НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1

з дисципліни:

«ПРОЕКТУВАННЯ, РОЗРОБКА І РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ СИСТЕМ»

Дослідження реалізацій протоколу SSL

Виконала:

Студентка групи ФІ-22мн

Калитюк Дар'я

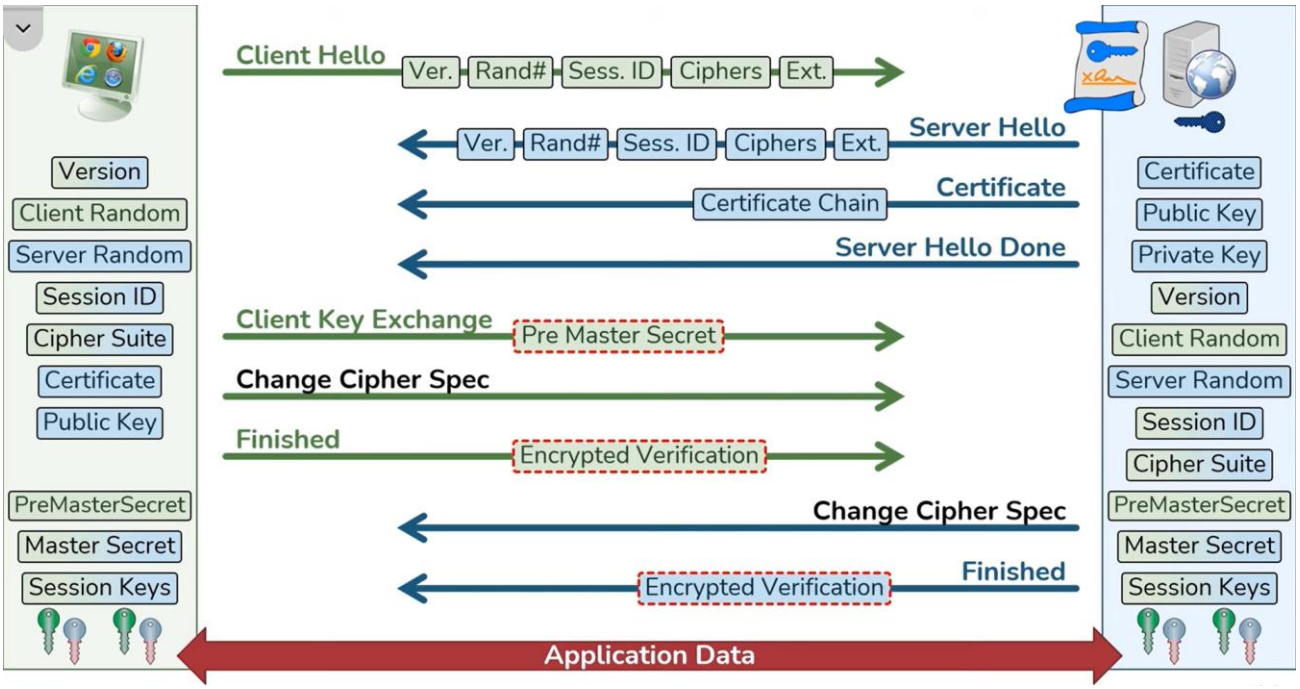
КИЇВ 2023

Мета роботи: дослідження особливостей реалізації криптографічних механізмів протоколу SSL/TLS.

Хід роботи

SSL, або шар захищених сокетів, була оригінальною назвою протоколу безпечного зв'язку, який розробила компанія Netscape у середині 90-х. SSL Коли 3тю версію протоколу випустили 1999 року, її стандартизувала спеціальна робоча група проєктування мережі Інтернет і дала їй нову назву: захист транспортного рівня, або TLS. Як говориться в TLS-документації, «різниця між цим протоколом та SSL 3.0 не є критичною». TLS і SSL формують серію протоколів, що постійно оновлюються, і їх часто об'єднують під назвою SSL/TLS. Мета протоколу - забезпечити захищене передавання даних. При цьому для автентифікації використовуються асиметричні алгоритми шифрування, а для збереження конфіденційності - симетричні. Процес, згідно з яким клієнт і сервер домовляються про ключі сесії, називається рукостисканням.

Формування сеансу:



Клієнт надсилає запит «Client Hello», який містить в собі 5 полів:

- **Version:** highest version of TLS/SSL Client Supports
- **Random Number - 32 bytes / 256 bits:** timestamp encoded in first four bytes
- **Session ID - 8 bytes / 32 bits:** 00000... All O's in initial Client Hello
- **Cipher Suites:** list of Cipher Suites Client supports
- **Extensions:** optional additional features added to TLS/SSL

Сервер надсилає запит «Server Hello», який містить в собі ті самі поля: версію протоколу, підтримувану Сервером, власне випадкове число. Також Сервер генерує ID сесії, обирає шифр із списку шифрів, отриманих від Клієнта і, за потреби, додає додаткові дані. Після цього Сервер

надсилає Клієнту сертифікат/и, і запит «Server Hello Done», що означає, що Сервер надіслав все, що треба.

Після отримання сертифікату у Клієнта виникає два питання:

1) Чи є сертифікат легітимним?
2) Чи дійсно цей сертифікат належить серверу?

