Дослідження реалізацій протоколу SSL

Недождій Максим, Буржимський Ростислав

ФІ-42мн

Мета

Дослідження особливостей реалізації криптографічних механізмів протоколу SSL/TLS

1 Структура протоколів SSL та TLS

1.1 Багаторівнева архітектура

Протокол TLS має чітко визначену багаторівневу структуру. На найнижчому рівні розташовується надійний транспортний протокол TCP. Над ним працює TLS Record Protocol, який відповідає за:

- Фрагментацію даних (максимум 16384 байти)
- Компресію (виведений з використання через CRIME/BREACH атаки)
- Обчислення MAC для цілісності
- Шифрування з використанням узгоджених алгоритмів

Record Protocol надає послуги чотирьом протоколам вищого рівня:

1. Handshake Protocol – встановлення безпечного з'єднання, узгодження криптографічних параметрів та автентифікація
2. Change Cipher Spec Protocol – сигналізує про перехід на узгоджені параметри безпеки
3. Alert Protocol – повідомлення про помилки та закриття з'єднання
4. Application Data Protocol – передача даних додатку

1.2 TLS Record Protocol

Структура запису:

Поле	Опис
Content Type	1 байт: 20 (ChangeCipherSpec), 21 (Alert), 22 (Handshake), 23 (AppData)
Protocol Version	2 байти: 0x0303 для TLS 1.2
Length	2 байти: довжина фрагмента (макс. 16384)
Fragment	Дані + MAC (у TLS 1.2) або AEAD tag (у TLS 1.3)

1.3 Handshake Protocol: повне встановлення з'єднання (TLS 1.2)

Фаза 1: Обмін привітаннями

1. ClientHello → Сервер:
   - Максимальна версія TLS
   - 32-байтове випадкове число (4 байти timestamp + 28 випадкових)
   - Session ID (порожній для нового сеансу)
   - Список підтримуваних блоків шифрування
   - Методи компресії та розширення
2. ServerHello ← Сервер:
   - Обрана версія протоколу
   - 32-байтове випадкове число сервера
   - Session ID (новий або відновлюваний)
   - Обраний блок шифрування та метод компресії

Фаза 2: Автентифікація сервера та обмін ключами

3. Certificate ← Сервер: ланцюжок сертифікатів X.509
4. ServerKeyExchange ← Сервер (для DHE/ECDHE):
   - Параметри DH: $(p, g, g^x \bmod p)$ або ECDH: $(curve, G, d_s \cdot G)$
   - Цифровий підпис параметрів приватним ключем сервера
5. CertificateRequest ← Сервер (опціонально)
6. ServerHelloDone ← Сервер

Фаза 3: Автентифікація клієнта

7. Certificate → Сервер (якщо запитано)
8. ClientKeyExchange → Сервер:
   - RSA: зашифрований 48-байтовий PreMasterSecret
   - DHE/ECDHE: публічне значення клієнта $(g^y \bmod p)$ або $(d_c \cdot G)$
9. CertificateVerify → Сервер (якщо надано сертифікат)

Фаза 4: Фіналізація

10. ChangeCipherSpec → Сервер
11. Finished → Сервер: verify_data =
    = PRF(master_secret, "client finished Hash(handshake_messages))
12. ChangeCipherSpec ← Сервер
13. Finished ← Сервер: verify_data =
    = PRF(master_secret, "server finished Hash(handshake_messages))

Виведення ключів:

1.4 Відновлення існуючого сеансу (TLS 1.2)

Session ID Resumption:

1. ClientHello → Сервер: включає збережений Session ID
2. ServerHello ← Сервер: підтверджує той самий Session ID
3. Обидві сторони генерують нові ключі сеансу з існуючого master_secret
4. ChangeCipherSpec + Finished
5. Передача даних

Переваги: 1-RTT замість 2-RTT, немає дорогих операцій з відкритим ключем.
Session Tickets (RFC 5077):

- Сервер інкапсулює стан сеансу, шифрує його та надсилає клієнту
- Клієнт зберігає непрозорий ticket
- При відновленні клієнт включає ticket у розширення SessionTicket
- Сервер розшифровує ticket та витягує master_secret
- Не потребує серверного кешу сеансів

