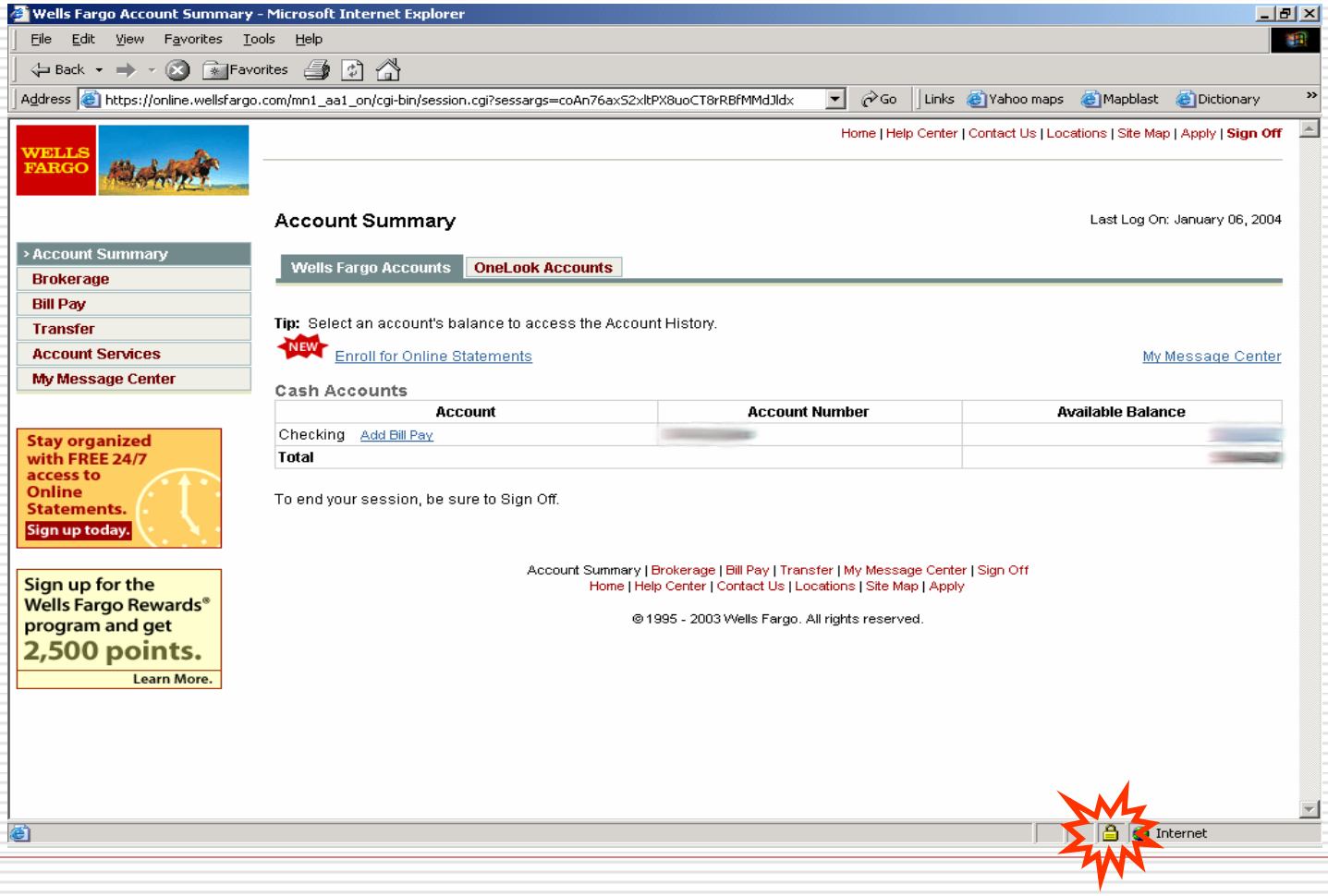
Современные криптографические протоколы

Протоколы SSL/TLS

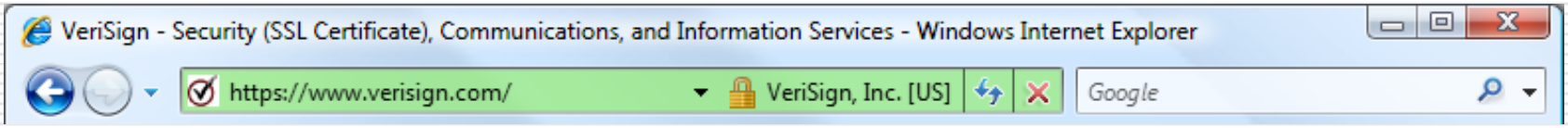
Что такое SSL / TLS?

- Протокол Transport Layer Security
- Основан на Secure Sockets Layers, версия 3.0
  - Похожий дизайн, разные алгоритмы
  - Стандарт де-факто для криптографической защиты веба
  - “Основная задача TLS обеспечить конфиденциальность и целостность коммуникаций между двумя приложениями”
    - … против «человека в середине»
  - Используется для защиты веб-серверов, VoIP, email, и т.д.
- Есть практически в любом браузере

SSL / TLS в реальном мире



Что означает этот «замочек»?



