MS Cryptoの10年

過去、現在、未来

渡辺清, CISSP
Security Center of Excellence
マイクロソフト株式会社

本プレゼンテーションについて

- 私個人の見解であり、会社としての総意ではありません。

目次

- 暗号APIの発展
  - CryptoAPI
  - .NET Crypto
  - Next Generation Cryptographic API (CNG)
  - .NET CNG
- これからの10年 - Crypto Agility 取組
  - 暗号は何処へ（開発者としての視点）
  - BRIC諸国
  - ロシアの例
    - Crypto Agilityでいい点
    - Crypto Agilityであまりよくない点
    - Crypto Agilityが難しい点
    - Crypto Agilityが…
  - まとめ- 今後

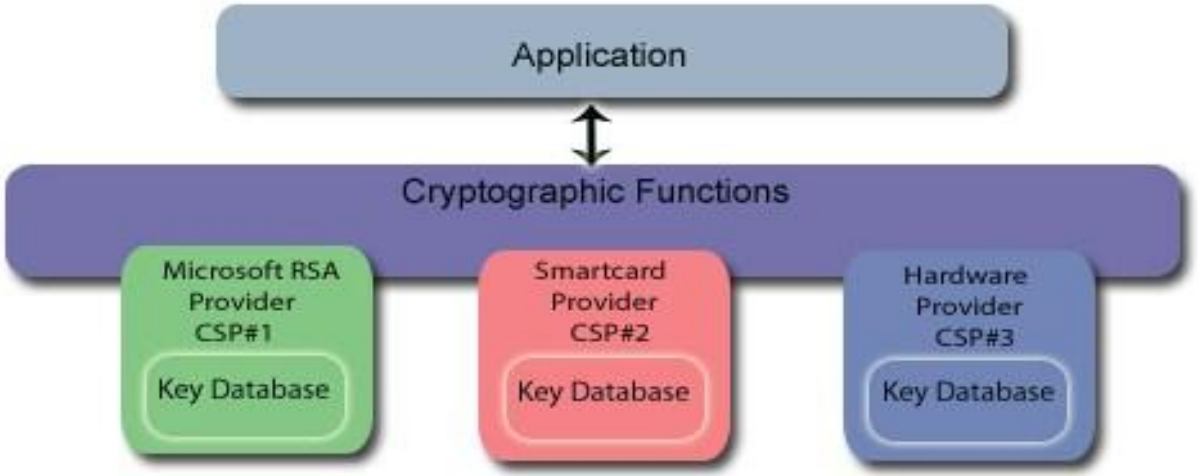
CRYPTOAPI

CryptoAPI

- Microsoft Cryptography API
  - CryptoAPI, 又はCAPI
  - Windows NTで導入 (4.0)
  - WIN32API in ADVAPI.DLL
  - ユーザモードで利用(≠Ring 0)
  - CryptoAPIは、CSPと実アプリの橋渡し
  - Windows NT 4.0 Cryptography Functions
    - Context Functions
      - CryptAcquireContextA/W
    - Key Generation Functions
    - Data Encryption Functions
    - Key Exchange Functions
    - Hashing and Signature Functions

Cryptographic Service Provider(CSP)

- CSPはDLL (Software Library)
  - Microsoft Base Cryptographic Provider
  - Microsoft Enhanced Cryptographic Provider
  - Smart Card Cryptographic Service Providers



- CSPは、マイクロソフトにより電子署名
- また、改ざん等検査のため、定期的なスキャン

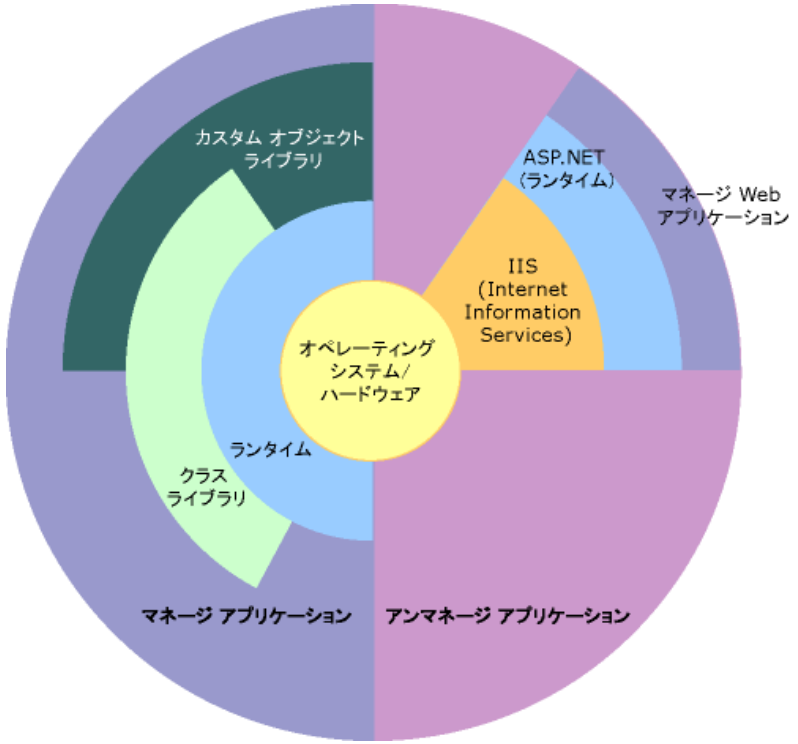
サンプル

```
#include <windows.h>
#include <wincrypt.h>
#include <stdio.h>

#pragma comment(lib, “advapi32.lib”);

void main(int argc, char *argv[]) {
    BOOL fResult;
    HCRYPTPROV hProv;
    HCRYPTKEY hKey;
    DWORD dwFlags, dwBits = 128;
    CHAR Message[8] = “Test”;
    DWORD BufLen = 8, MessageLen = 5;

    fResult = CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV,
                                  PROV_RSA_FULL, CRYPT_VERIFYCONTEXT);

    // Generate an exportable and random 128bit RC2 key
    dwFlags = dwBits << 16 | CRYPT_EXPORTABLE;
    fResult = CryptGenKey(hProv, CALG_RC2, dwFlags, &hKey);

    // Message will contain the encrypted bytes
    fResult = CryptEncrypt(hKey, 0, TRUE, 0, (LPBYTE)Message, &MessageLen, BufLen);

    // Message will contain the original text
    fResult = CryptDecrypt(hKey, 0, TRUE, 0, (LPBYTE)Message, &MessageLen);
    printf(“Message = %s¥n”, Message);
}
```

.NET CRYPTO

.NET 概要(設計思想)

- オブジェクト コードがローカルに保存され実行されるか、インターネットに分散されローカルに実行されるか、リモートで実行されるかにかかわらず、一貫したオブジェクト指向プログラミング環境を提供すること。
- ソフトウェア配置やバージョン管理の競合を最小化するコード実行環境を提供すること。
- 作成者が不明なコードや、完全には信頼できないサード パーティ製のコードも含めて、コードを安全に実行できるようにするためのコード実行環境を提供すること。
- スクリプトやインタープリターが実行される環境でのパフォーマンスの問題を回避するコード実行環境を提供すること。
- Windows ベースのアプリケーションや Windows ベースや Web ベースのアプリケーションなど、多種のアプリケーションの開発において、ユーザーが一貫した方法で作業できること。

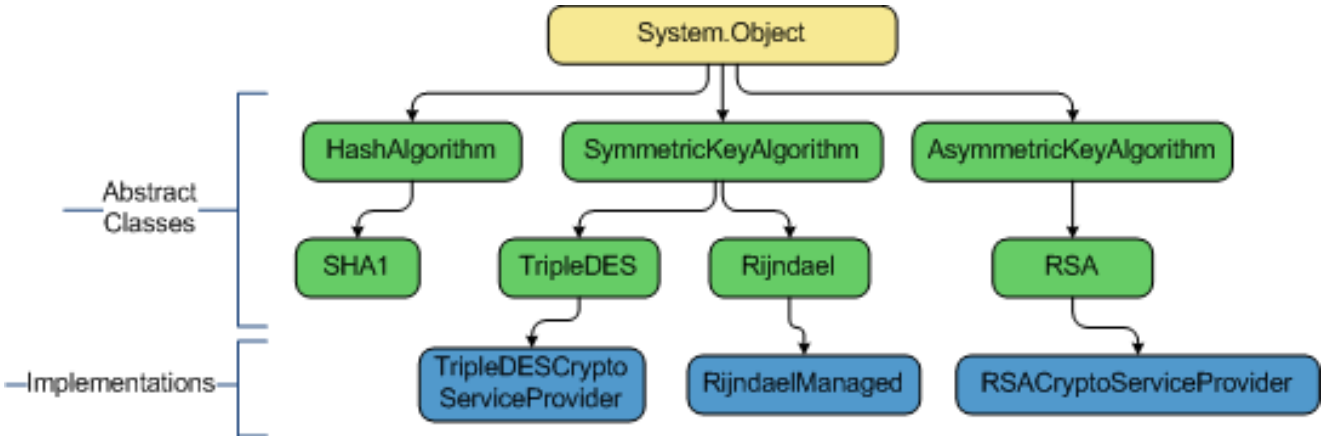


.NET Crypto

- .NET - System.Security.Cryptographyの名前空間に存在
  - Symmetric key encryption/decryption
  - Asymmetric (public/private) key encryption/decryption
  - Hashing
  - Signing
  - Random number generation
  - Misc. other support classes