1.5 TLS 1.3: скорочене встановлення з'єднання

1-RTT Handshake:

1. ClientHello → Сервер: включає key_share для ECDHE
2. Клієнт ← ServerHello + EncryptedExtensions + Certificate + CertificateVerify + Finished: всі повідомлення після ServerHello шифруються
3. Finished → Сервер
4. Передача даних

0-RTT Resumption (TLS 1.3):

- Клієнт надсилає дані додатку у першому повідомленні
- Шифрується з ключами з попереднього сеансу
- Обмеження: немає forward secrecy, вразливо до replay attacks
- Дозволено лише для ідемпотентних операцій

2 Порівняльний аналіз версій SSL/TLS

Версія	Рік випуску	Основні нововведення	Вразливості та статус
SSL 1.0	1994 (не випущено)	–	Критичні вразливості безпеки, відсутність номерів послідовності
SSL 2.0	1995	Перший публічний реліз	Недоліки: ідентичні ключі для MAC та шифрування, слабка конструкція MAC (MD5 з префіксом), незахищене встановлення з'єднання, вразливість до downgrade атак, використання TCP close як сигнал завершення. Статус: Застарілий (RFC 6176, 2011)
SSL 3.0	1996 (RFC 6101)	Розділені ключі MAC та шифрування, покращена конструкція MAC, автентифікація handshake, явне повідомлення close_notify	POODLE (2014): padding oracle атака на CBC, експлуатація через protocol downgrade. Статус: Заборонено (RFC 7568, 2015)
TLS 1.0	1999 (RFC 2246)	HMAC замість власного MAC, PRF з MD5+SHA1, посилена перевірка сертифікатів, несумісність з SSL 3.0	BEAST (2011): CBC IV chaining, передбачувані вектори ініціалізації. Підтримка слабких блоків шифрування (RC4, DES). Статус: Застарілий (2020)
TLS 1.1	2006 (RFC 4346)	Явні IV для CBC, покращена обробка padding errors, зміни щодо передчасного закриття	Продовжує підтримувати RC4, DES, 3DES. Немає обов'язкового forward secrecy. Статус: Застарілий (2020)

Версія	Рік випуску	Основні нововведення	Вразливості та статус
TLS 1.2	2008 (RFC 5246)	Гнучка PRF, AEAD підтримка (AES-GCM, ChaCha20-Poly1305), узгодження алгоритмів підпису, видалення MD5/SHA1 з PRF	CRIME (2012): компресія TLS. FREAK/Logjam (2015): експорт блоків шифрування, слабкі DH групи. Lucky13 (2013): timing атака на CBC+HMAC. Статус: Активно використовується
TLS 1.3	2018 (RFC 8446)	1-RTT handshake, 0-RTT resumption, обов'язковий forward secrecy, шифрування handshake після ServerHello, 5 блоків шифрування, видалено: RSA key exchange, CBC mode, RC4, DES, 3DES, MD5, SHA1, компресію, renegotiation	Режим 0-RTT вразливий до replay (обмежено ідемпотентними операціями). Статус: Сучасний стандарт

Ключові тенденції еволюції:

- Безпека: від слабких MAC до HMAC до AEAD, від передбачуваних IV до явних IV
- Forward Secrecy: від RSA key exchange до обов'язкового ефемерного DH
- Продуктивність: від 2-RTT до 1-RTT до 0-RTT
- Мінімалізм: від багатьох блоків шифрування до 5 суворо перевірених
- Приватність: від відкритого handshake до зашифрованого (крім ClientHello/ServerHello)

3 Структура сертифікатів X.509 та PKI

3.1 Структура сертифіката X.509 v3

Сертифікат X.509 складається з трьох основних компонентів:

1. tbsCertificate: дані для підпису

2. signatureAlgorithm: OID алгоритму підпису (напр. sha256WithRSAEncryption)

3. signatureValue: цифровий підпис CA

Структура tbsCertificate:

Поле	Опис
version	0 (v1), 1 (v2), 2 (v3). Якщо є extensions → v3
serialNumber	Унікальне ціле число (макс. 20 октетів, $\geq$ 64 біти ентропії). Використовується в CRL та OCSP
signature	AlgorithmIdentifier
issuer	Distinguished Name CA
validity	notBefore та notAfter (UTCTime або GeneralizedTime). Макс. 398 днів для публічних TLS сертифікатів (з 2020)
subject	Distinguished Name суб'єкта. Може бути порожнім якщо є критичне subjectAltName
subjectPublicKeyInfo	алгоритм (RSA, ECDSA, Ed25519) + параметри + subjectPublicKey
issuerUniqueID	(v2+, застарілий)
subjectUniqueID	(v2+, застарілий)
extensions	(v3) Набір розширень з OID, critical flag, та extnValue

3.2 Стандартні розширення X.509 v3

Розширення	Critical	Опис
basicConstraints	Так	CA: TRUE/FALSE, pathLenConstraint: макс. кількість проміжних CA
keyUsage	Так	9 бітів: digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly
subjectKeyIdentifier	Ні	Унікальний ідентифікатор відкритого ключа
authorityKeyIdentifier	Ні	Ідентифікатор ключа CA, що підписав сертифікат
subjectAltName	Ні	DNS імена, Email, IP адреси, URI. Основний ідентифікатор для TLS
extendedKeyUsage	Ні	Спеціалізовані використання: serverAuth, clientAuth, codeSigning, emailProtection, timeStamping, OCSPSigning
certificatePolicies	Ні	OID політик видачі CA, CPS URI, user notices
cRLDistributionPoints	Ні	URI для завантаження CRL

Розширення	Critical	Опис
authorityInfoAccess	Ні	OCSP responder URI, CA Issuers URI
nameConstraints	Так	Обмеження на DNS імена, email, IP для підлеглих CA

3.3 Ієрархія PKI та ролі центрів сертифікації

Типи CA:

1. Root CA:
   - Самопідписаний сертифікат
   - Термін дії: 15-25 років
   - Офлайн, фізично захищений HSM
   - Включений у trust stores (Mozilla NSS, Microsoft, Apple, Google Chrome)
   - Підписує проміжні CA
2. Intermediate CA:
   - Підписаний Root CA або іншим Intermediate CA
   - Термін дії: 5-10 років
   - Ізолює Root CA від щоденних операцій
   - Може бути Policy CA (політики безпеки) або Issuing CA (видача сертифікатів)
3. Issuing CA:
   - Онлайн CA для видачі кінцевих сертифікатів
   - Термін дії: 2-5 років
   - Підписує сертифікати серверів, користувачів, пристроїв

Ланцюжок довіри:

$$\text{End-Entity Cert} \xleftarrow{\text{підписаний}} \text{Issuing CA} \xleftarrow{\text{підписаний}} \text{Intermediate CA} \xleftarrow{\text{підписаний}} \text{Root CA}$$

Валідація ланцюжка (RFC 5280):

1. Перевірка підписів: signature кожного cert верифікується з subjectPublicKeyInfo наступного
2. Перевірка термінів дії: notBefore $\leq$ поточний час $\leq$ notAfter
3. Перевірка basicConstraints: cA=TRUE для всіх проміжних CA, pathLenConstraint
4. Перевірка keyUsage: keyCertSign для CA
5. Перевірка відкликання: CRL або OCSP
6. Перевірка nameConstraints (якщо є)
7. Кореневий CA в trust store

3.4 Акредитовані CA: вимоги та стандарти

WebTrust для CA (AICPA/CPA Canada):

- 5 принципів: бізнес-практики, цілісність транзакцій, конфіденційність, безпека систем, операційна доступність
- Щорічні аудити ліцензованими CPA
- Стандарти: ISAE 3000, SSAE 18, CSAE 3000/3001
- Обов'язково для включення до Mozilla/Microsoft/Apple/Chrome Root Programs

CA/Browser Forum Baseline Requirements:

- Мінімальні стандарти для публічно довірених TLS сертифікатів
- Серійні номери: ≥ 64 біти ентропії
- Validity period: ≤ 398 днів (з вересня 2020, раніше 825 днів)
- Методи валідації домену:
  - Email: підтвердження на admin@, postmaster@, webmaster@domain
  - DNS: TXT запис з випадковим токеном
  - HTTP: файл з токеном на /.well-known/pki-validation/
  - TLS ALPN Challenge (RFC 8737)
- Організаційна валідація: перевірка юридичних документів, бізнес-реєстрів, телефонного підтвердження
- Управління ключами: HSM з FIPS 140-2 Level 2+ для Root CA, Level 3+ рекомендовано
- Відкликання протягом 24 годин при компрометації

Програми довірених коренів:

- Mozilla NSS: ∼150 Root CA, Mozilla Root Store Policy
- Microsoft Trusted Root Program: 200+ Root CA
- Apple Root Certificate Program
- Google Chrome Root Store: окремий з 2021 (раніше використовував NSS/Windows)

3.5 Certificate Revocation List

Структура X.509 v2 CRL:

- version: 0 (v1) або 1 (v2)
- signature: AlgorithmIdentifier
- issuer: DN CA, що видав CRL
- thisUpdate: дата генерації CRL

- nextUpdate: дата наступного CRL
- revokedCertificates: список відкликаних сертифікатів
  - serialNumber: серійний номер сертифіката
  - revocationDate: дата відкликання
  - crlEntryExtensions: reasonCode (11 кодів: keyCompromise, cACompromise, affiliationChanged, superseded, cessationOfOperation, certificateHold, removeFromCRL, privilegeWithdrawn, aACompromise, ...), invalidityDate
- crlExtensions: authorityKeyIdentifier, cRLNumber, deltaCRLIndicator, issuingDistributionPoint
- signatureValue: цифровий підпис CA

Типи CRL:

- Full CRL: повний список всіх відкликаних сертифікатів
- Delta CRL: містить лише зміни з моменту Base CRL (менший розмір, швидше завантаження)
- Indirect CRL: CRL підписаний іншим CA

Розповсюдження:

- Розширення cRLDistributionPoints у сертифікаті містить URI (HTTP, LDAP, FTP)
- Клієнт завантажує CRL, перевіряє підпис CA, кешує до nextUpdate
- Розмір: від кількох КБ до десятків МБ для великих CA

3.6 Online Certificate Status Protocol

Структура OCSP запиту (RFC 6960):

- CertID:
  - hashAlgorithm (SHA-1, SHA-256)
  - issuerNameHash = Hash
  - issuerKeyHash = Hash
  - serialNumber
- tbsRequest: версія, requestorName (опціонально), requestList (може містити декілька CertID)
- optionalSignature

Структура OCSP відповіді:

- responseStatus: successful, malformedRequest, internalError, tryLater, sigRequired, unauthorized
- responseBytes:
  - producedAt: час генерації відповіді

  - responses: для кожного CertID:
    - certStatus: good, revoked (з revocationTime та revocationReason), unknown
    - thisUpdate, nextUpdate (опціонально)
  - сертифікат OCSP responder
- signatureAlgorithm + signature: підпис CA або делегованого OCSP responder

OCSP Stapling (RFC 6066):

- Сервер періодично запитує OCSP відповідь для свого сертифіката
- Кешує підписану відповідь (до nextUpdate)
- Надсилає ("staples") OCSP відповідь у TLS handshake
- Переваги: зменшення затримки, конфіденційність, масштабованість

Порівняння CRL vs OCSP:

Характеристика	CRL	OCSP
Частота оновлень	Періодична (години/дні)	Реальний час
Розмір даних	Великий (КБ-МБ)	Малий (КБ)
Автономна робота	Так (кешування)	Ні (онлайн)
Затримка	Низька (локально)	Мережевий round-trip
Конфіденційність	Краща (не відстежує)	Гірша (CA знає запити)
Масштабованість	Краща (CDN)	Вузьке місце responder

Сучасні тенденції (2024-2025):

- Let's Encrypt припиняє підтримку OCSP (2025), покладається на короткі терміни сертифікатів
- Chrome впроваджує CRLite (стиснуті Bloom filters для CRL)
- OCSP soft-fail: браузери продовжують при відмові OCSP (для доступності)
- OCSP Stapling стає стандартом
- Тренд: ≤ 90 днів validity $\rightarrow$ менша потреба у відкликанні