Для валідації сертифікату достатньо перевірити підпис за допомогою відкритого ключа, що міститься в сертифікаті. Для відповіді на друге питання треба перевірити, чи дійсно Сервер має відповідний приватний ключ. Для цього Клієнт надсилає запит «Client Key Exchange», разом з яким надсилає:

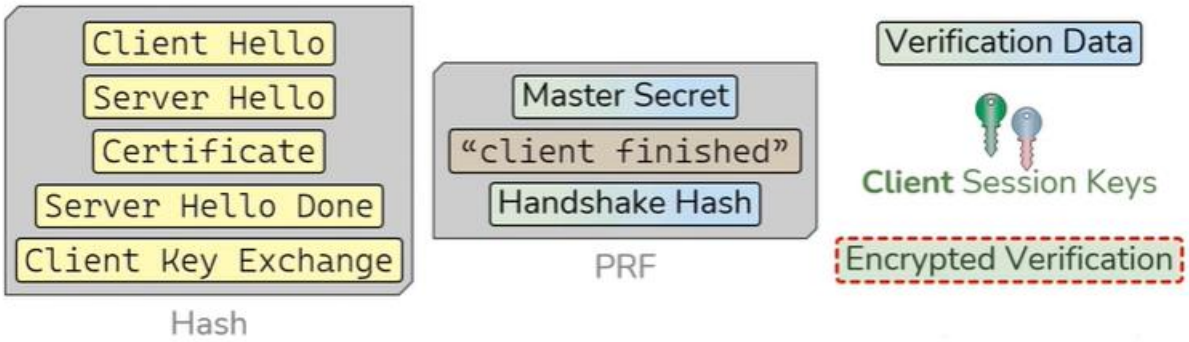
• **Pre-Master-Secret**: 2 bytes - TLS/SSL Version 46 bytes – Random

Після цього етапу обидві сторони – Клієнт і Сервер – мають всі необхідні ключи для початку обміну даними, але жодна зі сторін не знає, чи коректні ключі у іншої.

Після формування ключів Клієнт надсилає запит «Change Cipher Spec», який говорить про те, що сторона Клієнт готова перейти до користуванням обраним на початку шифром. Це повідомлення не є частиною TLS «рукостскання».

Щоб Сервер перевірив, що Клієнт має правильні ключі, Клієнт надсилає «Finished»:

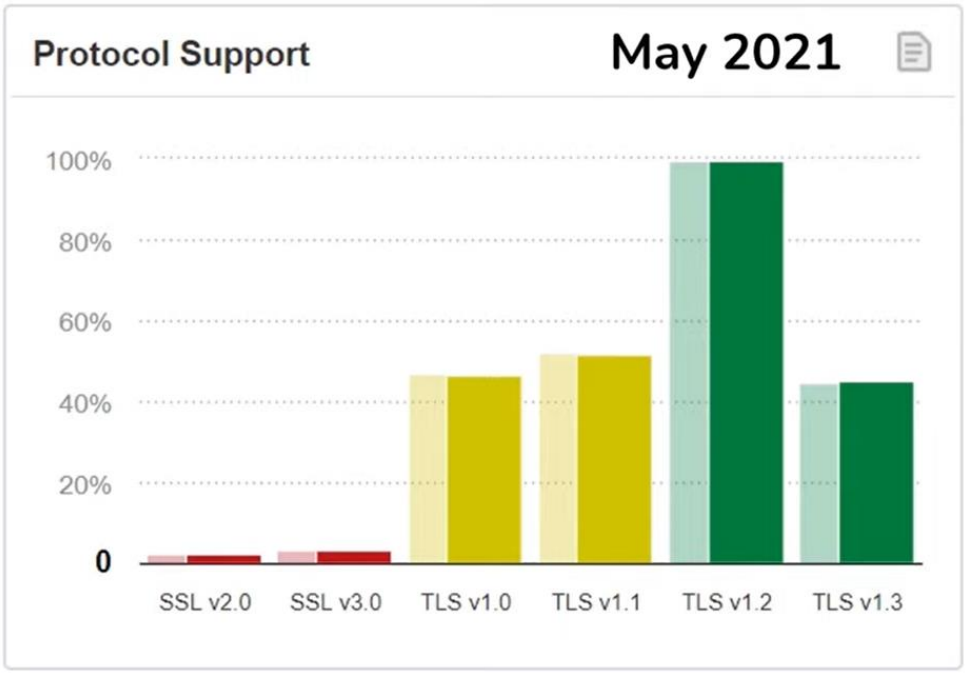
• **Verification Data**



Відновлення існуючого сеансу: якщо клієнт в запиті «Hello Client» надсилає ID встановленої раніше сесії, то встановлення зв'язку відбувається за спрощеною схемою: етап генерації симетричних ключів опускається. Однак тут є практичне обмеження: оскільки сервер має зберігати дані про всі відкриті сесії, це призводить до проблеми з популярними ресурсами, які

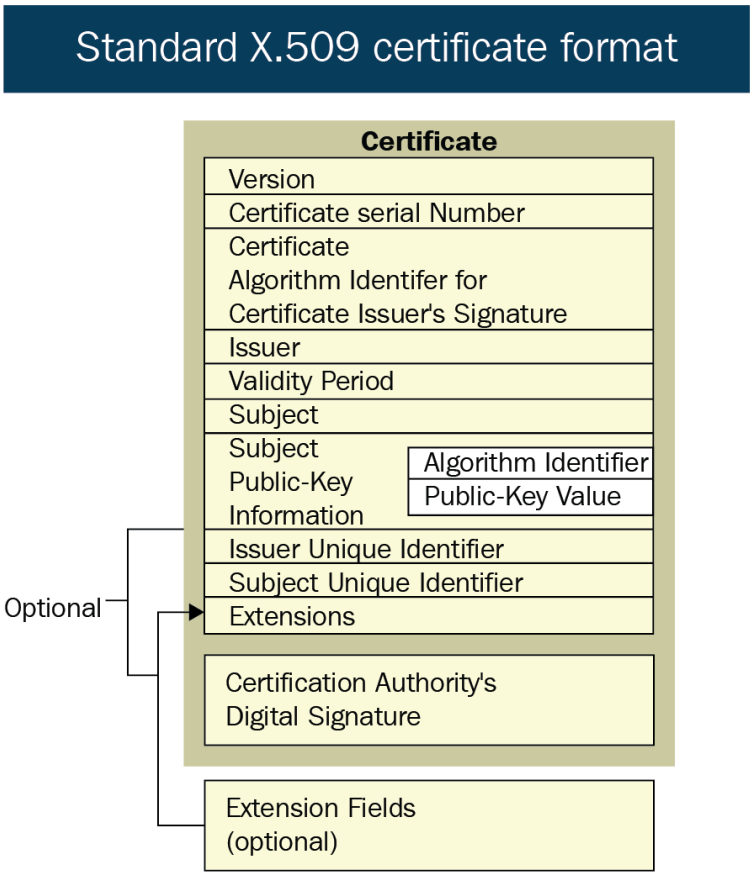
одночасно запитуються тисячами і мільйонами клієнтів. Для обходу цієї проблеми було розроблено механізм **«Session Ticket»**, який усуває необхідність зберігати дані кожного клієнта на сервері. Якщо клієнт під час початкового встановлення з'єднання вказав, що він підтримує цю технологію, то сервер під час TLS Handshake надсилає клієнту так званий **«Session Ticket»** - параметри сесії, зашифровані закритим ключем сервера. Під час наступного відновлення сесії клієнт разом із **«Client Hello»** відправляє наявний у нього **«Session Ticket»**. Таким чином, сервер позбавлений необхідності зберігати дані про кожне з'єднання, але з'єднання, як і раніше, безпечне, оскільки **«Session Ticket»** зашифрований ключем, відомим тільки на сервері.