.NET Crypto

- オブジェクト指向に向くクラス設計
- 多くは単純にCryptoAPIのWrapper、その他は、マネージコードで記述
- クラス階層



.NET Crypto

アルゴリズム	暗号クラス	内容
DES	DESCryptoServiceProvider	DESアルゴリズム 標準CSPのWrapperクラス
RC2	RC2CryptoServiceProvider	RC2アルゴリズム 標準CSPのWrapperクラス
TripleDES	TripleDESCryptoServiceProvider	TripleDESアルゴリズム 標準CSPのWrapperクラス
RSA	RSACryptoServiceProvider	RSAアルゴリズム 標準CSPのWrapperクラス
SHA1	SHA1CryptoServiceProvider, SHA1Managed	アンマネージクラス マネージクラス

サンプル

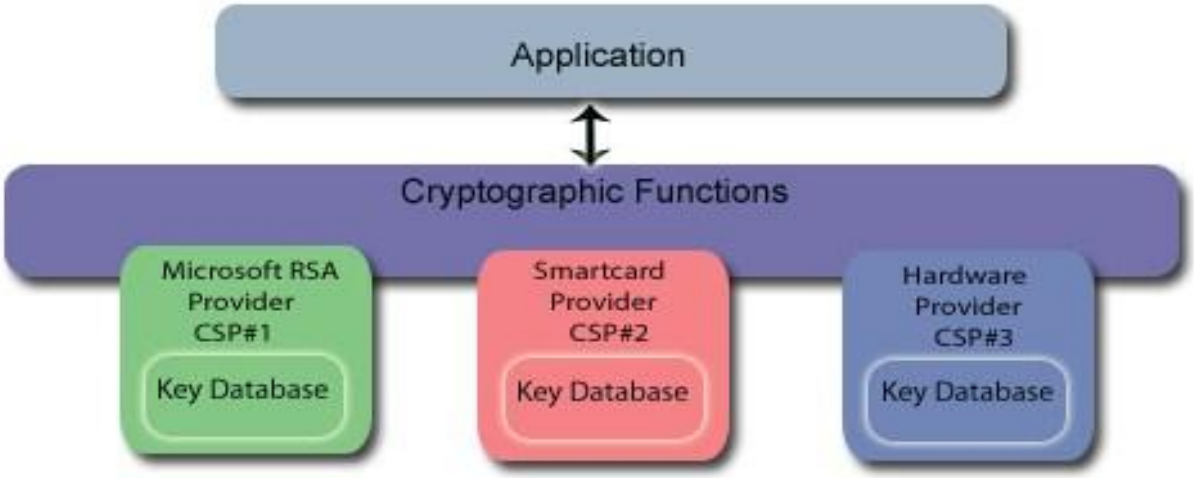
```
static void Main(string[] args)
{
    string PlainText = "My Message.";
    byte[] bPlainText = Encoding.Unicode.GetBytes(PlainText);
    byte[] CipherText;
    byte[] bPlainTextNew;

    TripleDESCryptoServiceProvider TDES = new
    TripleDESCryptoServiceProvider();

    TDES.GenerateKey();
    TDES.GenerateIV();

    ICryptoTransform Encryptor = TDES.CreateEncryptor();
    ICryptoTransform Decryptor = TDES.CreateDecryptor();

    // Encrypt plaintext
    CipherText = Encryptor.TransformFinalBlock(bPlainText, 0, bPlainText.Length);

    // Decrypt ciphertext
    bPlainTextNew = Decryptor.TransformFinalBlock(CipherText, 0,
    CipherText.Length);

     strPT = Encoding.Unicode.GetString(bPlainTextNew);


       Console.WriteLine(strPT);
}
```

CNG (CRYPTOGRAPHY NEXT GENERATION)