4 Криптографічні методи SSL/TLS

4.1 Методи узгодження ключів

Метод	Механізм	Forward Secrecy	Характеристики
RSA Key Exchange	Клієнт генерує 48-байтовий PreMasterSecret (2 байти version + 46 random), шифрує відкритим RSA ключем сервера, надсилає у ClientKeyExchange	Ні	Найшвидший. Якщо приватний ключ скомпрометовано, то весь минулий трафік розшифрується. Застарілий у TLS 1.3.
DHE	Сервер: обирає (p, g), генерує x, обчислює $X = g^x \bmod p$, підписує (p, g, X), надсилає ServerKeyExchange. Клієнт: генерує y, обчислює $Y = g^y \bmod p$, надсилає ClientKeyExchange. Обидві сторони: PreMasterSecret $= g^{xy} \bmod p$	Так	Забезпечує PFS, обидві сторони вносять випадковість. Повільніше за RSA. Розміри ключів: 2048-4096 біт.
ECDHE	Сервер: обирає криву (secp256r1, x25519), генерує d_s, обчислює $Q_s = d_s \cdot G$, підписує, надсилає. Клієнт: генерує d_c, обчислює $Q_c = d_c \cdot G$, надсилає. PreMasterSecret = x-coordinate$(d_c \cdot Q_s)$	Так	Набагато швидше за DHE. 256-біт ECDHE ≈ 3072-біт RSA за безпекою. Обов'язковий у TLS 1.3. Криві: secp256r1 (P-256), secp384r1, x25519 (рекомендовано), x448

4.2 Методи автентифікації

Алгоритм	Процес	Розмір ключа	Характеристики
RSA Signatures	Підпис: $s = \text{Hash}(m)^d \bmod n$. Верифікація: $\text{Hash}' = s^e \bmod n$, порівняння Hash == Hash'	2048-4096 біт	Схеми: PKCS#1 v1.5, RSA-PSS. Алгоритми: sha256WithRSA, sha384WithRSA, sha512WithRSA. Verification швидше за signing. Застарілий: md5WithRSA, sha1WithRSA (з 2017)
ECDSA	Підпис: $e = \text{Hash}(m)$, обрати випадковий k, $(x, y) = k \cdot G$, $r = x \bmod n$, $s = k^{-1}(e+rd) \bmod n$. Підпис $= (r, s)$. Верифікація: обчислити точку, порівняти координату з r	256-384 біт	256-біт ECDSA $\approx$ 3072-біт RSA. Signing швидше за RSA, verification повільніше. Дуже важливим зауваженням є те, що повторне використання nonce k дозволяє обчислити приватний ключ (Sony PS3 hack, 2010). Алгоритми: ecdsa-with-SHA256/384/512. Тільки підпис
DSA	Федеральний стандарт США (FIPS 186). Схожий на ECDSA, але в $\mathbb{Z}_p^*$ замість EC	2048-3072 біт	Значною мірою замінено ECDSA. Вилучено з TLS 1.3. Тільки підпис

4.3 Симетричне шифрування

Алгоритм	Ключ/Блок	Mode	Статус	Примітки
DES	56 біт / 64 біт	CBC	Заборонено	Brute-force за години (Deep Crack, 1998). Sweet32 attack
3DES	168 біт / 64 біт	CBC	Застарілий	$C = E_{K3}(D_{K2}(E_{K1}(P)))$. Sweet32 (64-bit block). Повільний. Заборонено у TLS 1.3
RC4	128 біт / Stream	–	Заборонено	Систематичні упередження у виході, особливо перші байти. Заборонено у TLS 1.3