- Что веб-страница загружена с того узла, который указан в адресной строке
  - Но пользователь должен уметь интерпретировать ту информацию, которую он получает от браузера
- Содержимое страницы не простматривалось и не модифицировалось «человеком в середине»

Назначение SSL / TLS

- SSL/TLS обеспечить безопасность коммуникаций в условиях присутствия злоумышленника в сети
  - Вы сидите с ноутбуком в интернет-кафе, подключенном к контролируемому спецслужбами провайдеру во враждебной авторитарной стране (постарайтесь этого избегать ;-)
- Злоумышленник владеет сетью
  - WiFi, DNS, маршрутизаторы, его собственные веб-сайты
  - Может прочитать или изменить любой пакет, внедрить собственные пакеты в сессию

История SSL

- SSL 1.0
  - Внутренняя разработка Netscape, начало 1994?
  - Не использовался в реальности
- SSL 2.0
  - Опубликован Netscape в ноябре 1994
  - Имеет несколько уязвимостей
- SSL 3.0
  - Разработка Netscape и Paul Kocher, ноябрь 1996
- TLS 1.0
  - Общий стандарт, основанный на SSL 3.0, январь 1999
  - Несовместим с SSL 3.0
    - TLS использует HMAC вместо MAC

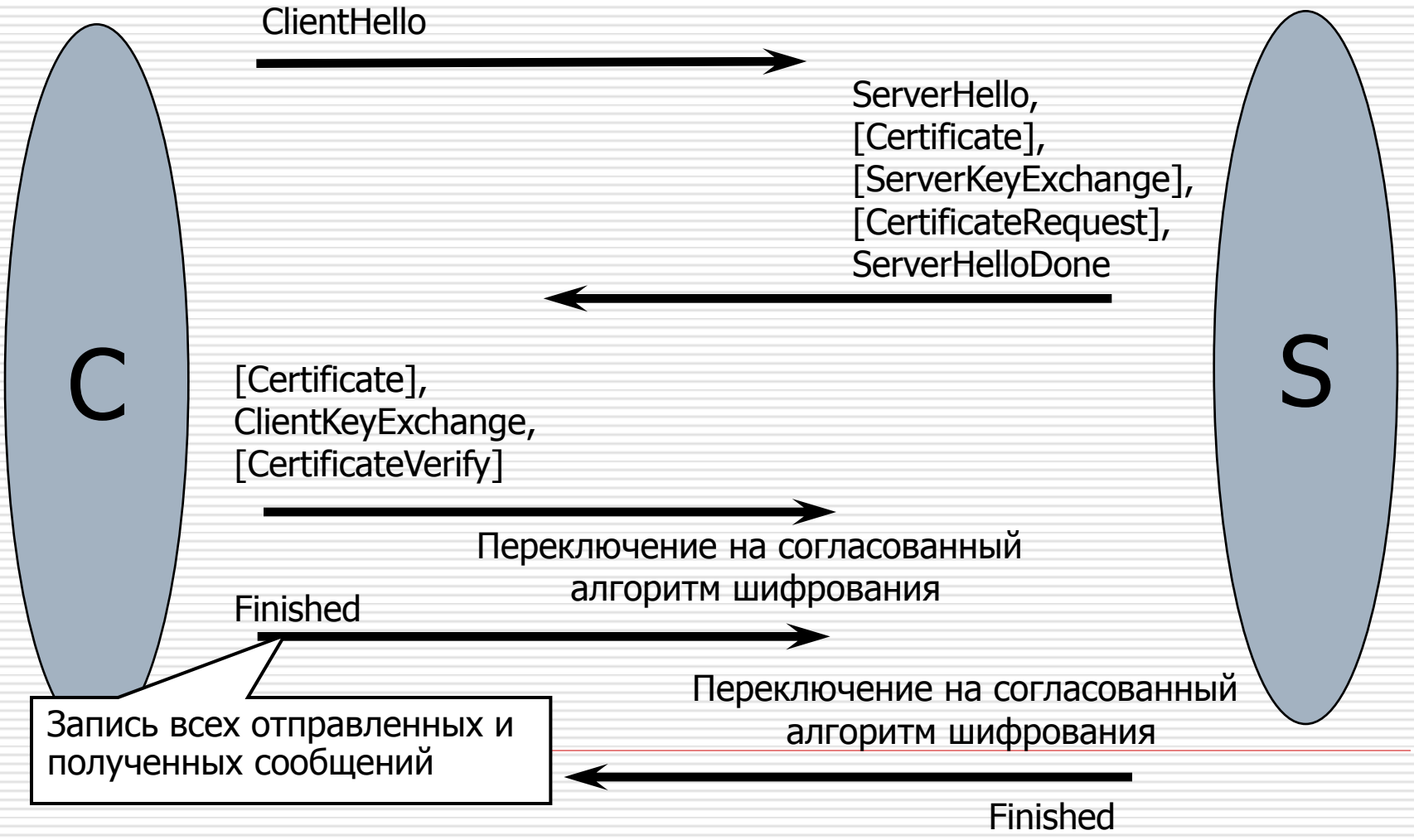
Основы SSL

- SSL состоит из двух протоколов
- Протокол рукопожатия
  - Использует ассиметрическую криптографию для установления общего секрета между клиентом и сервером
- Протокол обмена сообщениями
  - Использует общий секрет, выработанный в первой фазе, для шифрования трафика между клиентом и сервером