CAPIの問題点

- CryptoAPIの歴史をみると…
  - CAPIはNTプラットフォームの拡張だった
    - Lowレベルでのサポート
    - インターネット時代の暗号機能追加
  - 但し、十分な拡張性ではなかった、むしろ必要ではなかった…
    - 暗号は輸出規制の規制対象
    - CAPIは米国当局の認可対象
    - CAPIはこの時代に設計
      - All CSPはマイクロソフトによって署名され、厳しく管理。オープンインターフェイスの性格はなかった(必要とされていた？)。
      - 暗号ファンクション(Crypto Functions)と鍵(Key)は、厳しく管理し、承認したCSPでその鍵を利用する
        - ビジネスを理由に独自アルゴリズムを追加したり、別で作成した鍵を別のCSPの鍵として利用するのは難しかった

CAPIの問題点


Application
Cryptographic Functions
Microsoft RSA Provider CSP#1
Key Database
Smartcard Provider CSP#2
Key Database
Hardware Provider CSP#3
Key Database

なぜCNG?

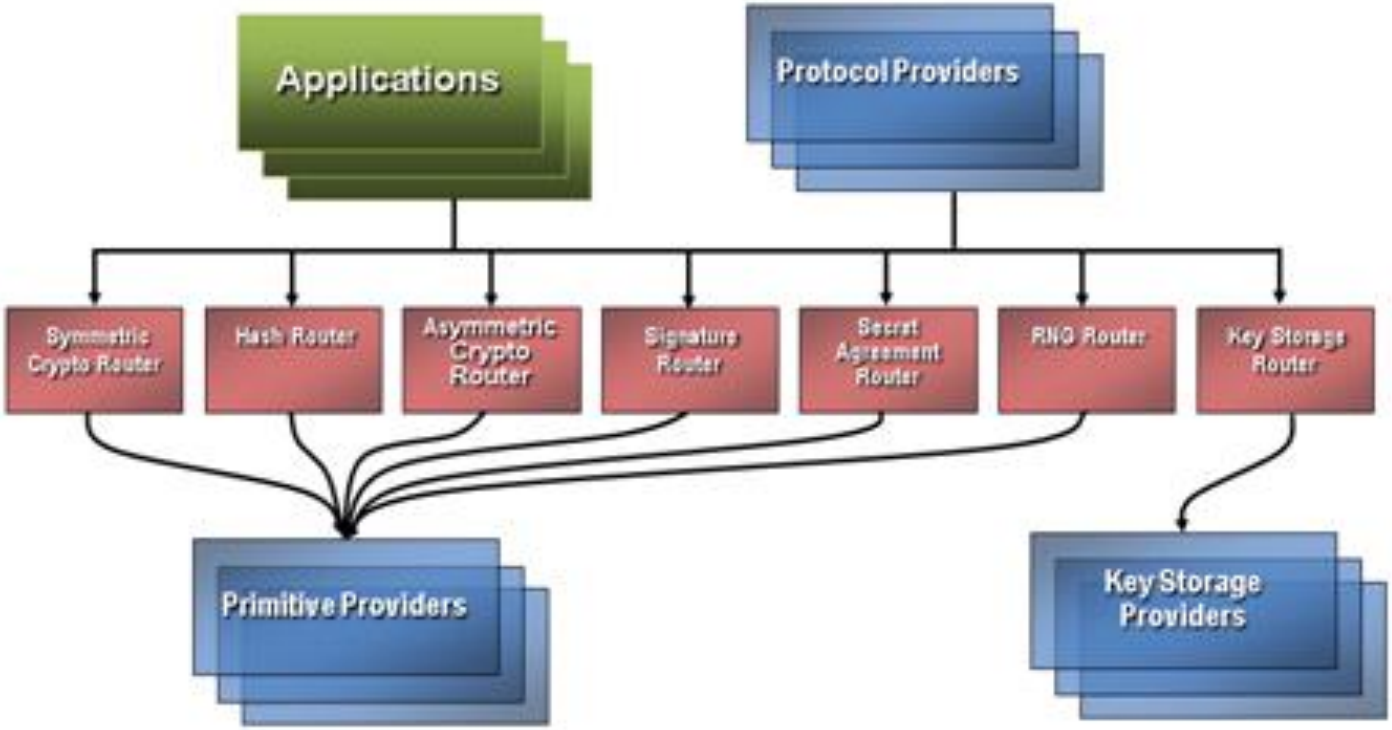
- 2つの主な理由
  - ビジネス
    - Crypto Agility
      - CryptoAPIは、ハードコードされたアルゴリズムのみサポートしていた
      - 顧客やマイクロソフトさえも、規制や承認プロセスのため、簡単にアルゴリズムを追加して、プロトコルやインフラストラクチャに実装するのが難しかった。
      - 世間では、MD5及びSHAコリジョン。柔軟に新しいアルゴリズムの追加が叫ばれる
      - 幾らかの国が独自のアルゴリズムを規格し、実装、展開し、ビジネスに大きなインパクトが出てくるようになる。
    - FIPS 140-2 Suite Bアルゴリズムのサポート
      - AES - AES-128, AES-256, AES-256
      - SHA - SHA-256, SHA-384
      - Key Exchange (ECDH) - P-256, P-384
      - Signature (ECDSA) - P-256, P-384
  - 技術
    - 拡張性
    - カーネルモードサポート(プロトコルサポート)
    - メインテナンスおよび保守

