

open



USE



IMPROVE



EVANGELIZE

OpenSolaris Crypto Framework

Wolfgang Ley
Technology Consultant
Sun Microsystems

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
:::
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



OpenSolaris Crypto Framework Overview

- Motivation
- Glossary
- User-Level Crypto Framework (uCF)
 - Consumers and Providers
- Kernel-Level Crypto Framework (kCF)
 - Consumers and Providers
- Administration and Debugging
- Example Code
- Current and upcoming projects



Motivation

- Crypto used in a lot of places
 - Userland: SSL, PAM, GSS, Kerberos, ssh, md5, ...
 - Kernel: IPsec, WiFi drivers, ...
- Standard algorithms
 - Implemented over and over again
- Not extensible
 - Just think of adding elliptic curve crypto...
- No central management
 - Try to disable MD5 on the entire system
- Enable hardware acceleration?



Motivation

- **Several userland solutions (libraries)**
 - OpenSSL, Mozilla NSS, Java, ...
 - No uniform API and not easy extensible
- **Standardization of a Crypto API**
 - RSA developed Public Key Crypto Standard (PKCS)
 - PKCS#11 defines a generic and flexible API
- **Create a plugable framework**
 - To be used in userland (e.g. via PKCS#11 API)
 - Same implementations usable by kernel
 - Make other software to use this framework



Glossary

- Boring but needed
 - To understand standards, documentation and code
- PKCS#11 (you'll hear this often)
 - Public-Key Crypto Standard #11 from RSA Labs
 - Defines an API (Cryptoki, pronounced crypto-key)
- Consumer
 - Application, library or kernel requesting crypto services
- Provider
 - A userland library or kernel module which offers crypto services to the consumers



Glossary

- Plug-In
 - Different term for a provider (which can be plugged into the OpenSolaris Crypto Framework)
- Mechanism
 - Combination of algorithm (optional with mode) and the way it is being used
 - E.g. DES used for authentication vs. encryption
 - Defines which features we get and which arguments we need to use it



Glossary

- **Token**
 - Implements a mechanism
 - Can store information (e.g. initialization vectors, key, state etc.) for use with the mechanism
 - Might be hardware or software (“soft token”)
- **Slot**
 - Connectors to crypto services
 - Collection of tokens (implementations)
- **Metaslot**
 - Virtual (default) slot as superset of other slots
 - Selects best implementation (e.g. hardware)



Glossary

- Session
 - Active connection between consumer and a token
- Objects
 - Store information (e.g. keys)
 - Session objects: only valid during session lifetime
 - Token objects: persistent across sessions (e.g. `$HOME/.sunw/pkcs11_softtoken`)
- See paper for more information and as a reference



User-Level Crypto Framework (uCF)

- Libraries to be used by consumers
- Software plug-in (implements crypto)
- Plug-in to access kernel (hardware accl.)
- Administrative interface
- Subsystem to verify integrity
- Applications using this framework

Libraries (uCF)

- Main library `libpkcs11.so`
- Implements RSA PKCS#11 API version 2.20
- Additional convenience functions to ease initialization and key handling
 - `SUNW_C_GetMechSession`
 - `SUNW_C_KeyToObject`
- Engine (pkcs11) for OpenSSL Library
- Digest library (`libmd.so`)
- Private libraries: `libelfsign.so` and `libcryptoutil.so`



Software plug-in (uCF)

- Software token provider
 - Library `pkcs11_softtoken.so` (used via `libpkcs11.so` and not directly)
 - Implements a lot of mechanisms
 - See manpage `pkcs11_softtoken(5)` for list
- Obsolete software token providers
 - `pkcs11_softtoken_extra` (previously provided stronger crypto which is now part of the normal provider)
 - `SUNWcry` and `SUNWcryr` packages no longer needed (removed from OpenSolaris)



Plug-in to access hardware (uCF)

- Provider to access hardware via kernel
 - Plug-in `pkcs11_kernel.so` always active
 - Use hardware acceleration where available (e.g. UltraSparc T1 or T2 chips, Crypto cards like SCA-6000)
 - May not have any slots if no hardware support is available (framework will then use software only)
 - Work ongoing to add more hardware support (e.g. VIA PadLock hardware crypto)



Administration

- One central command: `cryptoadm (1M)`
- Show and control/configure all components
 - Show providers, slots, mechanism
 - Enable and disable components
 - Install and uninstall providers (e.g. new providers to use hardware features)
- Output to long for a slide (see paper)



Verification of providers

- Providers must have a valid ELF signature
 - Must be signed with a certificate that has been signed by Sun Microsystems Inc.
 - Required to comply with US export restrictions regarding pluggable cryptography
- See `elfsign(1)` for signature handling
 - Signing ELF binaries
 - Verify signatures
 - Generate PKCS#10 certificate request
- During runtime signature is verified by a daemon: `kcfcd(1M)`