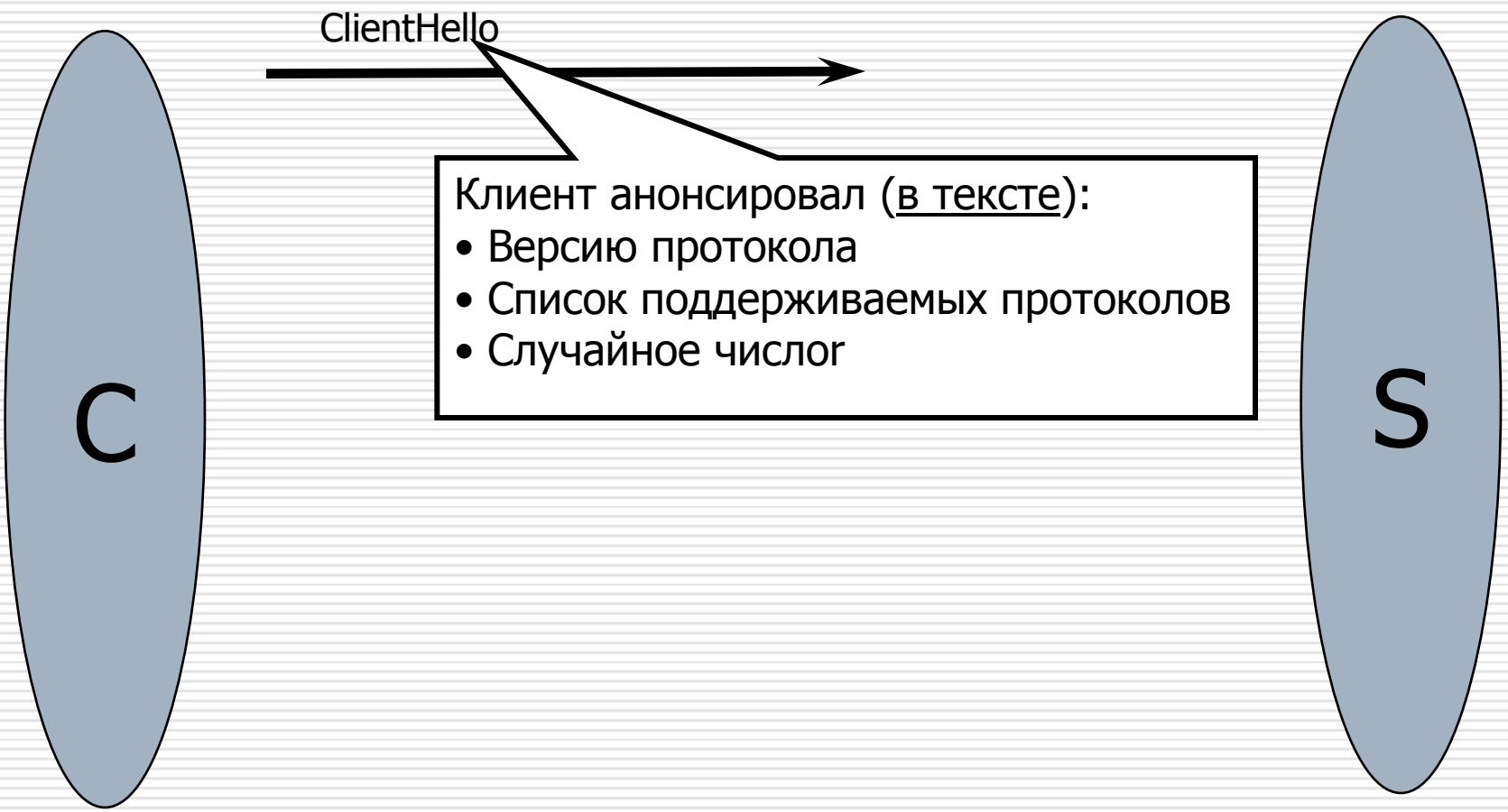
Протокол рукопожатия SSL

- Работает между клиентом (обычно, это веб-браузер) и сервером (возможно, это веб-сайт)
- Согласование версии протокола и перечня криптографических алгоритмов
- Аутентификация клиента и сервера (опционально)
  - Использует цифровые сертификаты чтобы узнавать публичные ключи друг друга и проверять подлинность
- Использует публичные ключи для выработки общего секрета

Структура протокола рукопожатия



ClientHello



ClientHello (RFC)

```
struct {
   ProtocolVersion client_version;
   Random random;
   SessionID session_id;
   CipherSuite cipher_suites;
   CompressionMethod
    compression_methods;
} ClientHello
```

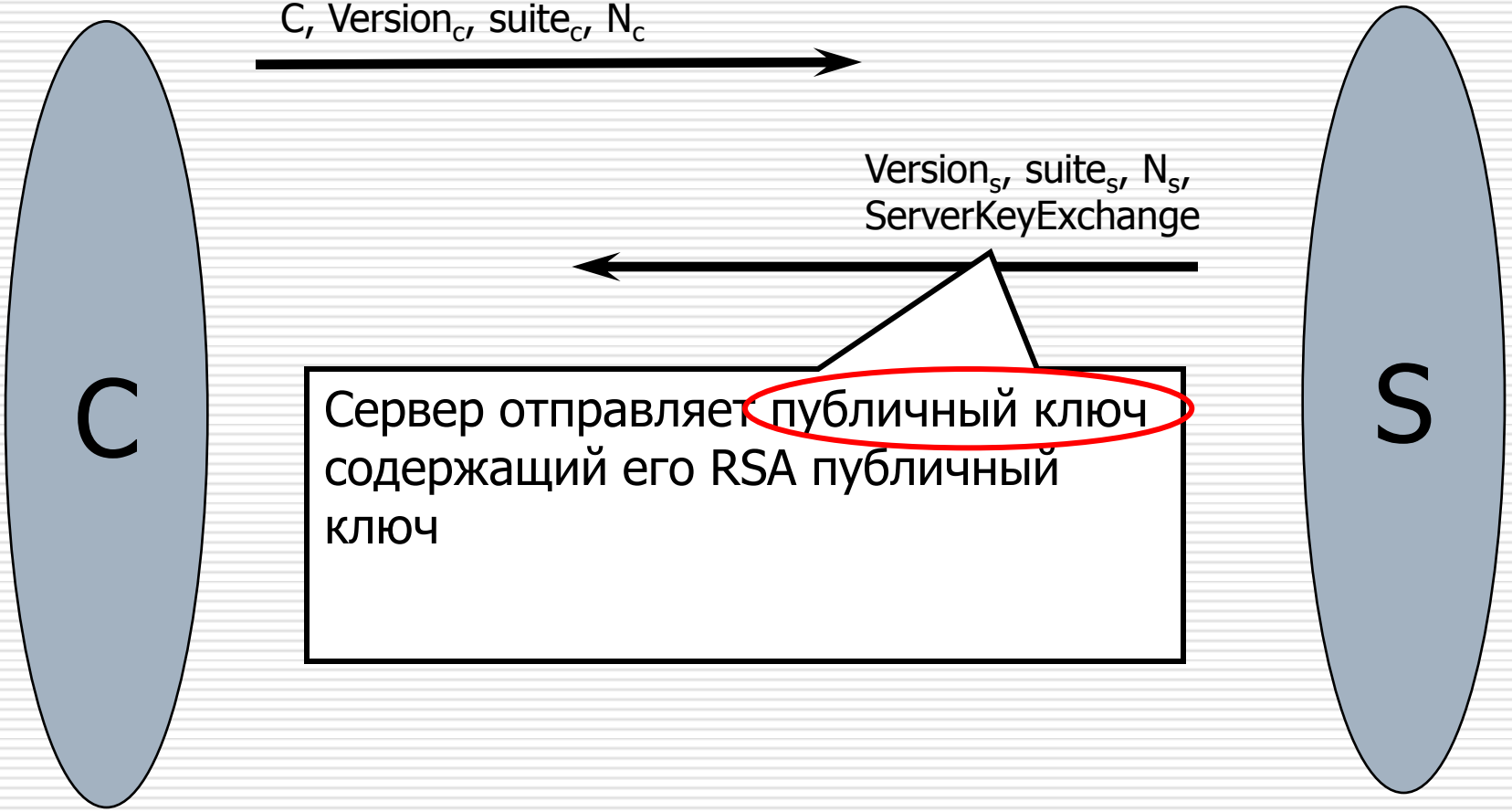
Максимальная версия протокола на клиенте

