 보안 모델입니다. 수퍼 유저 모델에서 관리자는 all-or-nothing 방식으로 시스템을 제어합니다. 일반적으로 시스템을 관리하려는 경우 사용자는 수퍼 유저(`root`)로 로그인하여 모든 관리 작업을 수행할 수 있습니다.

슬레이브 KDC 마스터 KDC의 복사본으로, 대부분의 마스터 기능을 수행할 수 있습니다. 각 영역에는 대개 여러 개의 슬레이브 KDC(마스터 KDC는 하나만)가 있습니다. KDC, 마스터 KDC도 참조하십시오.

시드 무작위 수를 생성하기 위한 숫자 스타터입니다. 스타터가 무작위 소스에서 기원할 때 시드를 무작위 시드라고 합니다.

알고리즘 암호화 알고리즘입니다. 입력을 암호화하거나 해시하는 확립된 순환적 계산 프로시저입니다.

암호 정책 암호를 생성하는 데 사용할 수 있는 암호화 알고리즘입니다. 암호 변경 주기, 암호 시도 허용 회수, 기타 보안 고려 사항 등 암호와 관련한 일반적인 사안이라고 할 수 있습니다. 보안 정책에 암호가 필요합니다. 암호 정책에서는 암호를 AES 알고리즘으로 암호화해야 하고, 추가로 암호 강도와 관련된 요구 사항이 필요할 수 있습니다.

암호화 알고리즘 알고리즘을 참조하십시오.

암호화 프레임워크의 정책 Oracle Solaris의 암호화 프레임워크 기능에서 정책은 기존 암호화 방식을 사용 안함으로 설정합니다. 그러면 방식을 사용할 수 없습니다. 암호화 프레임워크의 정책은 DES와 같은 공급자가 `CKM_DES_CBC`와 같은 특정 방식을 사용하는 것을 금지할 수 있습니다.

애플리케이션 서버 네트워크 애플리케이션 서버를 참조하십시오.

액세스 제어 목록(ACL) 액세스 제어 목록(ACL)은 전통적인 UNIX 파일 보호보다 좀 더 세부적인 파일 보안을 제공합니다. 예를 들어, ACL을 사용하여 파일에 그룹 읽기 액세스를 허용하면서 해당 그룹의 한 멤버만 파일 쓰기를 허용할 수 있습니다.

역할 지정된 사용자만 맡을 수 있는 권한 있는 응용 프로그램을 실행하기 위한 특수한 신원입니다.

영역 1. 단일 Kerberos 데이터베이스와 일련의 KDC(키 배포 센터)에 의해 제공된 논리적 네트워크입니다.

2. 주체 이름의 세번째 부분입니다. 주체 이름 `jdoe/admin@CORP.EXAMPLE.COM`의 경우 영역은 `CORP.EXAMPLE.COM`입니다. 주체 이름도 참조하십시오.

유효 세트 현재 프로세스에 발효 중인 권한 세트입니다.

이름 서비스 범위 역할이 작동하도록 허가된 범위입니다. 즉, NIS LDAP와 같은 지정된 이름 지정 서비스에서 제공하는 개별 호스트 또는 모든 호스트를 말합니다.

인스턴스 주체 이름의 두번째 부분으로, 인스턴스는 주체의 기본 부분을 한정합니다. 서비스 주체의 경우 인스턴스는 필수 사항입니다. 인스턴스는 `host/central.example.com`과 같이 호스트의 정규화된 도메인 이름입니다. 사용자 주체의 경우 인스턴스는 선택 사항입니다. 그러나 `jdoe` 및 `jdoe/admin`은 고유한 주체입니다. 기본, 주체 이름, 서비스 주체, 사용자 주체도 참조하십시오.

인증 주체의 제시된 신원을 확인하는 프로세스입니다.

인증서 공개 키 인증서는 공개 키 값(인증서 이름 및 서명한 사람과 같은 일부 인증서 생성 정보 포함), 인증서의 해시 또는 체크섬, 해시의 디지털 서명을 인코딩하는 데이터 세트입니다. 이러한 값이 결합되어 인증서가 형성됩니다. 디지털 서명이 있으면 인증서가 수정되지 않은 것입니다.