CNG特徴

- Crypto Agilityの対応への第一歩
- Kernel Modeへの対応
  - 同じAPIでUser ModeとKernel Mode対応
- コモンクライテリアへの対応
- CAPI1.0でサポートされるアルゴリズムの継続サポート
- パフォーマンス改善
- 楕円暗号実装
- Key IsolationとKey Storageの統合

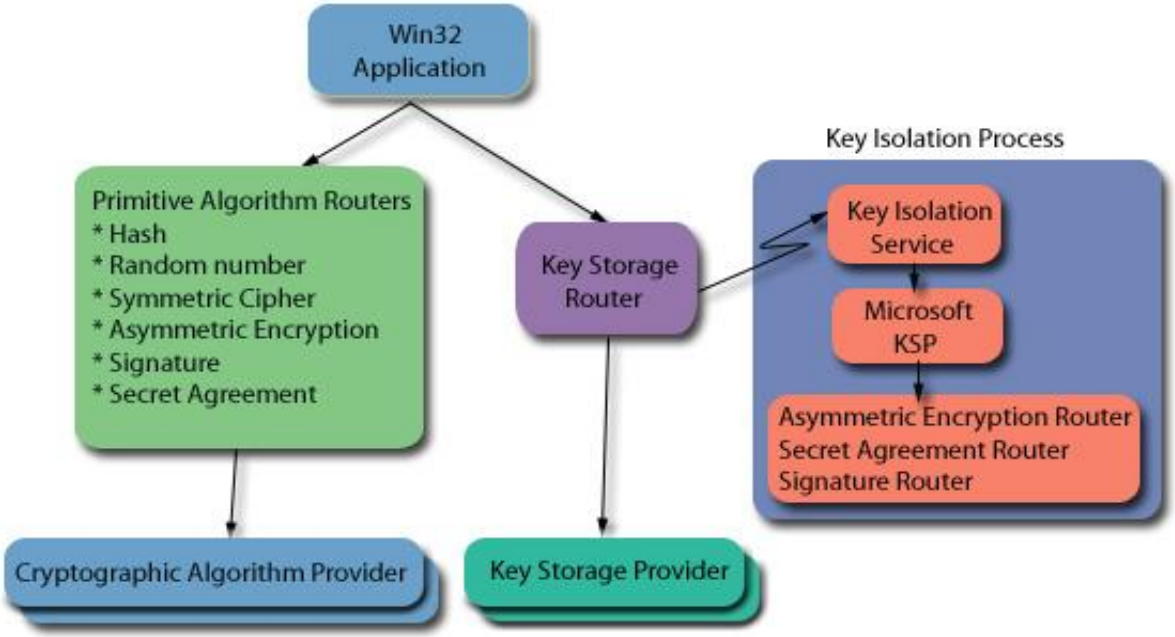
Crypto Architecture

- CNG Plug-inの3つのレイヤ



CNG Router-Providerモデル

- CNGはRouter-Providerモデル
- CNG APIを利用し、Routerを呼び出す
- RouterがアルゴリズムProviderを探し出す



プログラムの流れ

- 共通ステップ
- アプリケーションは、Crypto Objectのメモリ管理を実施
- RouterとProviderのメソッド
  - BCryptOpenAlgorithmProvider
    - アルゴリズム名とrouterを設定
  - BCryptGetProperty
    - プロバイダーを探す

CNGのサポート

- カーネルモードは制限やサポートされていない箇所が存在

.NET CNG

- .NET 3.5が２つの新クラスサポート
  - FIPS対応暗号アルゴリズム
  - Suite Bの対応
  - .NET CNGクラスはCNGのWrapper
  - 機能
    - Cryptographic Primitives
    - Secret-Key Encryption
    - Public-Key Encryption
    - Digital Signatures
    - Hash Values
    - Random Number Generation
    - ClickOnce Manifests
    - Suite B Support
    - Cryptography Next Generation (CNG) Classes

