PV181 Laboratory of security and applied cryptography

CryptoAPI

Marek Sýs, Danil Leksin, Zdeněk Říha

CRoCS

Centre for Research on Cryptography and Security

CAPI

CryptoAPI (Cryptographic Application Programming Interface, Microsoft Cryptography API, MS-CAPI or simply CAPI) is an application programming interface included with Microsoft Windows operating systems that provides services to enable developers to secure Windows-based applications using cryptography. It is a set of dynamically linked libraries that provides an abstraction layer which isolates programmers from the code used to encrypt the data. (CryptoAPI supports both public-key and symmetric key cryptography)

CAPI provides:

1. Secure data storing
2. Ability to transfer data
3. Validation from 3rd party users
4. Work with cryptographic standards
5. Extension

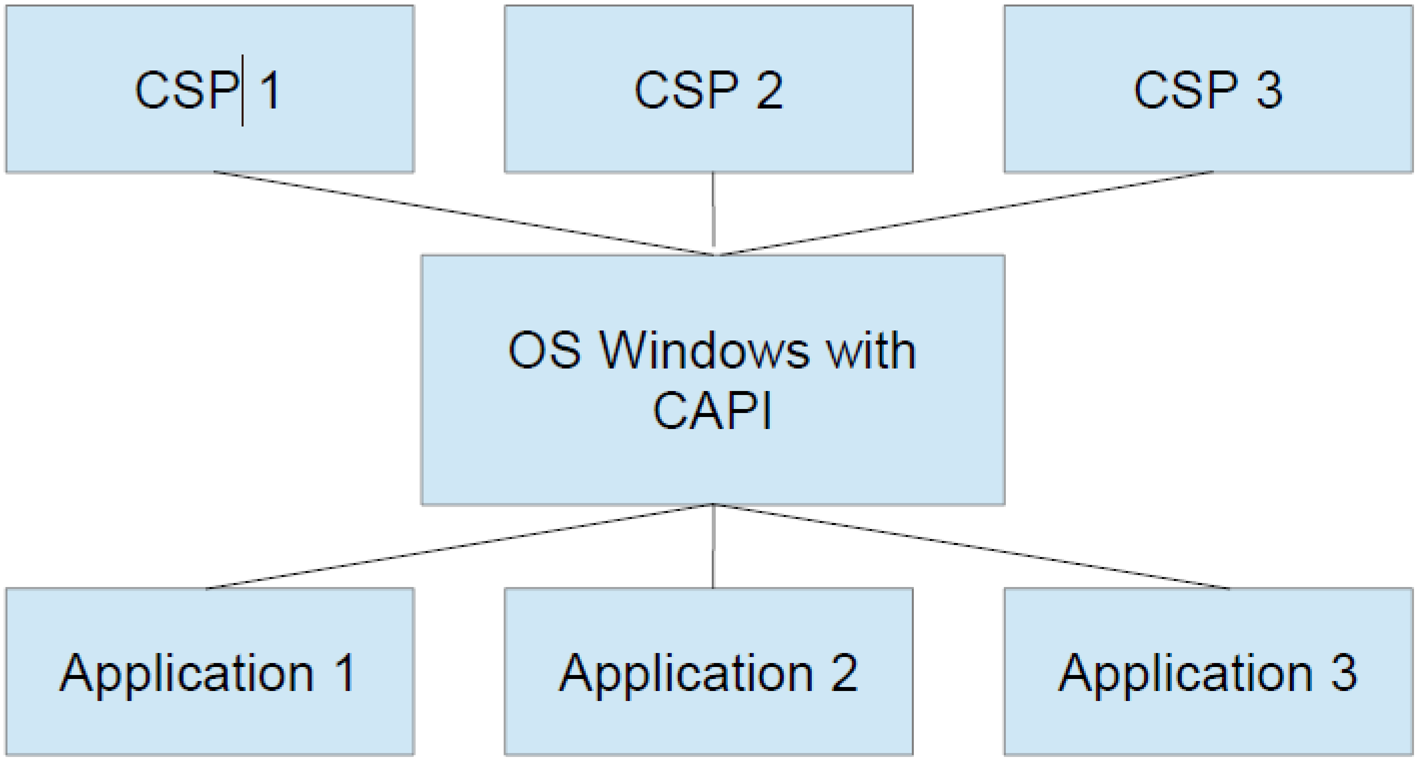
CAPI functionality groups:

1. Basic cryptographic functions:
   1. encoding / decoding
   2. hash function
   3. initializing CSP, working with context
   4. key generation
   5. key exchanging
2. Functions for working with certificates
3. High-level functions
4. Low-level functions

CSP

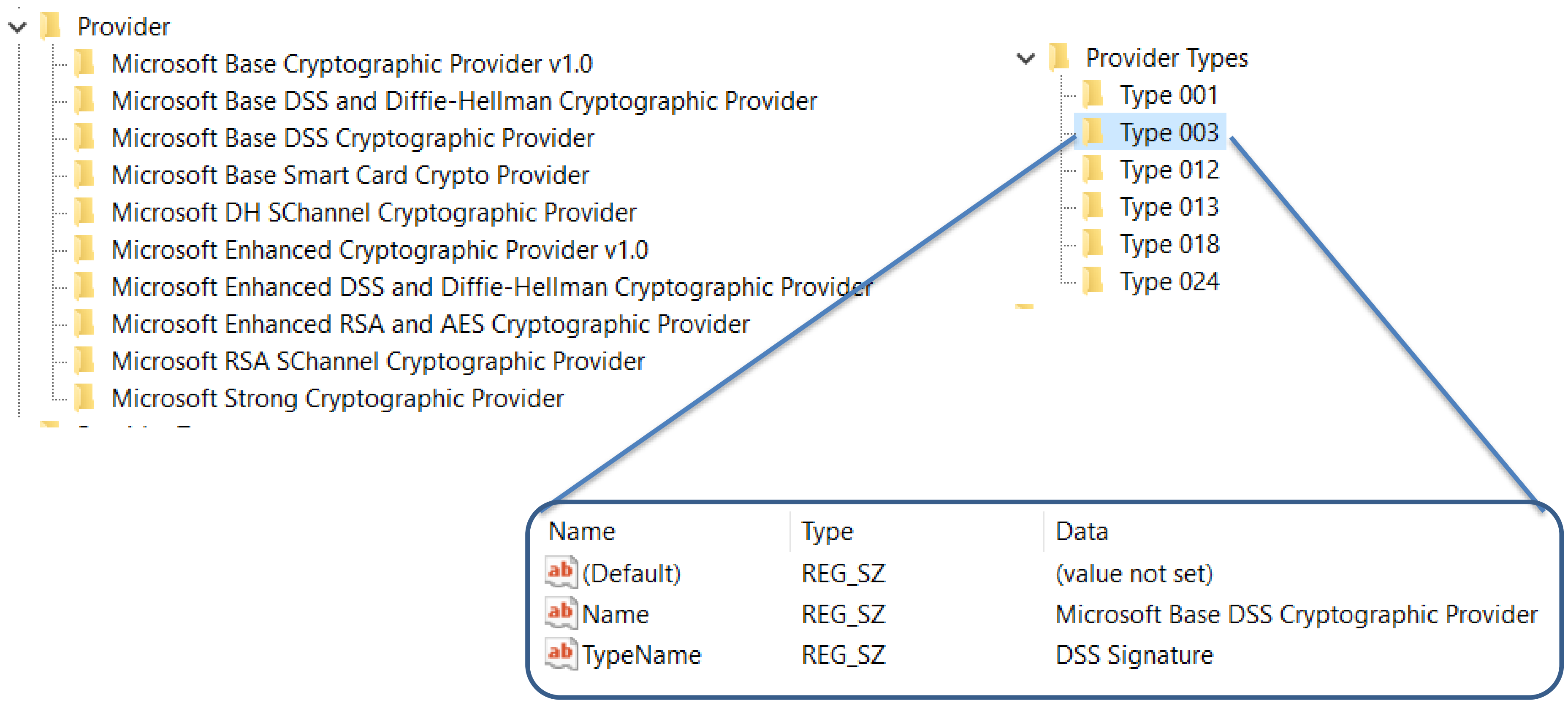
- CSP (Cryptography Service Provider) - is a software library that implements the Microsoft CryptoAPI (CAPI). CSPs implement **encoding** and **decoding** functions, which computer application programs may use.
- CSP provides:
  - implementation of the standard interface
  - work with encode / decode keys
  - inability to interference from third parties
- 2 function groups for working with CSP:
  - initialization of the context and getting CSP parameters
  - Key generation and function for work with them
  - encode / decode functions
  - Hash functions