자세한 내용은 키를 참조하십시오.

인증자 인증자는 티켓(KDC에서) 및 서비스(서버에서)를 요청할 때 클라이언트에 의해 전달됩니다. 최근 시점에서 확인할 수 있는 클라이언트 및 서버에만 알려진 세션 키를 사용하여 생성된 정보를 포함하므로 트랜잭션이 안전한 것으로 나타납니다. 티켓과 함께 사용할 경우 인증자는 사용자 주체를 인증할 수 있습니다. 인증자에는 사용자의 주체 이름, 사용자 호스트의 IP 주소, 시간 기록이 포함됩니다. 티켓과 달리, 인증자는 대개 서비스 액세스를 요청할 때 한번만 사용할 수 있습니다. 인증자는 클라이언트 및 서버에 대한 세션 키를 사용하여 암호화됩니다.

자격 증명 티켓 및 일치하는 세션 키를 포함하는 정보 패키지입니다. 주체의 신원을 인증하는 데 사용됩니다. 티켓, 세션 키도 참조하십시오.

자격 증명 캐시 KDC로부터 받은 자격 증명을 포함하는 저장 공간(대개 파일)입니다.

잘못된 티켓 아직 사용할 수 없는 후일자 티켓입니다. 잘못된 티켓은 유효해질 때까지 애플리케이션 서버에서 거부합니다. 유효화하려면 시작 시간이 지난 후에 `VALIDATE` 플래그 세트를 사용하여 TGS 요청의 클라이언트가 KDC에 잘못된 티켓을 제시해야 합니다. 후일자 티켓도 참조하십시오.

장치 정책 커널 레벨의 장치 보호입니다. 장치 정책은 장치에 두 개의 권한 세트로 구현됩니다. 첫번째 권한 세트는 장치의 읽기 액세스를 제어합니다. 두번째 권한 세트는 장치의 쓰기 액세스를 제어합니다. 정책을 참조하십시오.

장치 할당 사용자 레벨의 장치 보호입니다. 장치 할당은 하나의 장치를 한번에 한 사용자만 배타적으로 사용하도록 합니다. 장치 재사용 전에 장치 데이터를 비웁니다. 권한 부여를 사용하여 장치 할당이 허가된 사용자를 제한할 수 있습니다.

전달 가능 티켓 클라이언트가 원격 호스트에서 티켓을 요청하기 위해 전체 인증 프로세스를 거치지 않고도 사용할 수 있는 티켓입니다. 예를 들어, 사용자 `jennifer`의 시스템에 있는 동안 사용자 `david`가 전달 가능 티켓을 얻은 경우 `david`는 새 티켓을 얻지 않고도(다시 인증 받을 필요 없이) 자신의 시스템에 로그인할 수 있습니다. 프록시 가능 티켓도 참조하십시오.

정책 일반적으로, 의사결정 및 조치를 반영하거나 결정하는 계획이나 행동 방침입니다. 컴퓨터 시스템의 경우 정책은 대개 보안 정책을 의미합니다. 사이트의 보안 정책은 처리 중인 정보의 민감도를 정의하는 규칙 세트이자, 허용되지 않은 액세스로부터 정보를 보호하는 데 사용되는 측정치입니다. 예를 들어, 시스템을 감사하고 사용할 장치를 할당하며 암호를 6주마다 변경하도록 보안 정책을 수립할 수 있습니다.

Oracle Solaris OS의 특정 영역에서 정책을 구현하는 방법은 감사 정책, 암호화 프레임워크의 정책, 장치 정책, Kerberos 정책, 암호 정책 및 rights policy(권한 정책)을 참조하십시오.

제한 세트 프로세스와 그 자식에 사용 가능한 권한에 대한 외부 제한입니다.

종류 전통적으로 보안 종류와 인증 종류는 인증 유형(AUTH_UNIX, AUTH_DES, AUTH_KERB)을 지칭한 종류로서, 동일한 의미입니다. RPCSEC_GSS는 인증과 더불어 무결성과 프라이버시 서비스를 제공하지만 역시 보안 종류입니다.