Session id (если клиент хочет продолжить старую сессию)

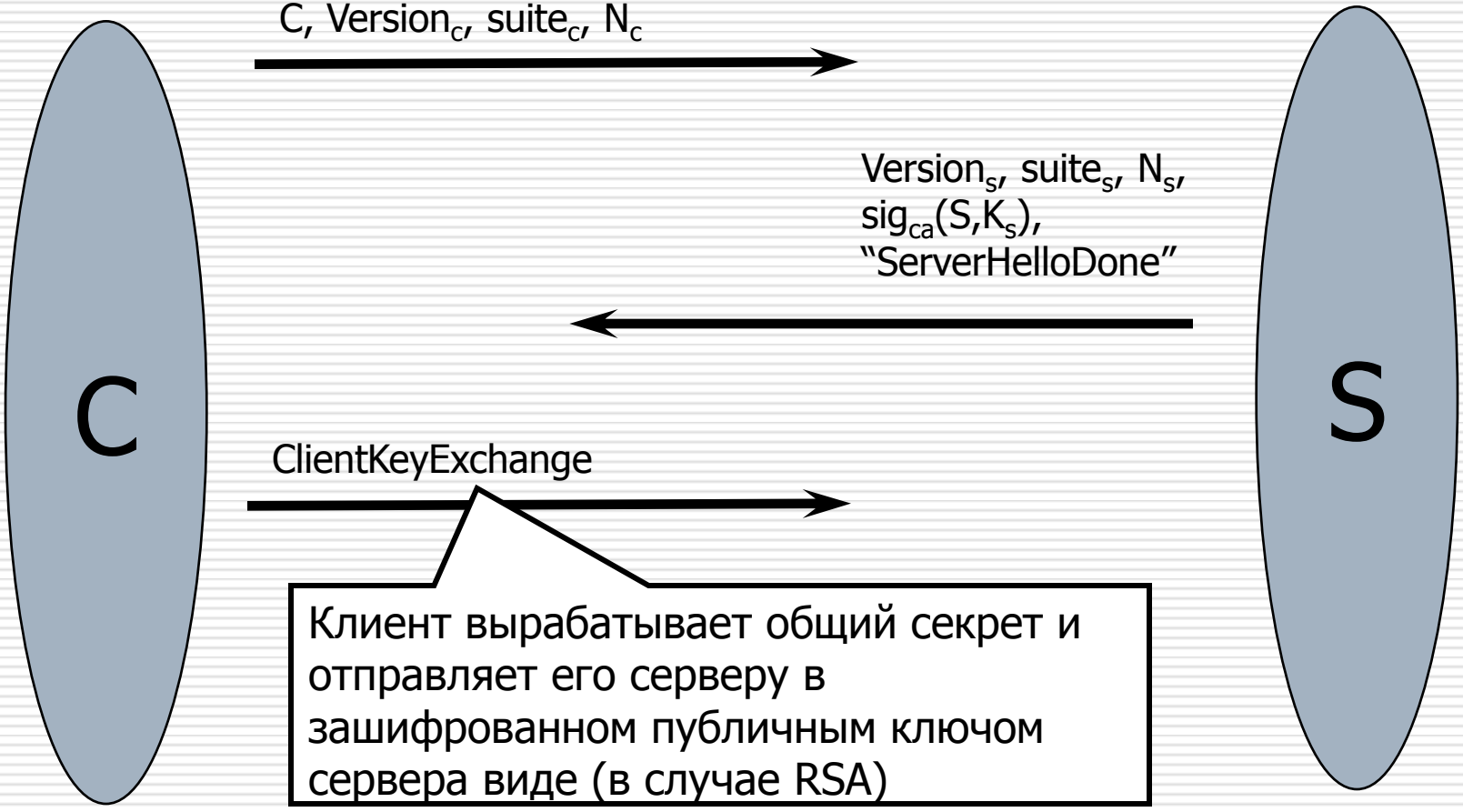
Список криптоалгоритмов на клиенте (RSA, Diffie-Hellman, и т.д.)

ServerHello

ServerKeyExchange



ClientKeyExchange



ClientKeyExchange (RFC)

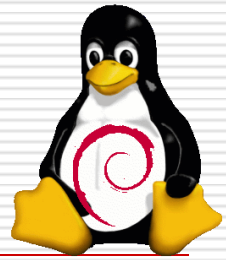
```
struct {
   select (KeyExchangeAlgorithm) {
      case rsa: EncryptedPreMasterSecret;
      case diffie_hellman: ClientDiffieHellmanPublic;
   } exchange_keys
} ClientKeyExchange

struct {
   ProtocolVersion client_version;
   opaque random[46];
} PreMasterSecret
```

Откуда берутся случайные биты?