CAPI & CSP & Apps



CSP on current machine

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\



- Provider
  - Microsoft Base Cryptographic Provider v1.0
  - Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
  - Microsoft Base DSS Cryptographic Provider
  - Microsoft Base Smart Card Crypto Provider
  - Microsoft DH SChannel Cryptographic Provider
  - Microsoft Enhanced Cryptographic Provider v1.0
  - Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
  - Microsoft Enhanced RSA and AES Cryptographic Provider
  - Microsoft RSA SChannel Cryptographic Provider
  - Microsoft Strong Cryptographic Provider

- Provider Types
  - Type 001
  - Type 003
  - Type 012
  - Type 013
  - Type 018
  - Type 024

Name	Type	Data
(Default)	REG_SZ	(value not set)
Name	REG_SZ	Microsoft Base DSS Cryptographic Provider
TypeName	REG_SZ	DSS Signature

CAPI and terminology

- Work with: CSPs + keys
- Key containers
  - store keys (symmetric, private or public)
  - associated to CSP

- Context - session between CAPI and client App
- Session key – volatile objects **never** leave CSP
  - import, export functions
- Key BLOB - contain an encrypted private key

CAPI programming

1. Call **CryptAcquireContext** function (returns handle)
2. work with handle – potentially other objects should be created (e.g. hash objects)

```
BOOL CryptAcquireContext(
  HCRYPTPROV *phProv,          // pointer to a handle of a CSP
  LPCSTR     szContainer,      // key container name
  LPCSTR     szProvider,       // name of the CSP
  DWORD      dwProvType,       // type of provider to acquire
  DWORD      dwFlags           // Flags)
```

See manual with examples:

https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-portal

CNG

- Cryptography API: Next Generation(CNG)
  - since Vista
- Two modes: kernel, user (same API)
  - user mode – CNG provider in Bcrypt.dll
  - kernel mode - *Ksecdd.sys*
- Functions: key funcs, crypto primitives
- Crypto agility:
  - easy to add functions

CAPI vs CNG

- CAPI
  - algs (**numeric constants**) defined in *wincrypt.h*
  - hard to add new algorithm
- CNG
  - algs are **strings**
  - new CSP can be created – no need to sign it by MS
  - possible to query CNG for supported algs
  - Newer algs – NSA Suite B and several others

CNG programming

Most of the CNG APIs require a provider or an object created by a provider.

1. Opening the Algorithm Provider
2. Getting or Setting Algorithm Properties
3. Creating or Importing a Key
4. Performing Cryptographic Operations
5. Closing the Algorithm Provider

CNG: Example

```
BCryptOpenAlgorithmProvider(&hAlg...)
BCryptGetProperty(hAlg,  BCRYPT_BLOCK_LENGTH,  &dwBlockSize...)
//allocate buffer, rounding up to next block size

BCryptGetProperty(hAlg,  BCRYPT_OBJECT_LENGTH, &cbKeyObjectLen...)
//allocate buffer for key object

BCryptGenerateSymmetricKey(hAlg, &hKey...)
BCryptEncrypt(hKey,...)                         //data is now encrypted

BCryptDestroyKey(hKey)
BCryptCloseAlgorithmProvider(hAlg,0)            //deallocate buffers
```