Порівняльний аналіз версій протоколів SSL та TLS:

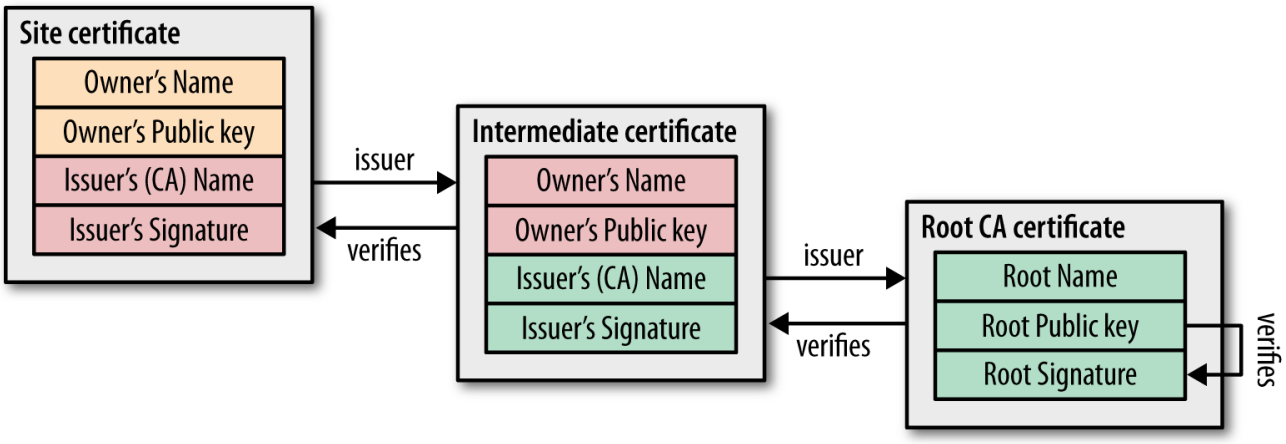


SSL 1.0	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
1994	**1995**	**1996**	**1999**	**2006**	**2008**	**2018**
- Netscape - Не було опубліковано - Багато вразливостей	- Netscape - Невдала спроба повністю оновити версію 1.0 - RSA Key Exchanges only	- Ланцюги сертифікатів - Стиснення даних - Опціональна підтримка додаткових Key Exchanges - POODLE	- HMAC - Обов’язкова підтримка додаткових Key Exchanges - (CBC атаки) BEAST -2021	- «Експортні» шифри - Захист від CBC атак -2021	- Безпеку генерації ключів - Опціональна підтримка AEAD шифрів - DES	- Коротке рукостискання (3 vs 5+) - Обов’язкова підтримка AEAD шифрів - forward secrecy - Вилучили нестійкі шифри - Вилучили переузгодження - Вилучили стиснення

Сертифікати: головною проблемою асиметричної криптографії є атаки типу «людина посередині», що вимагає в ключа наявності нового атрибуту – «довіри». Сертифікат ВК містить відомості про ключ і створює «довіру».



Центр сертифікації ключів – це організація, яка надає послуги електронного цифрового підпису. В залежності від топології Інфраструктури Відкритих Ключів він може бути один або декілька. Кореневий центр сертифікації ключів має самопідписаний – або кореневий – сертифікат.



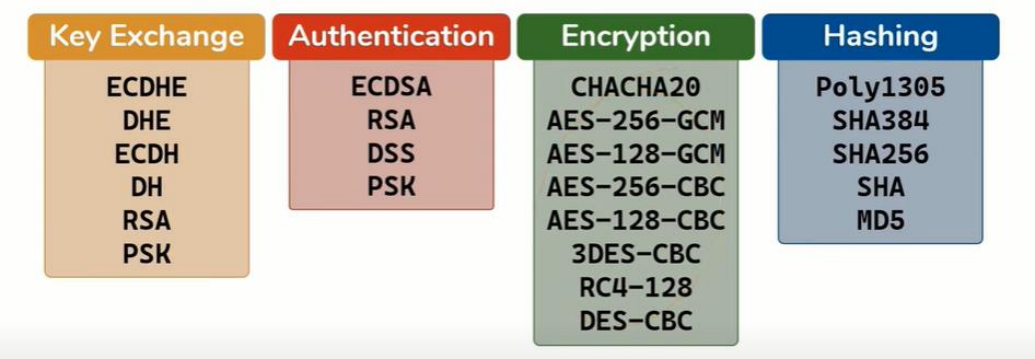
Звісно, виникають випадки, коли вже виданий сертифікат необхідно відкликати або анулювати. Отже, під час побудови ланцюжка довіри, необхідно перевіряти актуальність кожного довірчого вузла. Механізм цієї перевірки простий і в його основі лежить "Список відкликаних