Crypto Agilityの実際

これからの今後の10年

暗号は何処に向かうのか？

- 輸出規制の時代を経て…
- インターネット時代
  - クラウドへ…
- 暗号アルゴリズムの脆弱性
- 暗号の国際標準化
- BRIC諸国暗号政策

- また、暗号2010年問題
  - 対応はもう始まってます

BRIC諸国の暗号政策

- ブラジル、ロシア、中国、インド
  - 管理政策
    - 中国: regulation No. 273,
    - ロシア: FSB certification
    - インド：Crypto import and usage 規制(ICT)
    - ブラジル：推奨アルゴリズム
  - 技術
    - ロシア: GOST
    - 中国: 独自暗号、 Trusted Cryptographic Module (TCM, 2007)
  - 標準
    - 中国: GB/T 17895:1999 – Multi-level Protection Scheme, GB/T 20008:2005 – OS security eval crit, GB/T 20009:2005 – DBMS security eval criteria
    - ロシア: FSTEC sec products eval criteria, FSTEC system attestation

暗号政策

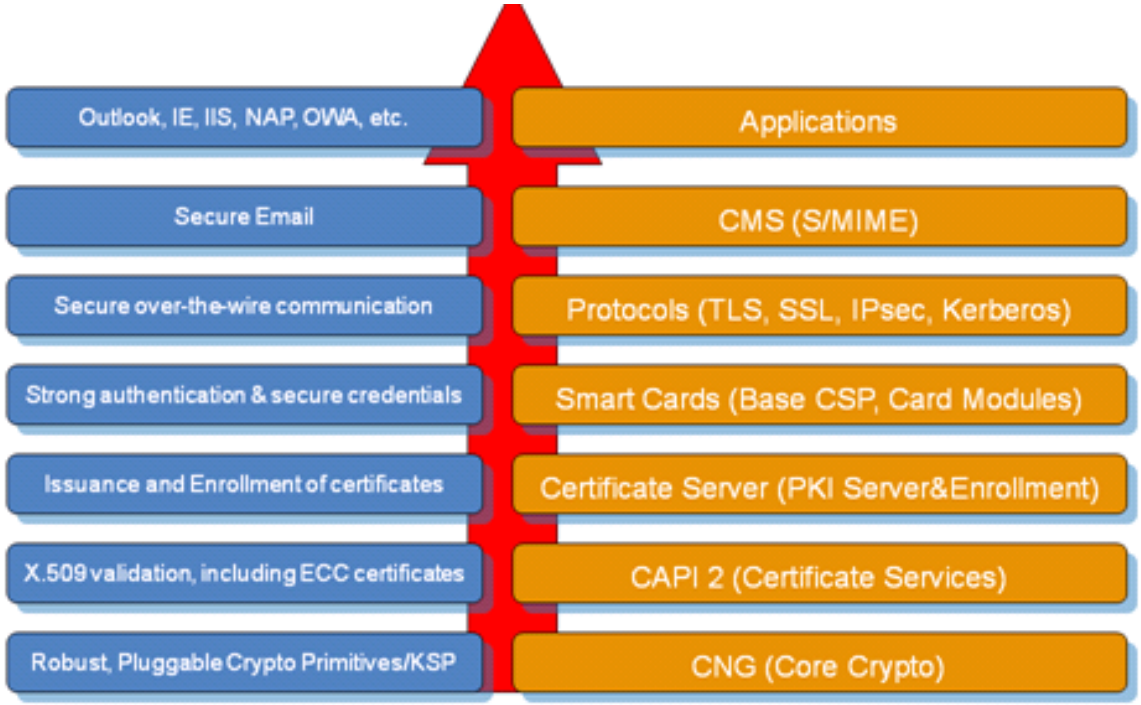
- 政府が暗号製品のR&D,販売,利用をコントロール
- Certified製品のみが政府機関での利用を許可
  - ロシア
    - GOST family of 暗号アルゴリズム(PRNG, hash, symm encryption, HMAC, EC-based DigSig and DH asymm crypto)
  - 中国
    - 独自暗号(PRNG, encryption, HMAC, hash, DigSig)
    - カスタム Trusted Crypto Module (TPM vNext)
    - Approved for national standardization in Apr 2008
  - 韓国
    - SEED symm アルゴリズム - rfc4269.txt

ロシアの実際

- 顧客からの要求
  - オフィース製品にGOSTを搭載
    - Word, Excel, Outlook, InfoPath, SharePoint
  - ネットワークセキュリティにGOSTを搭載
  - ADRMS(DRM製品)のGOSTサポート
  - GOSTベースのMicrosoft CA
  - GOSTを利用したSQLデータベース暗号、等