Алгоритм	Ключ/Блок	Mode	Статус	Примітки
AES-128-CBC	128 біт / 128 біт	CBC	Захищений	$C_i = E_K(P_i \oplus C_{i-1})$, $C_0 =$ IV. Вразливий до padding oracle attack (Lucky13). Видалено з TLS 1.3
AES-256-CBC	256 біт / 128 біт	CBC	Захищений	Аналогічно AES-128-CBC. Видалено з TLS 1.3
AES-128-GCM	128 біт / 128 біт	AEAD	Захищений	Counter Mode + GMAC. 128-біт auth tag. Паралелізуємий. Обов'язковий у TLS 1.3
AES-256-GCM	256 біт / 128 біт	AEAD	Захищений	Аналогічно AES-128-GCM. Обов'язковий у TLS 1.3
ChaCha20-Poly1305	256 біт / Stream	AEAD	Захищений	Stream cipher + MAC. Дуже швидкий у software (без AES-NI). Константний за часом. Обов'язковий у TLS 1.3. RFC 7539

4.4 Message Authentication Code

HMAC (RFC 2104):

$$\text{HMAC}_H(K, m) = H((K' \oplus \text{opad}) \parallel H((K' \oplus \text{ipad}) \parallel m))$$

де: ipad = 0x36 repeated, opad = 0x5C repeated, K' = padded key
TLS 1.2 MAC обчислення:

$$\begin{aligned}\text{MAC} = \text{HMAC-SHA256}(&\text{MAC\_write\_secret},\\ &\text{seq\_num} \parallel \text{type} \parallel \text{version} \parallel \text{length} \parallel \text{fragment})\end{aligned}$$

- seq_num: 64-bit counter (захист від replay)
- type: ContentType (1 byte)
- version: ProtocolVersion (2 bytes)
- length, fragment: дані

MAC алгоритми:
AEAD (у TLS 1.2/1.3):

- Немає окремого MAC
- Authentication tag генерується разом з шифруванням

MAC	Output	Безпека	Статус TLS 1.3
HMAC-MD5	128 біт	Broken	Заборонений
HMAC-SHA1	160 біт	Слабкий	Застарілий
HMAC-SHA256	256 біт	Захищений	Використовується в PRF
HMAC-SHA384	384 біт	Захищений	Використовується в PRF
GCM Auth Tag	128 біт	Захищений	У стандарті
Poly1305	128 біт	Захищений	У стандарті

- Additional Authenticated Data: seq_num || type || version || length
- GCM: GMAC
- Poly1305: one-time key from ChaCha20

4.5 Pseudorandom Function

TLS 1.2 PRF (RFC 5246):

$$\begin{aligned}
&\text{PRF}(secret, label, seed) = P_\text{hash}(secret, label + seed)\\
&P_\text{hash}(secret, seed) = \text{HMAC}(secret, A(1) + seed)\\
&\qquad\qquad + \text{HMAC}(secret, A(2) + seed) + \ldots\\
&\text{де } A(0) = seed, \quad A(i) = \text{HMAC}(secret, A(i-1))
\end{aligned}$$

Default hash: SHA-256 (для TLS 1.2)
TLS 1.3 HKDF (RFC 5869):

- Extract-then-Expand paradigm
- Більш гнучкий та захищений
- Окремі ключі для handshake та application traffic

5 Висновки

Протоколи SSL та TLS представляють складну еволюцію криптографічних механізмів безпечної комунікації. Від початкових недоліків SSL 2.0 до сучасного мінімалістичного TLS 1.3, кожна версія виправляла вразливості попередників та впроваджувала передові криптографічні практики.

Ключові досягнення:

- Перехід від слабких MAC конструкцій до HMAC та AEAD
- Обов'язковий forward secrecy через ефемерний ECDHE/DHE
- Скорочення handshake з 2-RTT до 1-RTT (та 0-RTT для resumption)
- Видалення застарілих алгоритмів (RC4, DES, 3DES, MD5, SHA1, CBC у TLS 1.3)
- Шифрування handshake для захисту приватності

Інфраструктура відкритих ключів з сертифікатами X.509, ієрархією CA, та механізмами відкликання (CRL/OCSP) забезпечує довіру та автентифікацію у глобальному масштабі. Акредитовані CA через WebTrust аудити та CA/Browser Forum Baseline Requirements підтримують високі стандарти безпеки.

Сучасні криптографічні методи — ECDHE для узгодження ключів, ECDSA/RSA для автентифікації, AES-GCM/ChaCha20-Poly1305 для шифрування, та HMAC для виведення ключів — формують надійну основу для захищеної комунікації у мережі Інтернет.