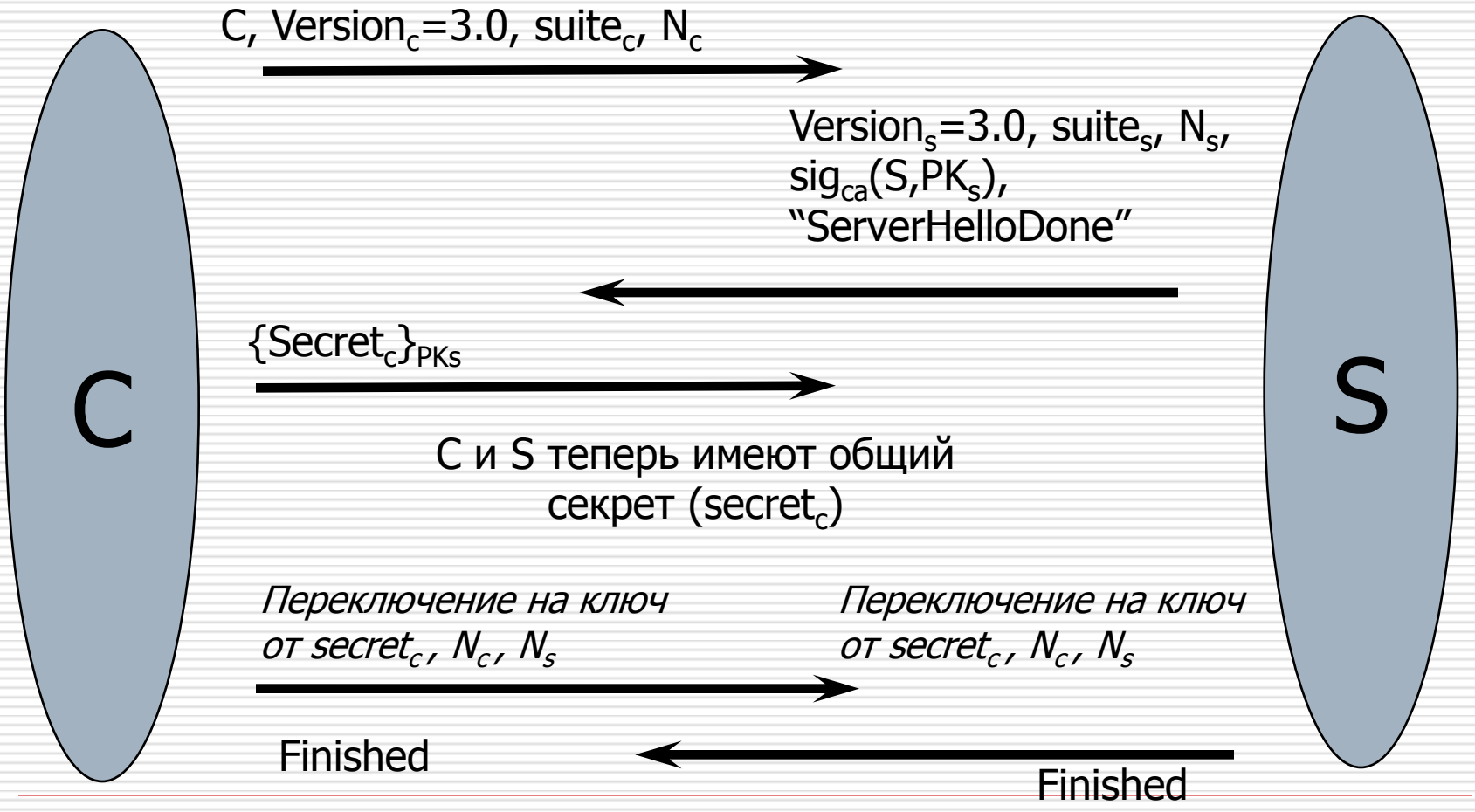
Случайные биты, из которых будут сгенерены симметричные ключи

Debian PRNG (2006-08)