сертифікатів" (CRL - "Certificate Revocation List"). У кожного з центрів сертифікації є цей список, що являє собою простий перелік серійних номерів відкликаних сертифікатів. Відповідно будь-хто, хто хоче перевірити справжність сертифіката, просто завантажує цей список і шукає в ньому номер сертифіката, що перевіряється. Тут очевидно присутня деяка технічна нераціональність: для перевірки лише одного сертифіката потрібно запитувати весь список відкликаних сертифікатів, що спричиняє уповільнення роботи. Для боротьби з цим було розроблено механізм під назвою "Протокол статусів сертифікатів" (OCSP - Online Certificate Status Protocol). Він дає змогу здійснювати перевірку статусу сертифіката динамічно: Клієнт робить запит до OCSP-серверу про валідність сертифікату і отримує відповідь: timestamp + змінна статусу (good, revoked, unknown). Природно, це знижує навантаження на пропускну здатність мережі, але водночас породжує кілька проблем:

- Центри сертифікації повинні справлятися з навантаженням у режимі реального часу;
- Центри сертифікації повинні гарантувати свою доступність у будь-який час;
- Через запити реального часу центри сертифікації отримують інформацію про те, які сайти відвідував кожен конкретний користувач.

Модифікація OCSP – OCSP stapling: Сервер періодично надсилає до OCSP-серверу запити про валідність сертифікату. Тепер кожному Клієнту Сервер може надсилати одну й ту саму відповідь під час TLS рукостискання – до моменту часу нового запиту.

Ciphers suites:



Список найбезпечніших наборів шифрів, IANA:

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_PSK_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

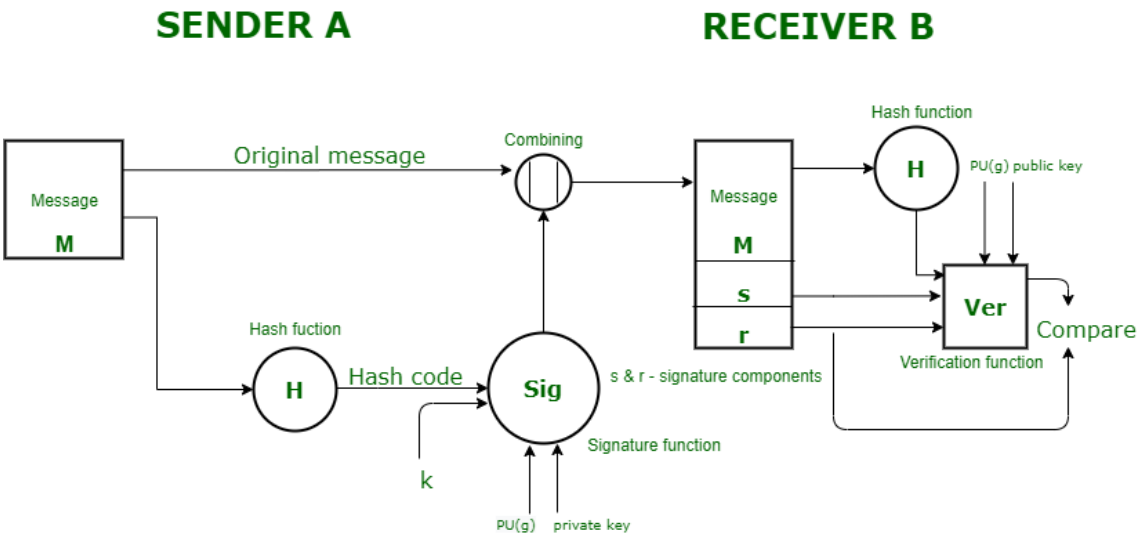
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

Internet Assigned Numbers Authority — американською організацією, що керує просторами IP-адрес, доменів верхнього рівня, а також реєструє типи даних MIME і параметри інших протоколів Інтернету.

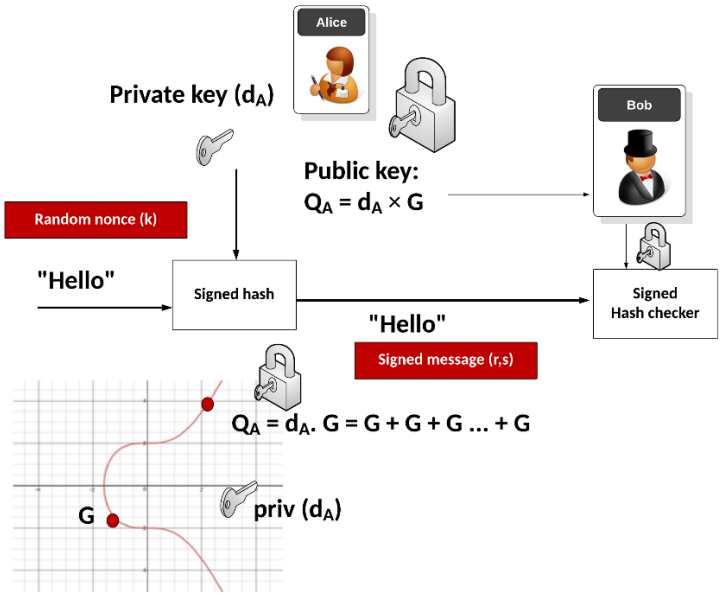
DH

Step	Alice	Bob
1	Parameters: p, g	
2	$A = \text{random}()$ $a = g^A \pmod p$	$\text{random}() = B$ $g^B \pmod p = b$
3	$a \longrightarrow$ $\longleftarrow b$	
4	$K = g^{BA} \pmod p = b^A \pmod p$	$a^B \pmod p = g^{AB} \pmod p = K$
5	$\longleftarrow E_K(data) \longrightarrow$	

