Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API
- PKCS#11 - Application C API

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API
- PKCS#11 - Application C API
- PKCS#15 - Credential object discovery and storage using path-based tree layout (atop ISO-7816-4)

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API
- PKCS#11 - Application C API
- PKCS#15 - Credential object discovery and storage using path-based tree layout (atop ISO-7816-4)
- OpenPGP Smartcard - Simple ISO-7816 profile

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API
- PKCS#11 - Application C API
- PKCS#15 - Credential object discovery and storage using path-based tree layout (atop ISO-7816-4)
- OpenPGP Smartcard - Simple ISO-7816 profile
- FIPS-201 (PIV) - U.S. Federal government technical standard

Interface Complexity

Design by Committees

- ISO-7816-4 - Contact cards. Command/response semantics and binary formats (APDUs)
- ISO-14443 - Contactless cards
- ISO-18092 - Contactless cards (NFC)
- CCID - USB-layer protocol (encapsulates ISO-7816 APDUs)
- PC/SC - Application C API
- PKCS#11 - Application C API
- PKCS#15 - Credential object discovery and storage using path-based tree layout (atop ISO-7816-4)
- OpenPGP Smartcard - Simple ISO-7816 profile
- FIPS-201 (PIV) - U.S. Federal government technical standard
- NIST SP 800-96 - ISO-7816 Profile. PC/SC API. ISO-14443, etc details.

Interface Complexity

Binary object formats

- ASN.1 DER

Interface Complexity

Binary object formats

- ASN.1 DER
- X.509

Interface Complexity

Binary object formats

- ASN.1 DER
- X.509
- PKCS#7

Interface Complexity

Binary object formats

- ASN.1 DER
- X.509
- PKCS#7
- OpenPGP - RFC 4408, RFC 6637, etc

Interface Complexity

Binary object formats

- ASN.1 DER
- X.509
- PKCS#7
- OpenPGP - RFC 4408, RFC 6637, etc
- OpenSSH - RFC 4251, RFC 4716, draft-miller-ssh-agent-02, etc

Software Complexity

Projects

- OpenSC - Incredibly comprehensive. Backbone of smartcard support in open source community.

Software Complexity
Projects

- OpenSC - Incredibly comprehensive. Backbone of smartcard support in open source community.
- GnuPG - Works well with OpenPGP Smartcard-based tokens out-of-the-box using system's native PC/SC module (OpenSC's libpcsclite.so on Linux, PCSC.framework on macOS, winscard.dll on Windows).

Software Complexity

Projects

- OpenSC - Incredibly comprehensive. Backbone of smartcard support in open source community.
- GnuPG - Works well with OpenPGP Smartcard-based tokens out-of-the-box using system's native PC/SC module (OpenSC's libpcsclite.so on Linux, PCSC.framework on macOS, winscard.dll on Windows).
- Scute - PKCS#11 bridge to GnuPG scdaemon. Limited to public-key operations.

Software Complexity

Projects

- OpenSC - Incredibly comprehensive. Backbone of smartcard support in open source community.
- GnuPG - Works well with OpenPGP Smartcard-based tokens out-of-the-box using system's native PC/SC module (OpenSC's libpcsclite.so on Linux, PCSC.framework on macOS, winscard.dll on Windows).
- Scute - PKCS#11 bridge to GnuPG scdaemon. Limited to public-key operations.
- PyKCS11 - Python bindings to PKCS#11 API.

Software Complexity

Projects

- OpenSC - Incredibly comprehensive. Backbone of smartcard support in open source community.
- GnuPG - Works well with OpenPGP Smartcard-based tokens out-of-the-box using system's native PC/SC module (OpenSC's libpcsclite.so on Linux, PCSC.framework on macOS, winscard.dll on Windows).
- Scute - PKCS#11 bridge to GnuPG scdaemon. Limited to public-key operations.
- PyKCS11 - Python bindings to PKCS#11 API.
- OpenSCDP - Java development framework and toolchain.

Software Complexity

OpenSC

- Legacy code, including built-in workarounds to card errata, makes it difficult to understand and modify.

Software Complexity

OpenSC

- Legacy code, including built-in workarounds to card errata, makes it difficult to understand and modify.
- Hardcodes too much knowledge about specific smartcards.

Software Complexity

OpenSC

- Legacy code, including built-in workarounds to card errata, makes it difficult to understand and modify.
- Hardcodes too much knowledge about specific smartcards.
- Emulation modes, number of moving parts create doubt about whether you're using hardware securely.

Software Complexity
OpenSC

- Legacy code, including built-in workarounds to card errata, makes it difficult to understand and modify.
- Hardcodes too much knowledge about specific smartcards.
- Emulation modes, number of moving parts create doubt about whether you're using hardware securely.
- Architecture primarily directed at implementing the standardized interfaces at expense of ability to develop above and below those layers.