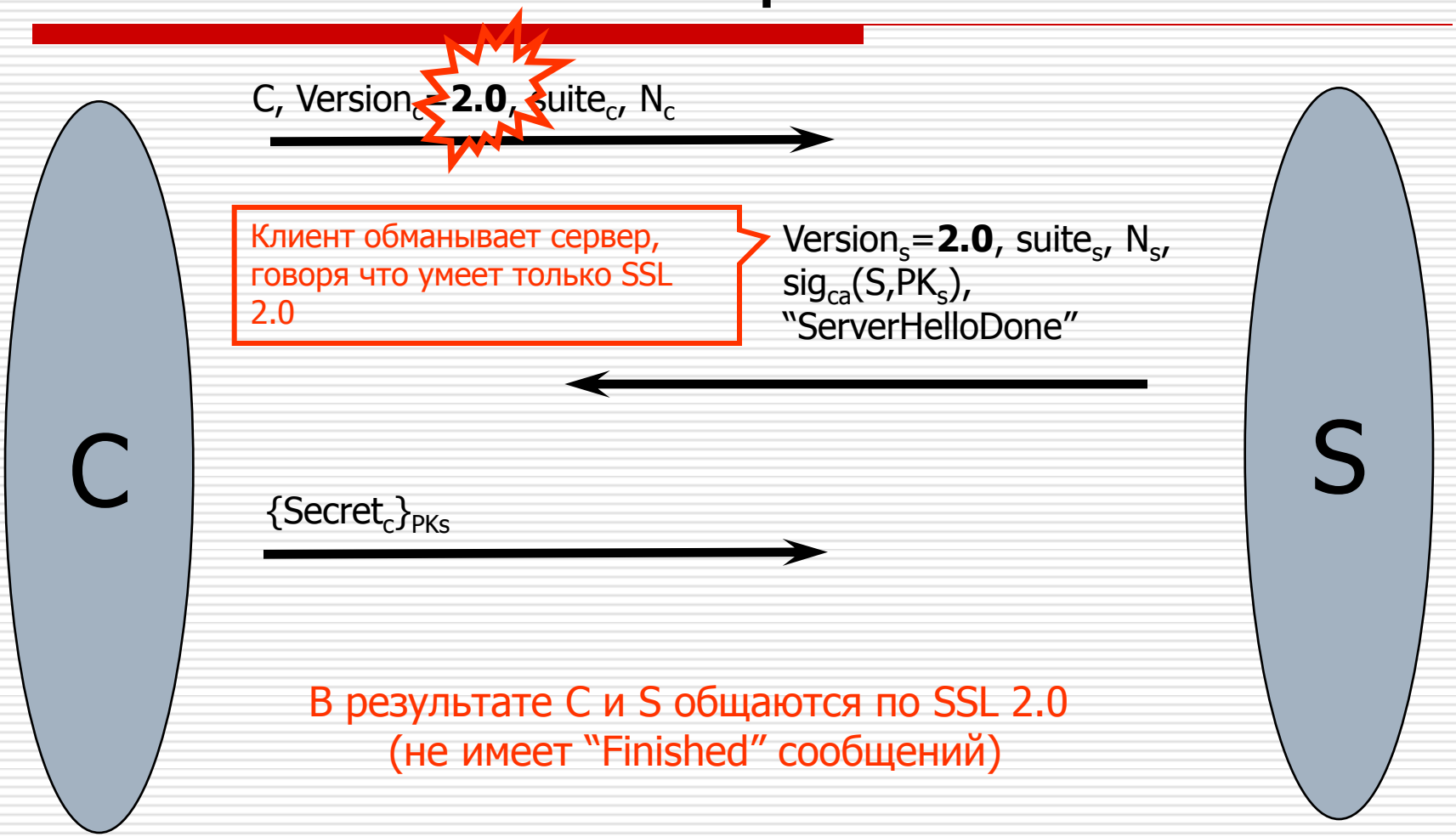


- Строчка кода в md_rand в Debian Linux
  - MD_Update(&m,buf,j); /* purify complains */
- Без этой строчки сид для PRNG зависит только от ID процесса
  - Максимум на Linux по-умолчанию = 32768
- Результат: все ключи, созданные с помощью Debian OpenSSL в 2006-08 сломаны
  - “Включая ключи SSH, OpenVPN, DNSSEC, ключевой материал сертификатов X.509 в SSL/TLS”

Рукопожатие в SSL 3.0



Атака отката версии

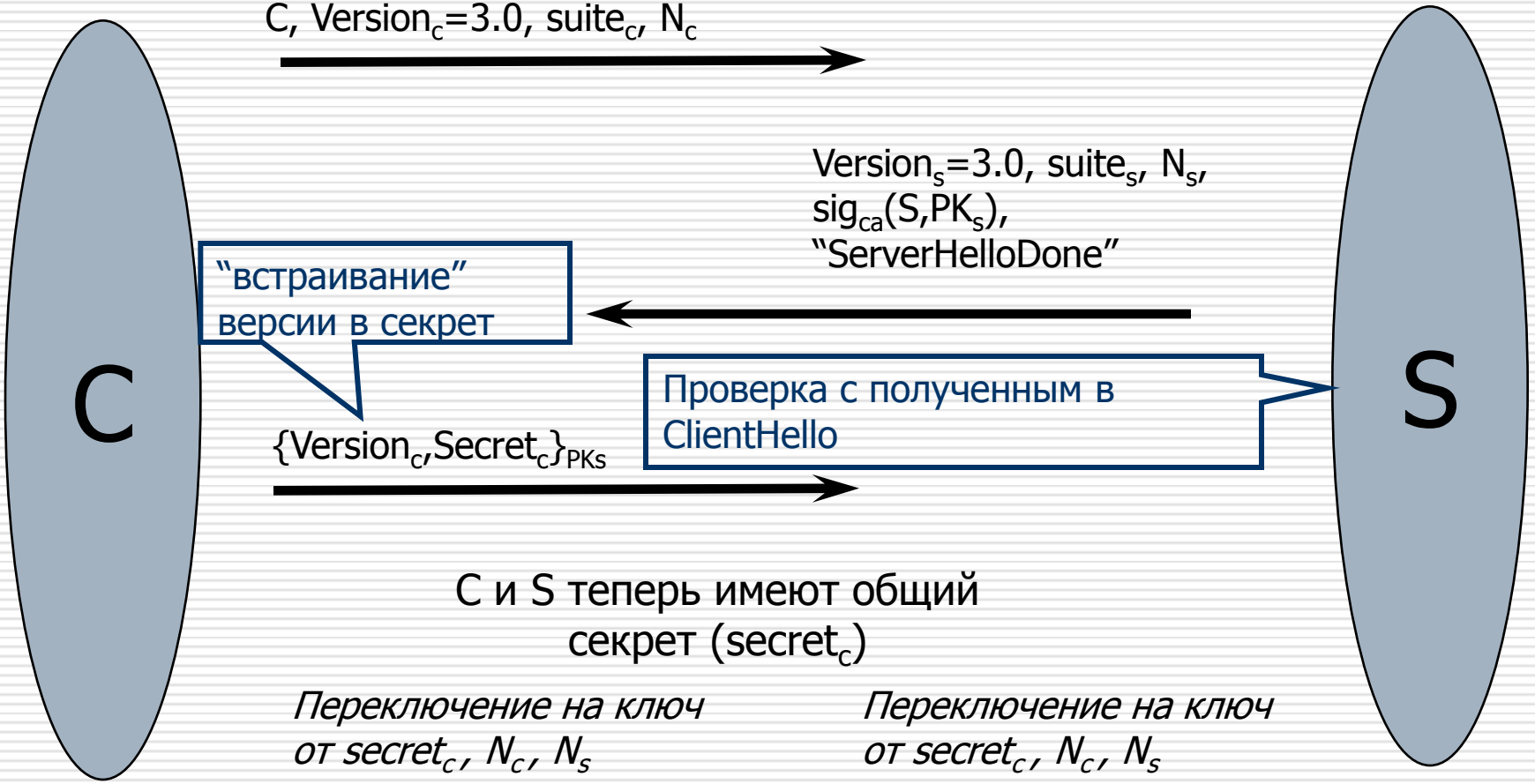


В результате C и S общаются по SSL 2.0
(не имеет "Finished" сообщений)