Verification of providers

- `kcfld(1M)` managed by SMF service
 - `svc:/system/cryptosvc:default`
 - Already started in single-user mode
- **Disabling `kcfld(1M)` will disable all Crypto**
 - Will also affect authentication (e.g. no login)
 - See console and messages for errors
 - `libpkcs11: Unable to contact kcfld: Bad file number`



Example consumers (userland)

- Have a look at source from some consumers
- Example applications:
 - `digest(1)` to calculate message digests
 - `mac(1)` to use message authentication codes
 - `encrypt(1)` and `decrypt(1)`
 - `pktool(1)` for generic key management
- A bit more complicated:
 - `pkcs11` engine for OpenSSL



Kernel-Level Crypto Framework (kCF)

- Consumer documentation in progress
 - Documentation not yet bundled with OpenSolaris
 - Parts of the functional specs available on the OpenSolaris webpage (to be completed in near future)
 - Draft manpages available for download
- Logic similar to PKCS#11
 - Lookup a mechanism
 - Create and initialize context
 - Use context (incl. finalize and collect result)
 - Destroy context



Kernel-Level Crypto Framework (kCF)

- Have a look at example source code
 - IPsec
 - WiFi drivers (see net80211 code)
 - Ongoing work (ZFS crypto, `lofi` crypto)
- Check OpenSolaris project page for updates
 - In particular for more upcoming documentation



Kernel-Level Providers

- Much better documented and stable
- Import up to 8 functions
- Export up to 67 routines
 - Depends on the type of provider and offered capabilities
- Two different provider types available
 - Kernel Software Provider
 - Kernel Hardware Provider



Kernel-Level Software Provider

- Software provider is a loadable kernel module (without external function calls)
- Each module implements one algorithm
 - Might offer various implementations or types of this algorithm
 - Will therefore offer one or more mechanisms
 - Identification with numbers like PKCS#11
- All calls to the provider are synchronous
- Kernel framework cares about the rest
 - Scheduling, callbacks for async use etc.



Kernel-Level Hardware Provider

- Hardware Device which offers crypto
 - Visible via a device driver
 - Node in the device tree
- Might implement multiple mechanisms
- Usually in asynchronous mode
 - Kernel framework cares about the rest like scheduling, callbacks, keeping state etc.
- Driver can optional require sessions
 - Session management not always needed though (e.g. for random number generator)



Debugging

- Check which mechanisms are available
 - `cryptoadm list -mv`
- Enable/Disable providers or mechanisms
 - Again with the `cryptoadm` command
- Debugging of `libcryptoutil.so`
 - Environmentvariable `SUNW_CRYPTO_DEBUG`
 - Can be set to `syslog` or `stderr`
- Loading and using providers
 - Start `kcfld(1M)` with `SUNW_CRYPTO_DEBUG`



Example Code Flow

- Longer code not really suitable for slides
- See the appendix of the paper for complete source code examples
- Short overview of used functions (in the example)
 - To get an impression about the different APIs



Example MD5 Code Flow PKCS#11 only

- `C_Initialize()`
- `C_GetSlotList(0, NULL, &count)`
- `malloc(count*sizeof(CK_SLOT_ID))`
- `C_GetSlotList()`
- **Find mechanism in a loop**
 - `C_GetMechanismInfo(slotlist[i])` until match
- `C_OpenSession()`
- `C_DigestInit()`
-



Example MD5 Code Flow PKCS#11 only

- `C_DigestUpdate()`
- `C_DigestFinal()`
- `C_CloseSession()`
- `C_Finalize()`



Example MD5 Code Flow PKCS#11 / Sun

- `SUNW_C_GetMechSession()`
- `C_DigestInit()`
- `C_DigestUpdate()`
- `C_DigestFinal()`
- `C_CloseSession()`
- `C_Finalize()`



Example MD5 Code Flow libmd

- `MD5Init()`
- `MD5Update()`
- `MD5Final()`
- **Or even simpler (if you have all data at once)**
 - `md5_calc()`



Active (and future) projects

- Merge and unify code: Project Highlander
 - Consolidate `libpkcs11`, `pkcs11_kernel` and `pkcs11_softtoken`
- Optimized userland algorithms
 - `libsoftcrypto` instead of generic implementation in `pkcs11_softtoken`
- Build kCF library in userland: `libkcf`
 - Helpful for code which is in kernel and userland (for example ZFS)
- More consumers: ZFS, lofi crypto
- Key Management Framework: KMF



References

- OpenSolaris Crypto Framework Project
 - <http://www.opensolaris.org/os/project/crypto/>
- Key Management Framework
 - <http://opensolaris.org/os/project/kmf/>
- Generic lofi crypto
 - <http://opensolaris.org/os/project/loficc/>
- ZFS encryption
 - <http://opensolaris.org/os/project/zfs-crypto/>
- See paper for more references

open



USE



IMPROVE



EVANGELIZE

Thank you!

Wolfgang Ley
Technology Consultant
wolfgang.ley@sun.com

“open” artwork and icons by chandan:
<http://blogs.sun.com/chandan>

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை