.NET 10 LTS · Январь 2026

Что нового в System.Security.Cryptography

Breaking Changes → Улучшения → PQC

⚠️ Breaking Changes | 🔍 SHA-256 Thumbprints | 🔑 AES Key Wrap | 🛡️ Post-Quantum

Руслан Каменский

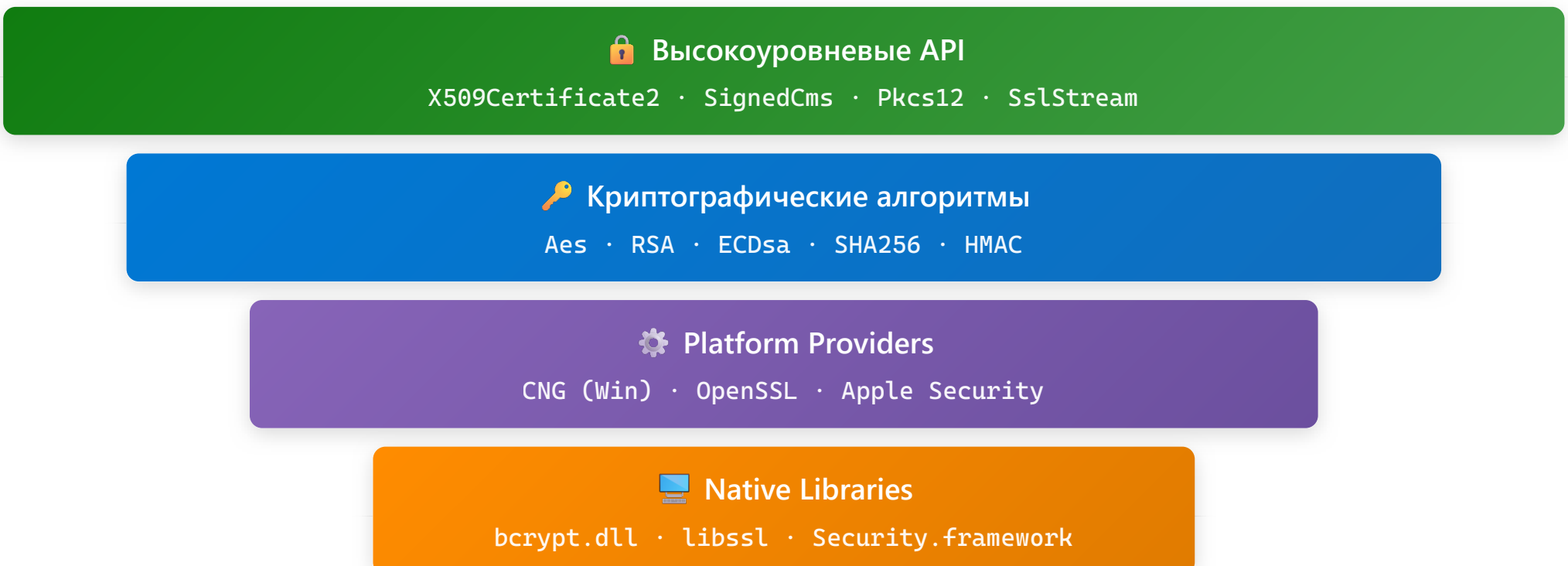
Сравнение моделей криптографии

Язык	Подход	Реализация	Особенности
.NET	Делегирование ОС	`CNG / OpenSSL / Apple`	⚠️ Платформа
Java	JCA/JCE провайдеры	`SunJCE / BouncyCastle`	✅ Pluggable ✅ Кроссплатф.
Python	Обёртки OpenSSL	`cryptography / PyCrypto`	⚠️ Много библиотек
Go	Своя реализация	`crypto/* (stdlib)`	✅ Единообразие

Плюс .NET: Сертифицированная криптография ОС

Минус .NET: Различия между платформами

Модель криптографии в .NET



Linux: OpenSSL 1.1.1+

OpenSSL 1.0.x больше не поддерживается

> ⚠️ **Breaking Change:** проверьте версии OpenSSL в ваших контейнерах и окружениях

macOS без OpenSSL

PQC на Mac не работает

> ⚠️ **Breaking Change:** OpenSSL через HomeBrew больше не поддерживается

Сертификаты X.509

Что это?

- → Публичный ключ + метаданные
- → Владелец (Subject)
- → Издатель (Issuer)
- → Срок действия
- → Цифровая подпись CA

Зачем?