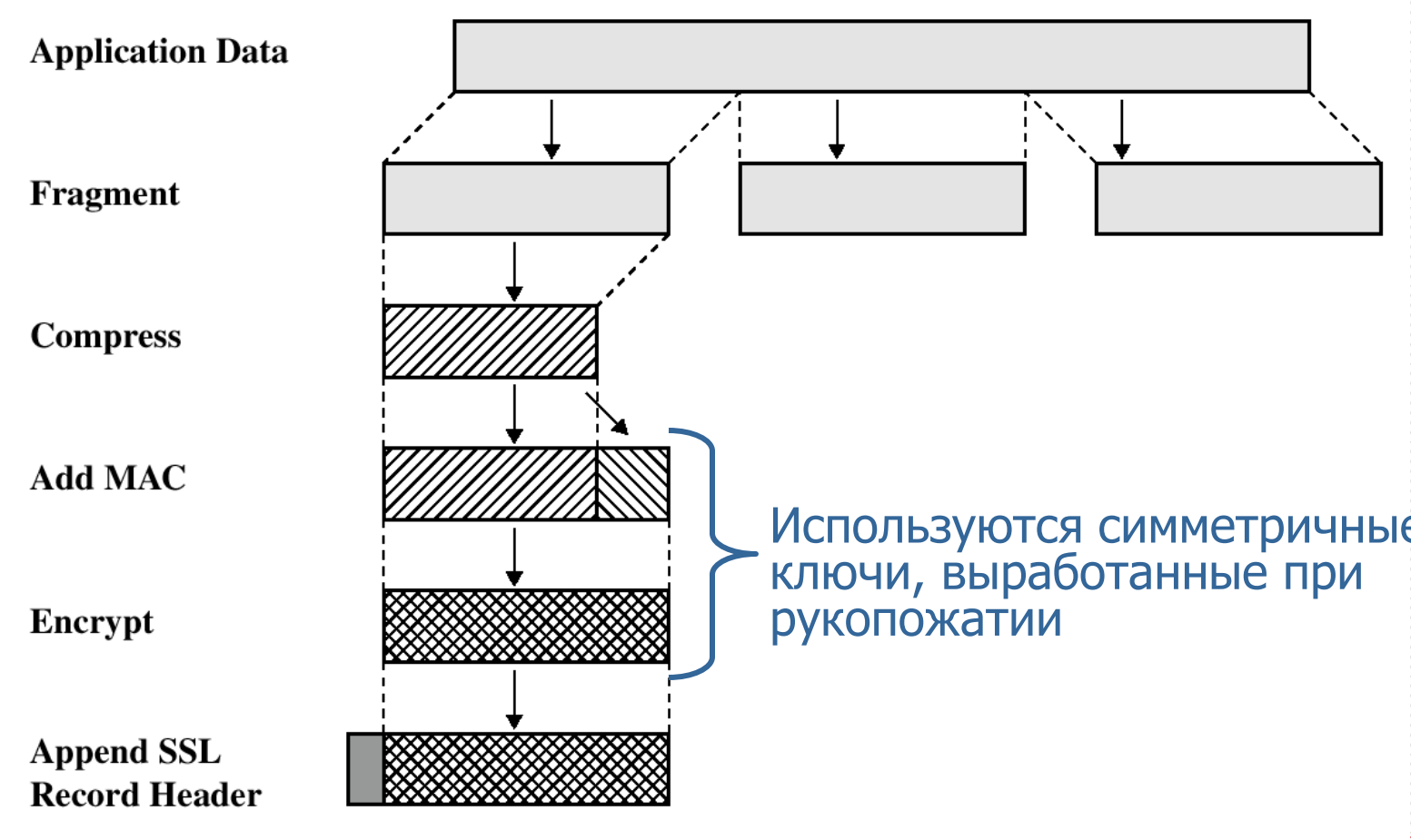
Уязвимости SSL 2.0

- Предпочтения криптоалгоритмов не аутентифицируются
  - Возможна атака “Cipher suite rollback”
- Слабый MAC construction
- SSL 2.0 использует паддинг при вычислении MAC, но поле длины паддинга не аутентифицировано
  - Атакующий может удалять байты в конце сообщения
- MAC использует только 40 бит
- Нет поддержки не-RSA алгоритмов