Software Complexity

GnuPG

- OpenPGP focused.

Software Complexity

GnuPG

- OpenPGP focused.
- Relies too heavily on scdaemon agent and IPC for interface abstraction.

Applying Lua

Importing and Exporting PKCS#11 and PC/SC Interfaces

- Best scripting language for creating bindings to PKCS#11 and PC/SC C APIs exported by driver modules.

Applying Lua

Importing and Exporting PKCS#11 and PC/SC Interfaces

- Best scripting language for creating bindings to PKCS#11 and PC/SC C APIs exported by driver modules.
- Can export PKCS#11 and PC/SC C APIs. Lua “framework” transparently fits into small, dynamically loadable module. Few or no issues related to symbol pollution, dependency pollution, or reentrancy.

Applying Lua

Importing and Exporting PKCS#11 and PC/SC Interfaces

- Best scripting language for creating bindings to PKCS#11 and PC/SC C APIs exported by driver modules.
- Can export PKCS#11 and PC/SC C APIs. Lua "framework" transparently fits into small, dynamically loadable module. Few or no issues related to symbol pollution, dependency pollution, or reentrancy.
- Allows rapid implementation of bridges and adapters so solutions are consumable using standard interfaces.

Applying Lua

Importing PC/SC Module

```
struct pcsc_dylib
  void *handle;

  long (*SCardEstablishContext)(pcsc_dword, const void *, const void *, pcsc_context *);
  long (*SCardReleaseContext)(pcsc_context);
  long (*SCardListReaders)(pcsc_context, const char *, char *, pcsc_dword *);
  long (*SCardGetStatusChange)(pcsc_context, pcsc_dword, struct pcsc_readerstate *, pcsc
  long (*SCardConnect)(pcsc_context, const char *, pcsc_dword, pcsc_dword, pcsc_card *,
  long (*SCardReconnect)(pcsc_card, pcsc_dword, pcsc_dword, pcsc_dword, pcsc_dword *);
  long (*SCardDisconnect)(pcsc_card, pcsc_dword);
  long (*SCardStatus)(pcsc_card, char *, pcsc_dword *, pcsc_dword *, pcsc_dword *, unsig
  long (*SCardBeginTransaction)(pcsc_card);
  long (*SCardEndTransaction)(pcsc_card, pcsc_dword);
  long (*SCardTransmit)(pcsc_card, const struct pcsc_io_request *, const unsigned char *
  long (*SCardControl)(pcsc_card, pcsc_dword, const unsigned char *, pcsc_dword, unsigne
};
```

Applying Lua

Using PC/SC Module

```lua
local pcsc = require"pcsc"
local driver = assert(pcsc.loadpath"/System/Library/Frameworks/PCSC.framework/PCSC")
local ctx = assert(driver:establish_context(pcsc.SCOPE_SYSTEM))
local function readers(driver, ctx)
  local blob = driver:list_readers(ctx)
  return coroutine.wrap(function ()
    for rdr in blob:gmatch("[^\000]+") do
      coroutine.yield(rdr)
    end
  end)
end
local function cards(driver, ctx)
  return coroutine.wrap(function ()
    for rdr in readers(driver, ctx) do
        local card, protocol = assert(driver:connect(ctx, rdr, pcsc.SHARE_EXCLUSIVE,
          pcsc.PROTOCOL_T0|pcsc.PROTOCOL_T1))
      coroutine.yield(card, protocol)
    end
  end)
end
for card in cards(driver, ctx) do
  local rdr, state, protocol, atr = driver:status(card)
  print(rdr)
  local token = require"openpgp.card".new(driver, card)
  for keyno=1,3 do
    local key = token:get_pubkey(keyno)
    print("KEYNO    ", keyno)
    print("KEYID    ", auxlib.tohex(key:keyid()):upper())
    print("KEYGROUP", key:keygrip())
    print(token:exportssh(keyno))
  --  print(token:exportpgp(keyno))
  end
end
```

Applying Lua

Using PC/SC Module

```
Yubico Yubikey NEO OTP+CCID
KEYNO     1
KEYID     A8F94B862EA7457F
KEYGROUP  2EA7457F
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCrrnJpsTd6b6nLClApabjYCKk7CI0Mv5rcL2zggp12jiZIJizr
KEYNO     2
KEYID     A61436808415E31F
KEYGROUP  8415E31F
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCN2Js3UD1NG/WAcqpLxOLiLpEYbrUDNuwt2SFAd7H9Vojr3xgk
KEYNO     3
KEYID     C935C3805CC81644
KEYGROUP  5CC81644
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCUICF3D+UDOJ++XbceWKMc/23DfvjlIK3SH1Ndx+jN7St5yal6
```