DSS

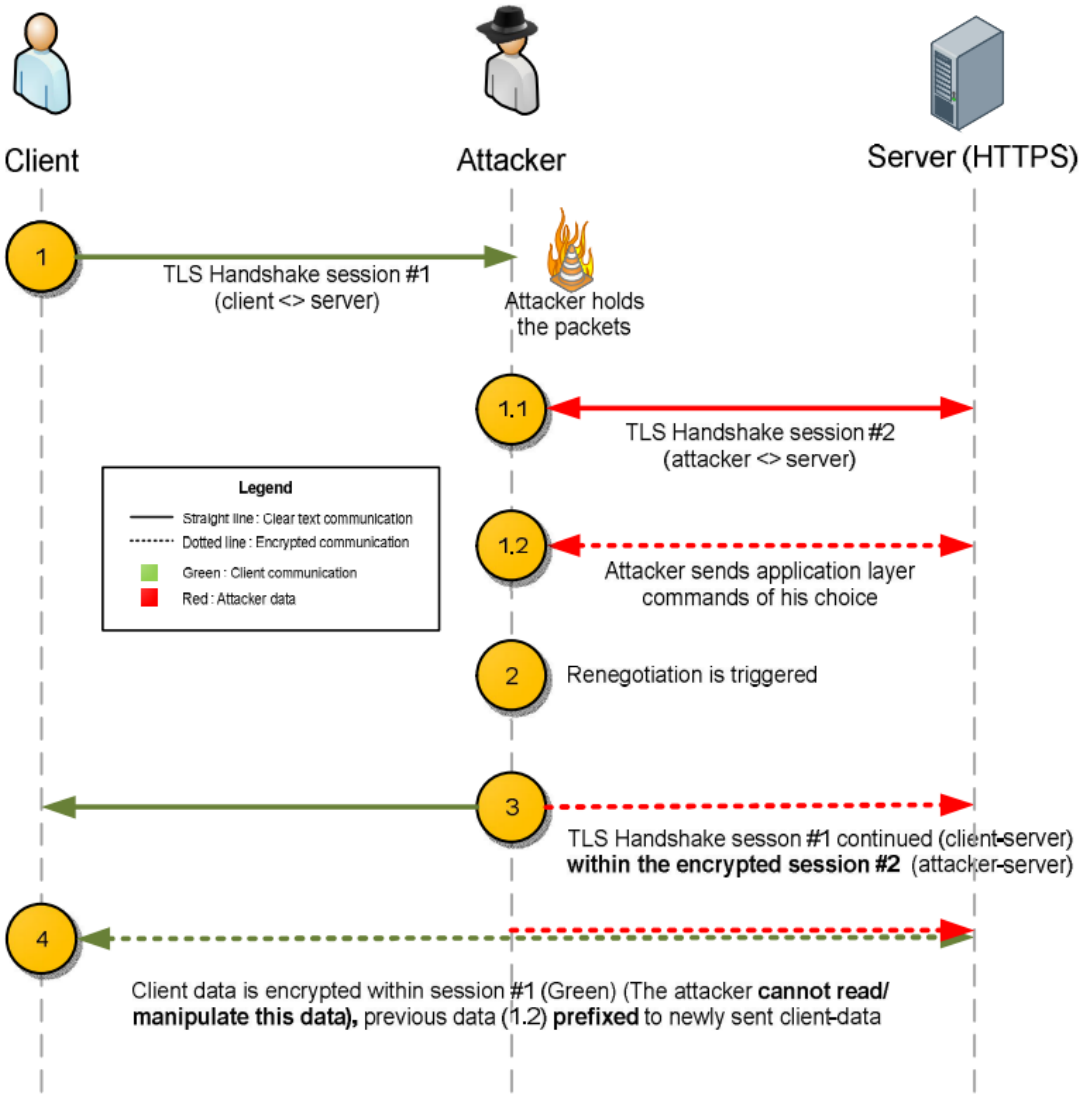


ECDSA



Вразливості SSL\TLS:

1. ***Атака переузгодження***

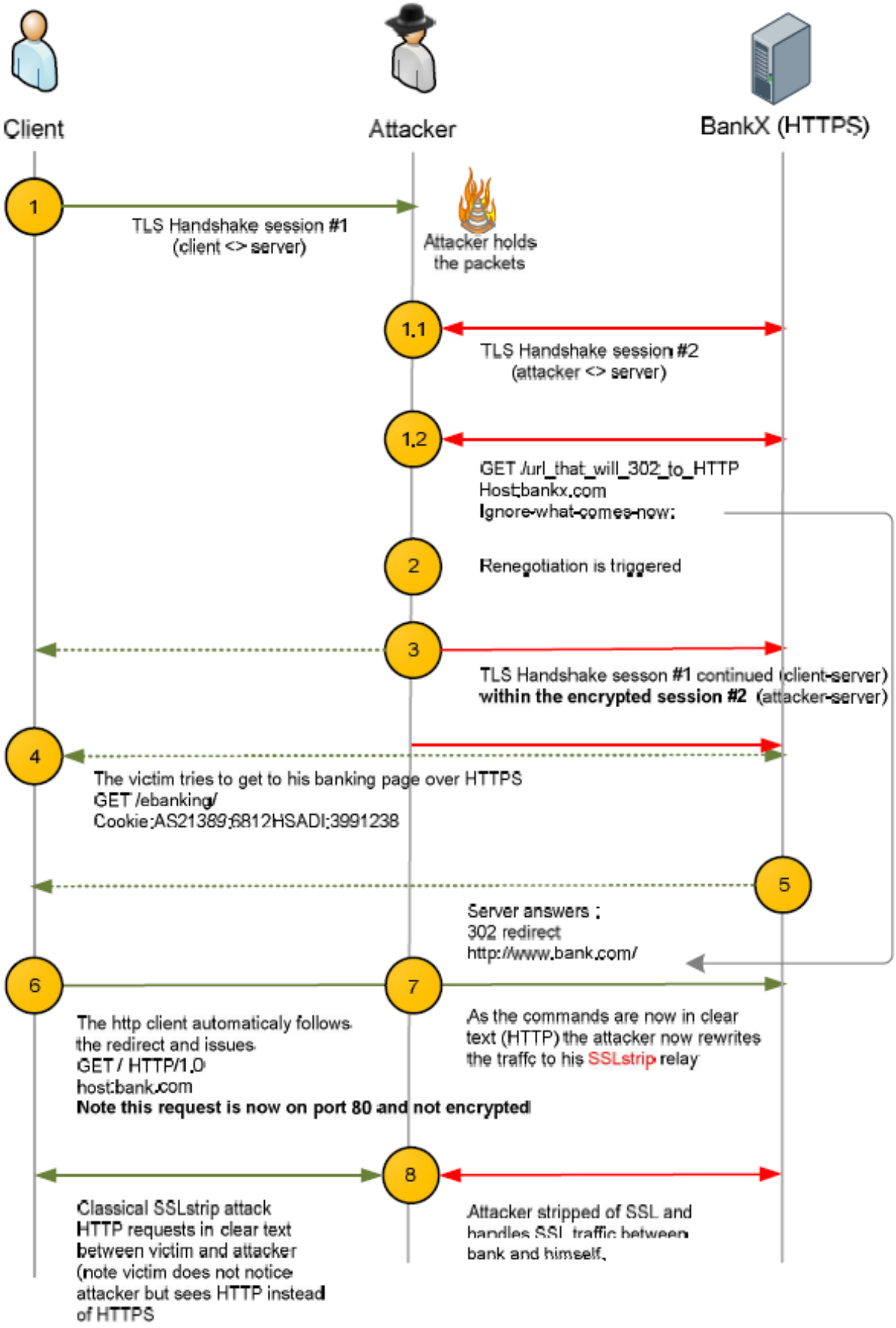


2. ***Атака відкату версії***

Під час атаки відкату або пониження версії зловмисник змушує систему переключитись на використання менш безпечного рівня роботи. Така атака є можливою завдяки властивостям зворотної сумісності системи. Атака відкату версії, як правило, є першим етапом в більш складній атаці.

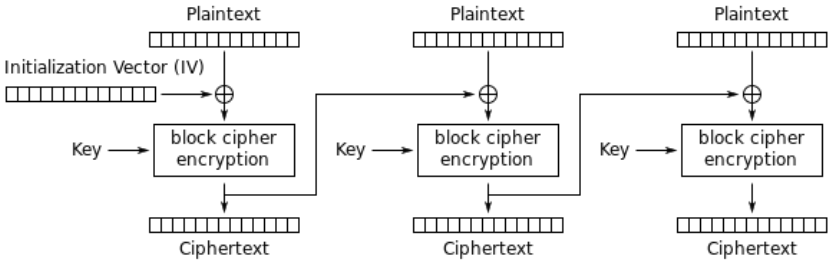
3. ***Атаки SSLstrip***

починається з атаки MITM і атаки переузгодження. Після зниження HTTPS до HTTP, зловмисник перехоплює всі незашифровані запити жерти, при цьому утримуючи HTTPS з'єднання із Сервером.


Client
Attacker
BankX (HTTPS)
1
TLS Handshake session #1
(client <> server)
Attacker holds
the packets
1.1
TLS Handshake session #2
(attacker <> server)
1.2
GET /url_that_will_302_to_HTTP
Host:bankx.com
Ignore-what-comes-now:
2
Renegotiation is triggered
3
TLS Handshake sesson #1 continued (client-server)
within the encrypted session #2 (attacker-server)
4
The victim tries to get to his banking page over HTTPS
GET /ebanking/
Cookie:AS21389:6812HSADI:3991238
5
Server answers :
302 redirect
http://www.bank.com/
6
7
The http client automaticaly follows
the redirect and issues
GET / HTTP/1.0
host:bank.com
Note this request is now on port 80 and not encrypted
As the commands are now in clear
text (HTTP) the attacker now rewrites
the traffc to his SSLstrip relay
8
Classical SSLstrip attack
HTTP requests in clear text
between victim and attacker
(note victim does not notice
attacker but sees HTTP instead
of HTTPS
Attacker stripped of SSL and
handles SSL traffic between
bank and himself.

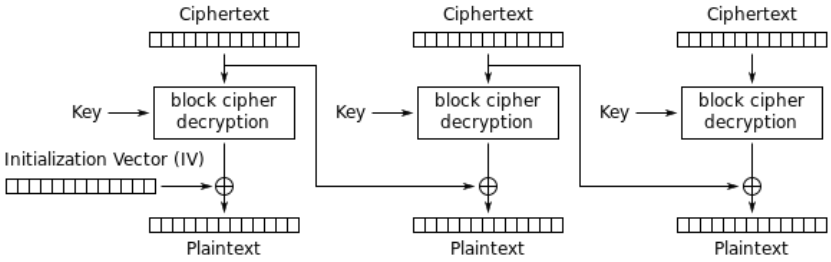
4. *Атака POODLE (Padding Oracle On Downgraded Legacy Encryption)*

використовує вразливості SSL 3.0, саме тому другим етапом (після атаки MITM) є атака відкату версії. POODLE використовує властивості симетричних блокових шифрів в режимі CBC і той факт, що в протоколі SSL 3.0 використовується підхід MAC-then-Encrypt, тобто цілісність падінгу не перевіряється.