주체 1. 네트워크 통신에 참여하는 고유한 이름이 지정된 클라이언트/사용자 또는 서버/서비스입니다. Kerberos 트랜잭션에는 주체들(서비스 주체 및 사용자 주체) 간의 상호 작용 또는 주체와 KDC 간의 상호 작용이 관여합니다. 다신 말해서, 주체는 Kerberos가 티켓을 지정할 수 있는 고유한 개체입니다. 주체 이름, 서비스 주체, 사용자 주체도 참조하십시오.

2. (RPCSEC_GSS API) 클라이언트 주체, 서버 주체를 참조하십시오.

주체 이름 1. *primary/instance@REALM* 형식의 주체 이름입니다. 인스턴스, 기본, 영역도 참조하십시오.

2. (RPCSEC_GSS API) 클라이언트 주체, 서버 주체를 참조하십시오.

책임 구분 최소한의 특권 개념의 일부입니다. 책임 구분을 통해 한 사용자가 트랜잭션을 완성하는 모든 작업을 수행하거나 승인할 수 없도록 합니다. 예를 들어, RBAC에서 보안 대체 지정으로부터 로그인 사용자 생성을 분리할 수 있습니다. 한 역할이 사용자를 만듭니다. 별도의 역할이 권한 프로파일, 역할, 기존 사용자의 권한과 같은 보안 속성을 지정할 수 있습니다.

초기 티켓 (기존 TGT(티켓 부여 티켓)에 기반하지 않고) 직접 발행된 티켓입니다. 암호를 변경하는 응용 프로그램과 같은 일부 서비스는 `initial`로 표시된 티켓이 필요할 수 있으므로 클라이언트가 보안 키를 알고 있다는 것을 스스로 보증해야 합니다. 초기 티켓은 클라이언트가 (오랫동안 존재해 왔던 TGT(티켓 부여 티켓)에 의존하는 대신) 최근에 자체 인증되었음을 나타내므로 이 보증은 매우 중요합니다.

최소 권한의 원칙 최소한의 특권을 참조하십시오.

최소한의 특권 지정된 프로세스를 일부 수퍼 유저 권한에만 제공하는 보안 모델입니다. 최소 권한 모델은 일반 사용자가 파일 시스템 마운트 및 파일 소유권 변경과 같은 개인적인 관리 작업을 수행

할 수 있도록 충분한 권한을 지정합니다. 반면에 프로세스는 완전한 수퍼 유저 권한(즉 모든 권한)이 아닌, 작업 완료에 필요한 권한으로만 실행됩니다. 버퍼 오버플로우 같은 프로그래밍 오류로 인한 손해는, 보호된 시스템 파일 읽기/쓰기 또는 시스템 정지 같은 중요한 능력에 액세스할 수 없는 비루트 사용자에 국한될 수 있습니다.

최소화 서버 실행에 필요한 최소 운영 체제를 설치합니다. 서버 작동에 직접적인 관련이 없는 소프트웨어는 설치되지 않거나 설치 후 삭제됩니다.

출처를 알 수 없는 감사 이벤트 AUE_BOOT 이벤트와 같이 개시자가 결정할 수 없는 감사 이벤트입니다.

클라이언트 좁은 의미로, 사용자 대신 네트워크 서비스를 이용하는 프로세스입니다. 예를 들어, `rlogin`을 사용하는 응용 프로그램이 있습니다. 어떤 경우 서버 자체가 다른 서버나 서비스의 클라이언트가 될 수 있습니다.

더 넓은 의미로, a) Kerberos 자격 증명을 수신하고 b) 서버에서 제공한 서비스를 이용하는 호스트입니다.

간단히 말하면, 서비스를 이용하는 주체입니다.

클라이언트 주체 (RPCSEC_GSS API) RPCSEC_GSS로 보안된 네트워크 서비스를 사용하는 클라이언트(사용자 또는 응용 프로그램)입니다. 클라이언트 주체 이름은 `rpc_gss_principal_t` 구조 형태로 저장됩니다.

클럭 불균형 Kerberos 인증 시스템에 참여하는 모든 호스트의 내부 시스템 클럭에 차이가 날 수 있는 최대 시간입니다. 참여하는 호스트 사이에 클럭 불균형을 초과할 경우 요청이 거부됩니다. 클럭 불균형은 `krb5.conf` 파일에 지정할 수 있습니다.

키 1. 일반적으로, 두 가지의 주요 키 유형 중 하나입니다.

- 대칭 키 - 암호 해독 키와 똑같은 암호화 키입니다. 대칭 키는 파일을 암호화하는 데 사용됩니다.
- 비대칭 키 또는 공개 키 - Diffie-Hellman 또는 RSA와 같은 공개 키 알고리즘에 사용되는 키입니다. 공개 키에는 한 사용자에만 알려진 개인 키, 서버나 일반 리소스에서 사용되는 공개 키, 그리고 둘을 조합한 개인-공개 키 쌍이 포함됩니다. 개인 키는 보안 키라고도 합니다. 공개 키는 공유 키 또는 공통 키라고도 합니다.

2. keytab 파일의 항목(주체 이름)입니다. keytab 파일도 참조하십시오.

3. Kerberos에서 암호화 키로 사용되며 다음 세 가지 유형이 있습니다.

- 개인 키 - 주체 및 KDC에서 공유되고 시스템 한도 밖에서 배포되는 암호화 키입니다. 개인 키도 참조하십시오.
- 서비스 키 - 이 키는 개인 키와 동일한 목적을 제공하지만, 서버 및 서비스에서 사용됩니다. 서비스 키도 참조하십시오.
- 세션 키 - 단일 로그인 세션 기간으로 제한된 수명 동안 두 주체 간에 사용되는 임시 암호화 키입니다. 세션 키도 참조하십시오.

키 저장소 키 저장소는 응용 프로그램별로 검색하기 위한 암호, 문장암호, 인증서 및 기타 인증 객체를 저장합니다. 키 저장소는 일부 응용 프로그램이 사용하는 기술 또는 위치에 따라 달라질 수 있습니다.

티켓 사용자의 신원을 서버나 서비스로 안전하게 전달하는 데 사용되는 정보 패킷입니다. 티켓은 단일 클라이언트에만, 그리고 특정 서버의 특정 서비스에만 유효합니다. 티켓에는 서비스의 주체 이름, 사용자의 주체 이름, 사용자 호스트의 IP 주소, 시간 기록, 티켓의 수명을 정의하는 값이 포함됩니다. 클라이언트 및 서비스에서 사용할 무작위 세션 키로 티켓이 생성됩니다. 일단 티켓이 만들어지면 만료될 때까지 재사용할 수 있습니다. 티켓은 새로운 인증자와 함께 제시될 때 클라이언트 인증에만 사용됩니다. 인증자, 자격 증명, 서비스, 세션 키를 참조하십시오.

티켓 파일 자격 증명 캐시를 참조하십시오.

프라이버시 전송된 데이터를 보내기 전에 암호화하는 보안 서비스입니다. 프라이버시에는 데이터 무결성과 사용자 인증도 포함됩니다. 인증, 무결성, 서비스를 참조하십시오.

프로파일 셸 권한 관리에서 역할(또는 사용자)이 권한 프로파일에 지정된 권한 있는 응용 프로그램을 명령줄에서 실행할 수 있는 셸입니다. 프로파일 셸 버전은 시스템에서 사용할 수 있는 셸(예: `bash`의 `pfbash` 버전)에 해당합니다.

프록시 가능 티켓 클라이언트에 작업을 수행하는 대신, 서비스에서 사용할 수 있는 티켓입니다. 따라서 서비스가 클라이언트의 프록시 역할을 한다고 말할 수 있습니다. 티켓을 사용하여 서비스는 클라이언트의 신원을 차용할 수 있습니다. 프록시 가능 티켓을 사용하여 다른 서비스에 대한 서비스 티켓을 얻을 수 있지만, TGT(티켓 부여 티켓)는 얻을 수 없습니다. 프록시 가능 티켓과 전달 가능 티켓의 차이점은, 프록시 가능 티켓은 단일 작업에만 유효하다는 것입니다. 전달 가능 티켓도 참조하십시오.

하드웨어 공급자 Oracle Solaris의 암호화 프레임워크 기능에서 장치 드라이버 및 해당 하드웨어 가속기입니다. 하드웨어 공급자는 컴퓨터 시스템에서 값비싼 암호화 작업 부담을 덜어주므로 CPU 리소스를 확보하여 다른 용도로 사용할 수 있습니다. 공급자를 참조하십시오.

허가된 세트 프로세스에서 사용할 수 있는 권한 세트입니다.

호스트 네트워크를 통해 액세스 가능한 시스템입니다.

호스트 주체 `ftp`, `rcp` 또는 `rlogin`과 같은 다양한 네트워크 서비스를 제공하기 위해 주체(기본 이름 `host`로 서명됨)가 설정되는 특정 인스턴스의 서비스 주체입니다. 호스트 주체의 예는 `host/central.example.com@EXAMPLE.COM`입니다. 서버 주체도 참조하십시오.

후일자 티켓 후일자 티켓은 생성 후 지정된 시간까지 유효해지지 않습니다. 예를 들어, 이러한 티켓은 밤늦게 실행하려는 일괄 처리 작업에 유용합니다. 티켓을 훔친 경우 일괄 처리 작업이 실행될 때까지 티켓을 사용할 수 없기 때문입니다. 후일자 티켓을 발행할 때 `invalid`로 발행되고 a) 시작 시간이 지날 때까지 b) 클라이언트가 KDC에서 검증으로 요청할 때까지 해당 방법을 유지합니다. 후일자 티켓은 보통 TGT(티켓 부여 티켓)의 만료 시간까지 유효합니다. 그러나 후일자 티켓이 `renewable`로 표시된 경우 티켓의 수명이 보통 TGT(티켓 부여 티켓)의 전체 수명 기간과 똑같이 설정됩니다. 잘못된 티켓, 갱신 가능 티켓도 참조하십시오.

AES Advanced Encryption Standard의 머리글자어로, 고급 암호화 표준입니다. 대칭 128비트 블록 데이터 암호화 기술입니다. 미국 정부는 2000년 10월 알고리즘의 Rijndael 변형을 암호화 표준으로 채택했습니다. AES가 정부 표준으로 사용자 주체 암호화를 대체합니다.

authenticated rights profile(인증된 권한 프로파일) 지정된 사용자나 역할이 프로파일에서 작업을 실행하기 전에 암호를 입력해야 하는 권한 프로파일입니다. 이 동작은 `sudo` 동작과 비슷합니다. 암호가 유효하며 구성 가능한 기간입니다.

Blowfish 32-448비트의 가변 길이 키를 사용하는 대칭 블록 암호화 알고리즘입니다. 저작자인 Bruce Schneier에 따르면, Blowfish는 키를 자주 바꾸지 않는 응용 프로그램에 최적화되어 있습니다.

DES Data Encryption Standard의 머리글자어로, 데이터 암호화 표준입니다. 1975년에 개발되고 1981년에 ANSI에 의해 ANSI X.3.92로 표준화된 대칭 키 암호화 방법입니다. DES에서는 56비트 키를 사용합니다.

Diffie-Hellman 프로토콜 공개 키 암호화라고도 합니다. 1976년 Diffie와 Hellman이 개발한 비대칭 암호화 키 계약 프로토콜입니다. 이 프로토콜을 사용하면 두 사용자가 사전 보안 없이 비보안 매체를 통해 보안 키를 교환할 수 있습니다. Diffie-Hellman은 Kerberos에서 사용됩니다.

DSA Digital Signature Algorithm의 머리글자어로, 디지털 서명 알고리즘입니다. 512-4096비트의 가변 키 크기를 사용하는 공개 키 알고리즘입니다. 미국 정부 표준인 DSS는 1024비트까지 지원합니다. DSA는 입력에 SHA1을 사용합니다.

ECDSA Elliptic Curve Digital Signature Algorithm의 머리글자어로, 타원 곡선 디지털 서명 알고리즘입니다. 타원 곡선 수학을 기반으로 하는 공개 키 알고리즘입니다. ECDSA 키 크기는 동일한 길이의 서명을 생성하는 데 필요한 DSA 공개 키의 크기보다 많이 작습니다.

FQDN 정규화된 도메인 이름입니다. 간단한 `denver`와 대조되는 `central.example.com`을 예로 들 수 있습니다.

GSS-API Generic Security Service Application Programming Interface의 약자. Kerberos 서비스를 포함하여 다양한 모듈형 보안 서비스를 지원하는 네트워크 계층입니다. GSS-API는 보안 인증, 무결성, 프라이버시 서비스를 제공합니다. 인증, 무결성, 프라이버시를 참조하십시오.

KDC Key Distribution Center의 머리글자어로, 키 배포 센터입니다. 세 가지 Kerberos V5 구성 요소가 있는 시스템입니다.

- 주체 및 키 데이터베이스
- 인증 서비스
- TGS(티켓 부여 서비스)

각 영역에는 마스터 KDC가 있고 하나 이상의 슬레이브 KDC가 있어야 합니다.

Kerberos 인증 서비스, 서비스에서 사용되는 프로토콜 또는 서비스 구현에 사용되는 코드입니다.

Kerberos V5 구현에 가장 근접한 Oracle Solaris의 Kerberos 구현입니다.

"Kerberos"와 "Kerberos V5"는 기술적으로 서로 다르지만 Kerberos 문서에서 종종 바꿔서 사용하기도 합니다.

Kerberos(Cerberus라고도 씀)는 그리스 신화에서 지옥의 문을 지키는 머리가 셋 달린 사나운 개입니다.

Kerberos 정책 Kerberos 서비스에서 암호 사용을 통제하는 규칙 세트입니다. 정책을 통해 주체의 액세스나 티켓 수명 매개변수를 규제할 수 있습니다.

keytab 파일 하나 이상의 키(주체)를 포함하는 키 테이블 파일입니다. 사용자가 암호를 사용하는 것처럼 호스트나 서비스는 keytab 파일을 사용합니다.

kvno 키 버전 번호. 생성 순서대로 특정 키를 추적하는 시퀀스 번호입니다. 가장 높은 kvno가 최신의 가장 현재 키입니다.

MAC 1. MAC(메시지 인증 코드)를 참조하십시오.

2. 레이블 지정이라고도 합니다. 정부 보안 기술에서 MAC은 필수 액세스 제어입니다. MAC의 예로 Top Secret 및 Confidential과 같은 레이블이 있습니다. MAC은 DAC(모든 액세스 제어)과 대조를 이룹니다. DAC의 예로 UNIX 사용 권한이 있습니다.

3. 하드웨어에서 LAN의 고유한 시스템 주소입니다. 시스템이 이더넷에 있는 경우 MAC은 이더넷 주소입니다.

MAC(메시지 인증 코드) MAC는 데이터 무결성을 보증하고 데이터 발신을 인증합니다. MAC는 도청에 대해 보호되지 않습니다.

MD5 디지털 서명을 포함하여 메시지 인증용으로 사용되는 반복적인 암호화 해시 함수입니다. 이 기능은 1991년 Rivest가 개발했습니다. 이 기술은 더 이상 사용되지 않습니다.

NTP Network Time Protocol의 약자. 네트워크 환경에서 정밀한 시간이나 네트워크 클럭 동기화(또는 둘 다)를 관리할 수 있는 델라웨어 대학교에서 설계한 소프트웨어입니다. NTP를 사용하여 Kerberos 환경에서 클럭 불균형을 유지 관리할 수 있습니다. 클럭 불균형도 참조하십시오.

PAM Pluggable Authentication Module의 약자. 여러 인증 방식에서 서비스를 재컴파일할 필요 없이 사용할 수 있는 프레임워크입니다. PAM은 로그인 시 Kerberos 세션 초기화를 사용으로 설정합니다.

privileged user(권한 있는 사용자) 컴퓨터 시스템에 대해 일반 사용자의 권한 밖에 있는 권한이 지정된 사용자입니다. trusted users(신뢰할 수 있는 사용자)도 참조하십시오.