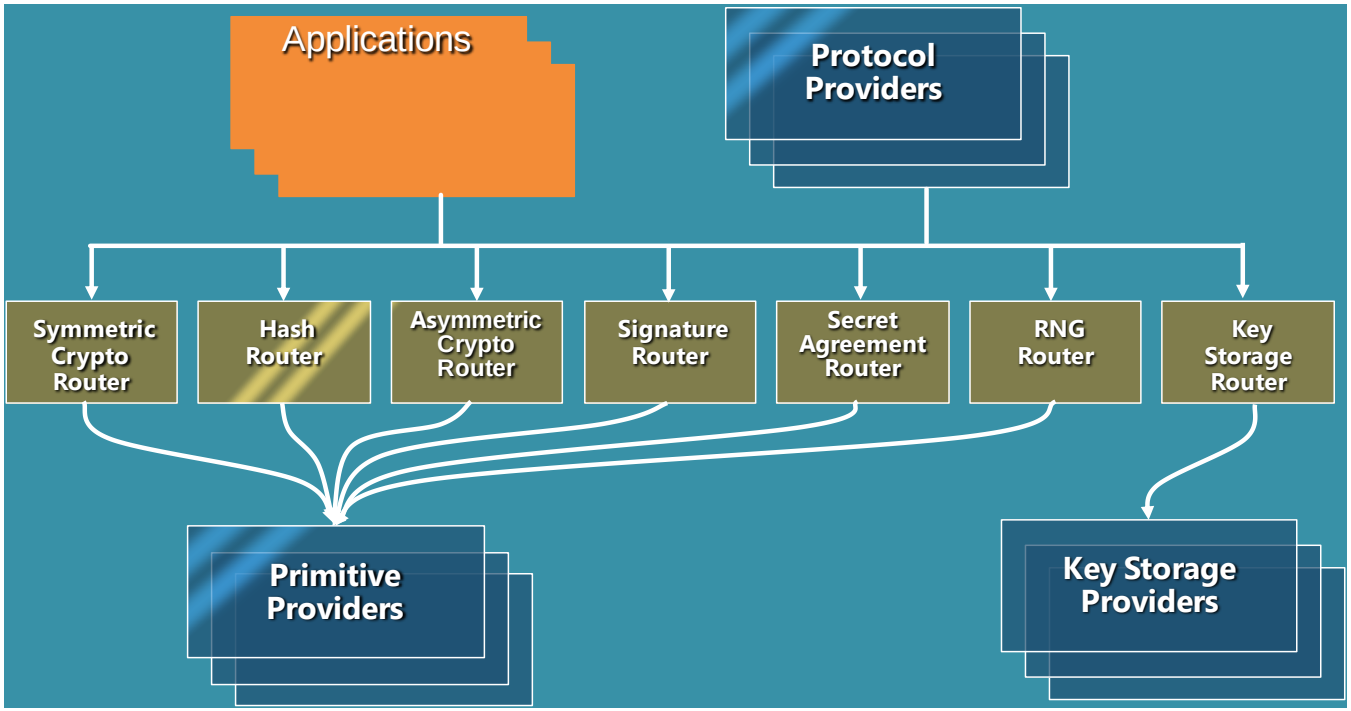
Crypto Agilityをレイヤで考えると

- アプリケーションからは数レイヤを考慮する必要あり。
  - CNGは一番下のレイヤ

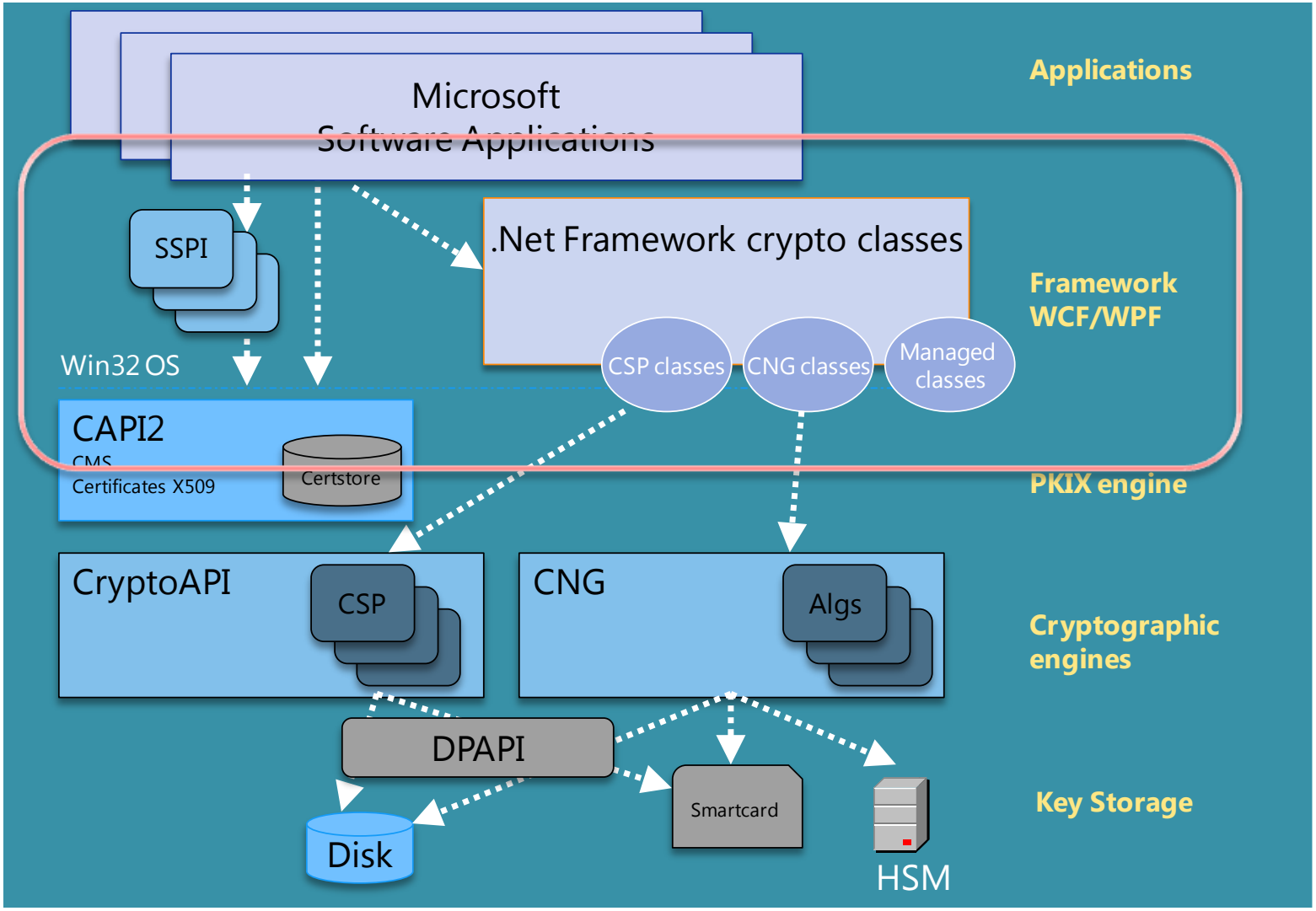


良い：CNG

- 制限はあるが、CryptoAgilityをサポート
- CAPIのSupersetで開発も容易



あまり良くない：プロトコルレベル



難しい：IIS

- 状況
  - MSの主力Web Server
  - IISは、SchannelとKernel ModeのTLSに依存
- 難しい点
  - TLS自体は拡張できるが、Schannelに他のアルゴリズムの利用設定が難しい

難しい：IPsec

- 状況
  - ネットワークセキュリティのコア技術
- 難しい点
  - IKEの実装で、鍵交換時に、他のアルゴリズム指定が難しい
    - ipsecsnp/ipsecsvc mngm & policy modules have hardcoded algos
    - Oakley (IKE phases 1 and 2) has numerous algo checks
    - ipsec.sys (ESP и AH) has hardcoded algos
    - (Win7) IKEEXT.dll and tcpip.sys (ESP и AH) have hardcoded checks for crypto algorithms

難しい：オフィス製品

- 状況
  - OpenXML フォーマット, XML Dig (Office 2010 ではXAdES)
  - Office 2007 SPでCNGサポート
- 難しい点
  - ハードコードアルゴリズムが幾つか記述されている

…：ADRMS & BitLocker

- 拡張性が現状ない

今後のCrypto 10年 – 私見（開発からの見地）

- 暗号の標準化と危殆化が続く（だろう）
- プロトコルレベルでの標準化も続く
- ローカルな暗号アルゴリズムや規制の存続？
- ソフトウェア開発や運用 – Crypto Agility
  - 標準化暗号の実装、サポート
  - 移行運用の経験値が吉とでるか？
  - Crypto Agilityは次期製品群の主要検討項目
  - ライブラリのCrypto Agilityだけでは保障されない
  - 設計、開発過程で、組織全体として取り組む必要(SDL)
  - Crypto Agilityのエコシステム
- 標準対応とローカル対応の狭間