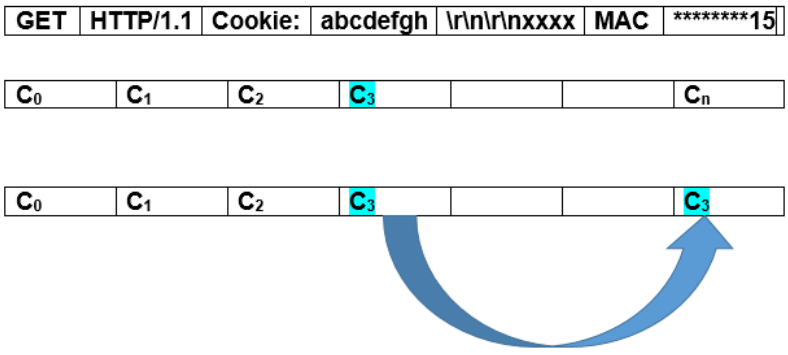
Cipher Block Chaining (CBC) mode encryption

$$C_n = E(P_n \oplus C_{n-1}) \Longrightarrow C_n = E(P_n) \oplus E(C_{n-1})$$



Cipher Block Chaining (CBC) mode decryption

$$P_n = D(C_n) \oplus C_{n-1}$$



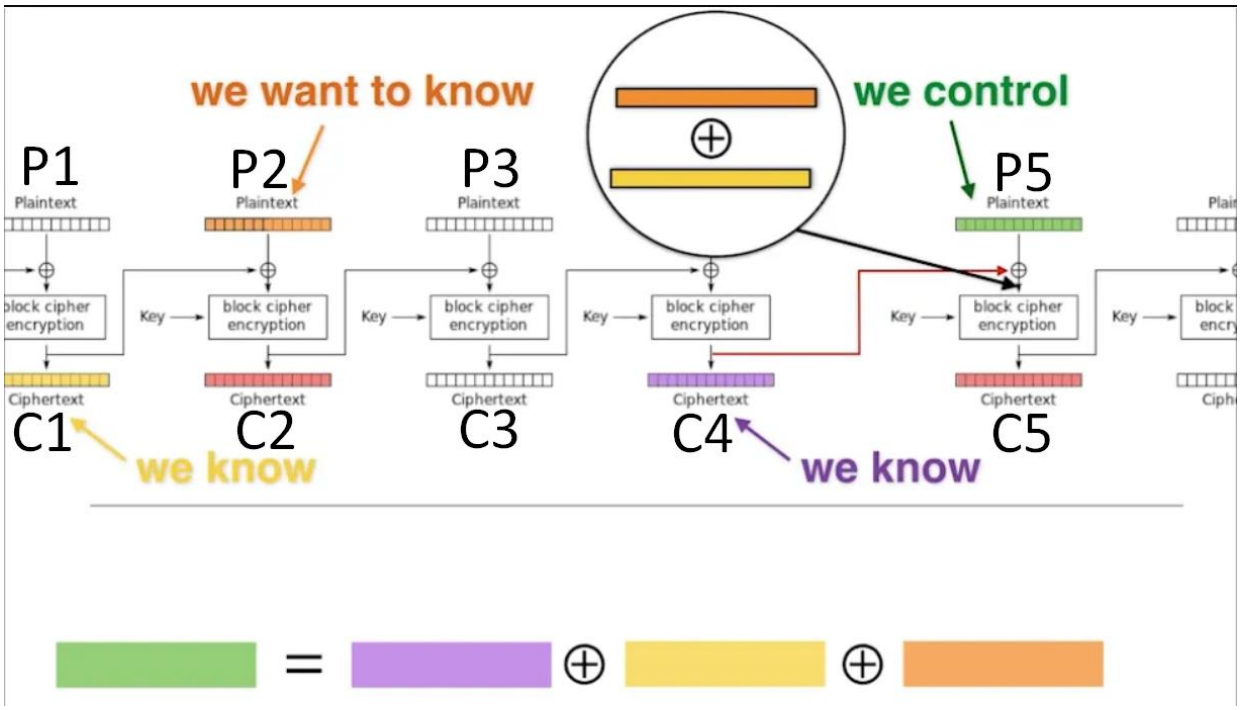
CBC Encryption of HTTP GET request which has Cookie

$$15 = D(C_3) \oplus C_{n-1}[15] = D\big(E(P_3) \oplus E(C_2)\big) \oplus C_{n-1}[15] = P_3 \oplus C_2 \oplus C_{n-1}[15]$$

$$\Longrightarrow P_3[15] = C_2 \oplus C_{n-1} \oplus 15$$

5. ***Атака BEAST (Browser Exploit Against SSL/TLS)***

використовує ті самі вразливості CBC режиму шифрування, що і POODLE. Тільки в цьому випадку для побайтового пошуку необхідної інформації зловмисник буде модифікувати не падінг, а міняти байт відкритого тексту таким чином, що відповідні блоки шифротексту співпали:



$$P_5 = D(C_5) \oplus C_4 = D(C_2) \oplus C_4 = P_2 \oplus C_1 \oplus C_4$$

6. ***Атака Lucky 13 (Обчислення TLS MAC включає 13 байт header information)***

це ще одна атака, що використовує властивості CBC режиму шифрування. В цій атаці, аналогічно до атаки POODLE, зловмисник замінює останній блок шифротексту, відкритий текст якого його цікавить, попереднім, додаючи до нього при цьому деяке значення $\triangle$, утворюючи таким чином співвідношення $P^* = P + \triangle$, де P^* -- модифікований ВТ, P^* -- шуканий ВТ. Відмінністю є те, що одна успішна атака дозволяє зловмиснику отримати одразу 2 байти шуканого ВТ. Головним показником успішності атаки в даному випадку є час відповіді сервера (що пов’язаний із залежністю швидкості обчислення MAC від розміру даних) на запит: в 2 випадках із трьох, не зважаючи на те, чи правильним виявився падінг, запит буде виконуватись довше. В третьому випадку, коли відповідь буде швидша, значення останній двох байтів буде 0x01, що дозволить зловмиснику з вищенаведеного рівняння отримати два байті P^*.

7. ***Атака CRIME (Compression Ratio Info-leak Made Easy)***

використовує стиснення для отримання зашифрованих даних, наприклад, cookies. Зловмисник спостерігає за розміром зашифрованого тексту, що надсилається Клієнтом, одночасно спонукаючи Клієнта здійснювати кілька продуманих веб-з'єднань з Сервером. Зловмисник спостерігає за зміною розміру стисненого корисного навантаження запиту, яке містить як

секретний файл cookie, що надсилається Клієнтом лише на Сервер, так і змінний вміст, створений зловмисником, по мірі зміни змінного вмісту. Коли розмір стисненого вмісту зменшується, можна зробити висновок, що існує ймовірність того, що деяка частина впровадженого вмісту збігається з частиною вихідного вмісту, який містить секретний вміст, який зловмисник бажає виявити.