QOP Quality of Protection의 머리글자어로, 보호 품질입니다. 무결성 서비스나 프라이버시 서비스와 함께 사용되는 암호화 알고리즘을 선택할 수 있는 매개변수입니다.

RBAC Oracle Solaris의 사용자 권한 관리 기능의 일종인 역할 기반 액세스 제어입니다. rights(권한)을 참조하십시오.

RBAC 정책 rights policy(권한 정책)을 참조하십시오.

reauthentication(재인증) 컴퓨터 작업을 수행하기 위해 암호를 제공하도록 요구하는 것입니다. 일반적으로 `sudo` 작업에 재인증이 필요합니다. 인증된 권한 프로파일에 재인증을 요구하는 명령을 포함할 수 있습니다. authenticated rights profile(인증된 권한 프로파일)을 참조하십시오.

rights policy(권한 정책) 명령과 연관된 보안 정책입니다. 현재 Oracle Solaris의 유효한 정책은 `solaris`입니다. `solaris` 정책은 권한 및 확장된 권한 정책, 권한 부여 및 `setuid` 보안 속성을 인식합니다.

rights(권한) all-or-nothing 수퍼 유저 모델의 대안입니다. 사용자 권한 관리 및 프로세스 권한 관리를 통해 조직은 수퍼 유저의 권한을 분담하고 사용자 또는 역할에 이를 지정할 수 있습니다. Oracle Solaris의 권한은 커널 권한, 권한 부여 및 프로세스를 UID 또는 GID로 실행하는 기능으로 구현됩니다. 권한 프로파일 및 역할에 권한을 수집할 수 있습니다.

RSA 디지털 서명 및 공개 키 암호화 체계를 얻기 위한 방법입니다. 1978년에 개발자 Rivest, Shamir, Adleman이 처음 기술했습니다.

SEAM Solaris 시스템의 Kerberos 초기 버전에 대한 제품 이름입니다. 이 제품은 MIT(Massachusetts Institute of Technology)에서 개발된 Kerberos V5 기술을 기반으로 합니다. 이제 SEAM을 Kerberos 서비스라고 부릅니다. MIT 버전과는 약간 다릅니다.

Secure Shell 비보안 네트워크를 통해 보안 원격 로그인 및 기타 보안 네트워크 서비스를 제공하기 위한 특수한 프로토콜입니다.

SHA1 Secure Hashing Algorithm의 머리글자어입니다. 이 알고리즘은 2^{64} 미만의 입력 길이에서 작동하여 메시지 다이제스트를 생성합니다. SHA1 알고리즘은 DSA에 입력됩니다.

stash 파일 stash 파일은 KDC용 마스터 키의 암호화된 복사본을 포함합니다. `kadmind` 및 `krb5kdc` 프로세스를 시작하기 전에 KDC를 자동으로 인증하도록 서버를 재부트할 때 이 마스터 키가 사용됩니다. stash 파일에 마스터 키가 포함되므로 stash 파일과 그 백업을 안전하게 보관해야 합니다. 암호화가 훼손되면 키를 사용하여 KDC 데이터베이스를 액세스하거나 수정해야 합니다.

TGS Ticket-Granting Service의 머리글자어로, 티켓 부여 서비스입니다. 티켓 발행을 담당하는 KDC의 부분입니다.

TGT Ticket-Granting Ticket의 머리글자어로, 티켓 부여 티켓입니다. 클라이언트가 다른 서비스에 대한 티켓을 요청할 수 있는 KDC에서 발행한 티켓입니다.

trusted users(신뢰할 수 있는 사용자) 결정된 사용자는 특정 신뢰 레벨에서 관리 작업을 수행할 수 있습니다. 대개 관리자가 신뢰할 수 있는 사용자의 로그인을 먼저 만들고 사용자의 신뢰 및 능력 레벨에 맞는 관리 권한을 지정합니다. 이러한 사용자는 시스템 구성 및 유지 관리 작업을 수행하는 데 도움이 됩니다. 권한 있는 사용자라고도 합니다.

VPN(가상 사설망) 암호화를 사용하고 공용 네트워크를 통한 사용자 연결을 터널링하여 보안 통신을 제공하는 네트워크입니다.

색인