- → HTTPS (TLS/SSL)
- → Подпись кода
- → Email шифрование
- → VPN аутентификация
- → Взаимная аутентификация

Форматы хранения

Как сертификаты хранятся на диске

PEM

Текстовый формат

```
-----BEGIN CERTIFICATE-----
MIIDXTCCAkWgAwIBAgIJ...
-----END CERTIFICATE-----
```

Использование: Linux, веб-серверы

PFX/PKCS#12

Бинарный контейнер

📜 Сертификат
🔑 Приватный ключ
🔐 Защита паролем

Использование: Windows, HTTPS

DER

Бинарный формат

Компактное представление

Использование: Java, Android

Как найти сертификат?

Thumbprint = отпечаток сертификата

Что такое Thumbprint?

Это хэш всего содержимого сертификата.

Используется как уникальный идентификатор.

Проблема SHA-1

SHA-1 считается устаревшим и небезопасным.

Переход на SHA-256.

```
// До .NET 10: только SHA-1
var cert = store.Certificates.Find(X509FindType.FindByThumbprint, thumbprint, false);

// .NET 10: любой алгоритм хеширования
var certs = store.Certificates.FindByThumbprint(HashAlgorithmName.SHA256, sha256Thumbprint);
```

X500DistinguishedName

Валидация стала строже

```
// .NET 10: выбросит CryptographicException
new X500DistinguishedName("Phone=!!");

// Используйте X500DistinguishedNameBuilder:
var builder = new X500DistinguishedNameBuilder();
builder.Add("2.5.4.20", "000-555-1234", UniversalTagNumber.UTF8String);
```

⚠️ **Breaking Change:** проверьте генерацию имён самоподписанных сертификатов

PemEncoding.FindUtf8

Меньше аллокаций при чтении сертификатов

```
// .NET 10: без копирования byte[] → char[]
byte[] fileContents = File.ReadAllBytes(path);
PemFields fields = PemEncoding.FindUtf8(fileContents);
```

Современный PFX

AES-256 вместо 3DES из 1970-х

```
var options = new Pkcs12ExportOptions
{
    EncryptionAlgorithm = PbeEncryptionAlgorithm.Aes256Cbc
};
byte[] pfx = cert.ExportPkcs12(password, options);
```

AES Key Wrap (AES-KW)

Безопасная передача ключей шифрования

Что это?

- → Алгоритм для защиты ключей
- → "Оборачивает" один ключ другим
- → Используется при передаче
- → Стандарт RFC 3394

Зачем?

- → Безопасная передача ключей
- → Хранение в базах данных
- → Интеграция с внешними API
- → HSM и key management

AES Key Wrap with Padding

RFC 5649 — работает с любой длиной

```
using Aes aes = Aes.Create();
aes.Key = kek; // Key Encryption Key

byte[] wrapped = aes.EncryptKeyWrapPadded(dataKey);
byte[] unwrapped = aes.DecryptKeyWrapPadded(wrapped);
```

В .NET 10: добавлены методы EncryptKeyWrapPadded/DecryptKeyWrapPadded для работы с ключами любой длины

Post-Quantum

Готовимся к квантовым компьютерам

ML-DSA

Цифровые подписи (FIPS 204)

EXPERIMENTAL

```
using MLDsa key = MLDsa.GenerateKey(MLDsaAlgorithm.MLDsa65);

byte[] signature = key.SignData(data);
bool valid = key.VerifyData(data, signature);
```

ML-KEM

Обмен ключами (FIPS 203)

EXPERIMENTAL

```
using MLKem kem = MLKem.GenerateKey(MLKemAlgorithm.MLKem768);

kem.Encapsulate(out ciphertext, out secret);
byte[] s = kem.Decapsulate(ciphertext);
```

SLH-DSA

Подписи на хешах (FIPS 205)

EXPERIMENTAL

```
using SlhDsa key = SlhDsa.GenerateKey(SlhDsaAlgorithm.SlhDsaSha2_128s);

byte[] signature = key.SignData(data);
bool valid = key.VerifyData(data, signature);
```

Что делать завтра?

- → Проверить OpenSSL на Linux
- → Обновить thumbprints на SHA-256
- → Попробовать PQC (если не Mac)

Хотите глубже?

Посмотрите полный доклад
«Криптография в .NET. Где заканчиваются гарантии безопасности»

Ссылка: youtube.com/watch?v=qaSJGTTfADM

Спасибо за внимание! 👋

Материалы по докладу — сканируйте QR:

📖 What's new

⚠️ Breaking Changes

💻 runtime repo

До встречи на SpbDotNet! 🎉

Квантовые компьютеры не страшны — страшно забыть обновиться 😊

Руслан Каменский