8. ***Атака BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext)***

є прикладом атаки CRIME (яка по суті своїй є загальною атакою, застосовною до багатьох алгоритмів) проти HTTP стиснення (***gzip*** або ***DEFLATE***).

9. ***Атака FREAK (Factoring RSA Export Keys)***

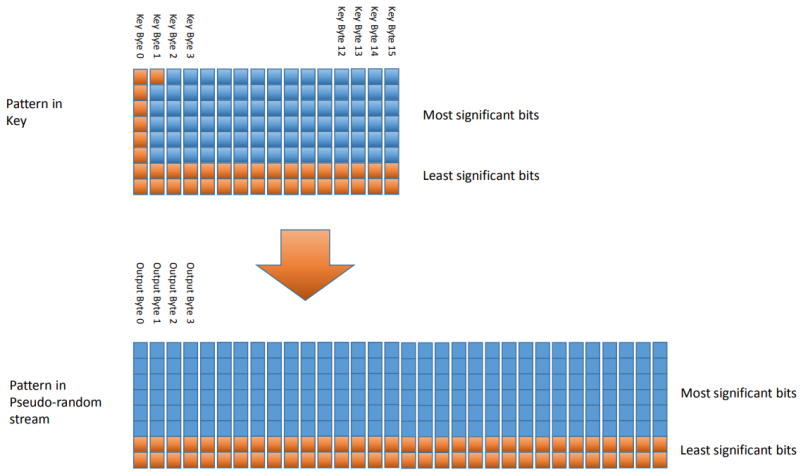
використовує підтримку більш старими версіями протоколу так званих Експорт-шифрів, які були заборонені в версії TLS 1.1. Використовуючи атаку переузгодження, зловмисник переконує сервер обрати набір шифрів, що містить «експортну » версію RSA замість звичайної, наприклад TLS_RSA_EXPORT_WITH_DES40_CBC_SHA, і отримує слабкий (512 біт) відкритий ключ, який він може факторизувати за декілька годин, і таким чином може дізнатись секретний ключ, а отже і отримати pre-master-secret, а з ним і всі інші сеансові ключі.

10. ***Атака Logjam***

діє аналогічно атаці FREAK, але ціллю в ній є алгоритм Діффі-Гелмана, де з невеликим відкритим ключем можна вирішити задачу дискретного логарифмування, наприклад, алгоритмом решета числового поля, і знайти секрет.

11. Атаки на шифр RC4

RC4 — потоковий шифр, розроблений Роном Рівестом, що виділявся швидкістю роботи і простотою реалізації, проте, як виявили згодом, мав багато вразливостей, через що наразі вважається ненадійним і не рекомендується до використання. Основною проблемою шифру RC4 є той факт, що для вироблення ключового потоку довгостроковий ключ просто конкатенується із nonce.



Це призводить до кореляції між ключем і ключовим потоком, а отже і до різноманітних атак на пов'язаних ключах. Також було показано, що ключовий потік містить статистичні зміщення, що дає змогу реалізувати атаку розрізнення.

Якщо говорити про атаку на SSL через атаку на RC4, то гарним прикладом є атака **Bar mitzvah**. Це атака на слабких ключах, яка дозволяє відновити молодші біти байтів відкритого тексту, і заснована на наявності L-подібного шаблону в деяких ключах RC4, які зберігають частину перестановки станів під час ініціалізації S-блоку, що визначають частину молодших бітів ключового потоку, що призводить до значного витоку байтів відкритого тексту з зашифрованого тексту. Щоб реалізувати атаку, зловмиснику потрібно визначити, в яких саме SSL-сесіях були використані слабкі ключі. Оскільки перші зашифровані байти включають в себе SSL-повідомлення «Finished» що має визначену структуру, при шифруванні із слабким ключовим потоком буде також генеруватись ШТ із видимим патерном.

12. Атаки на генератор псевдовипадкових чисел

PRF в протоколі SSL/TLS визначається функцією гешування, відповідно атака на PRF зводиться до атаки на геш-функцію, а саме на пошук колізій в ній. Ще у версії протоколу TLS 1.1 в якості PRF використовувалась комбінація SHA та MD5: обидва алгоритми виявились нестійкими до колізій. Починаючи з версії TLS 1.2 відбувся перехід на SHA-256\384, які вважаються стійкими.

13. Атака BERserk

дозволяла зловмисникам підроблювати підпис RSA, щоб видавати себе за легітимний Сервер. Атака використовує вразливість у передачі повідомлень у кодуванні ASN.1 під час перевірки підпису. Повідомлення ASN.1 складаються з різних частин, які кодуються за допомогою Basic Encoding Rules (BER) та/або Distinguished Encoding Rules (DER). Ця атака використовує той факт, що довжина поля в кодуванні BER може використовуватися для використання багатьох байтів даних. У вразливих реалізаціях ці байти потім пропускаються під час синтаксичного аналізу, що і дозволяє атаку.

14. Атака Heartbleed

Протокол SSL має розширення «heartbeat», яка дозволяє перевірку активності без переузгодження. В 2011му році виявилось, що його реалізація має дуже суттєву проблему – відсутність перевірки виходу запиту за допустимі межі: зловмисник може модифікувати «heartbeat» запит таким чином, щоб запитувати дані у пам'яті сервера, і отримувати до 64х кілобайт випадкових даних за один запит, що можуть містити в собі як секретні дані клієнтів, так і Сервера. За оцінками експертів половина всієї мережі, що підтримувала захищене з'єднання, опинилась під загрозою через цю помилку.