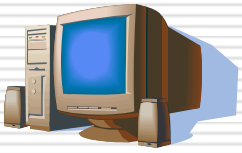
Проверка версии в SSL 3.0



Защита сообщений в SSL/TLS



HTTP/HTTPS



•Клиент

•HTTP на порт 80

•Перенаправление на HTTPS

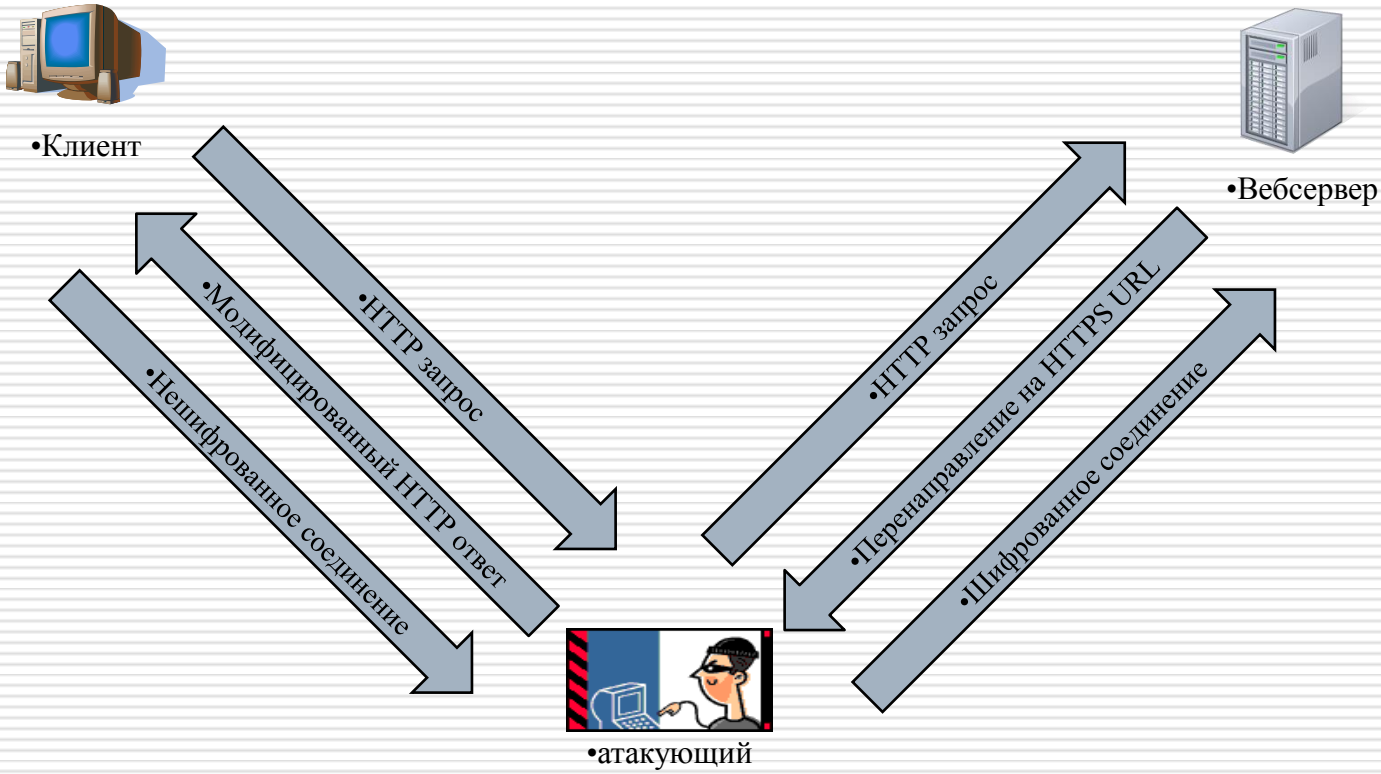
•HTTPS на порт 443

•Сертификат сервера

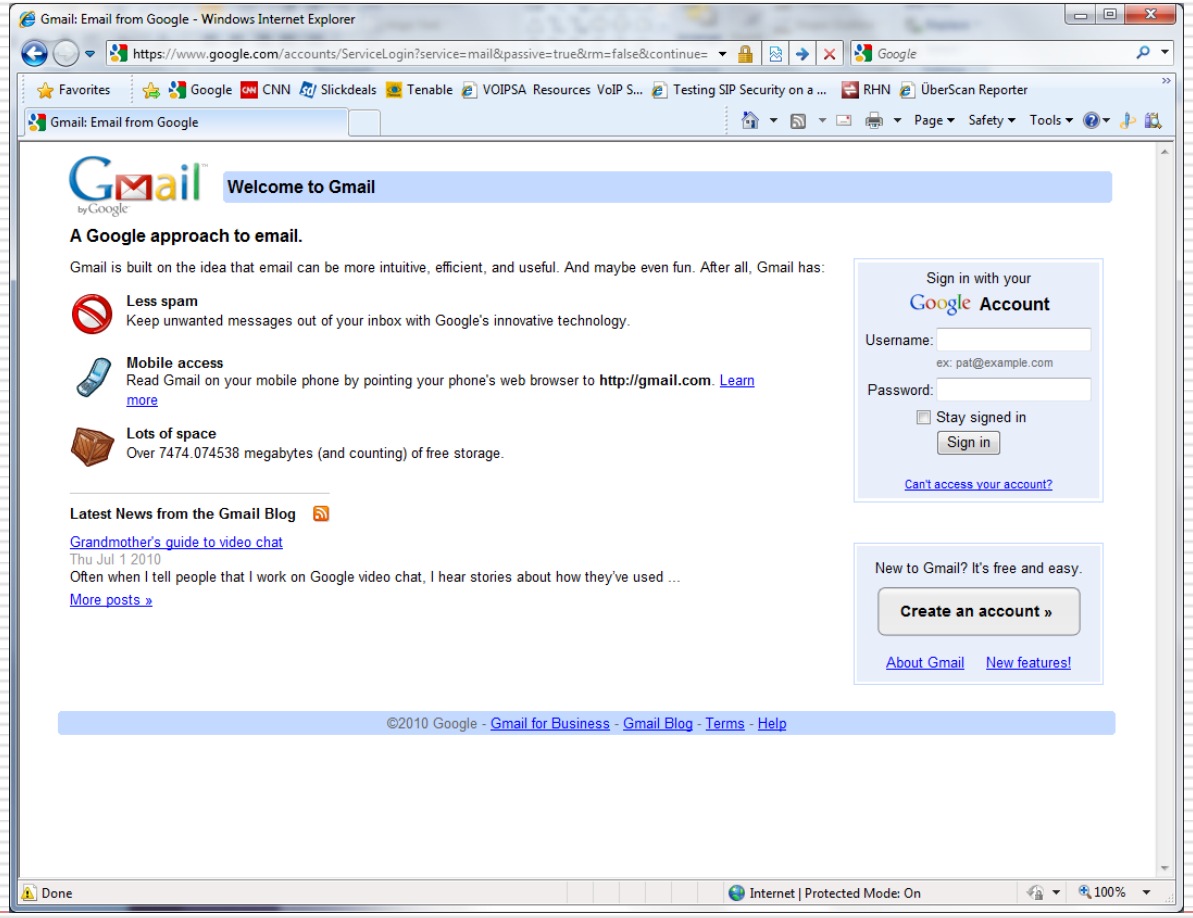
•Соединение установлено

•Сервер

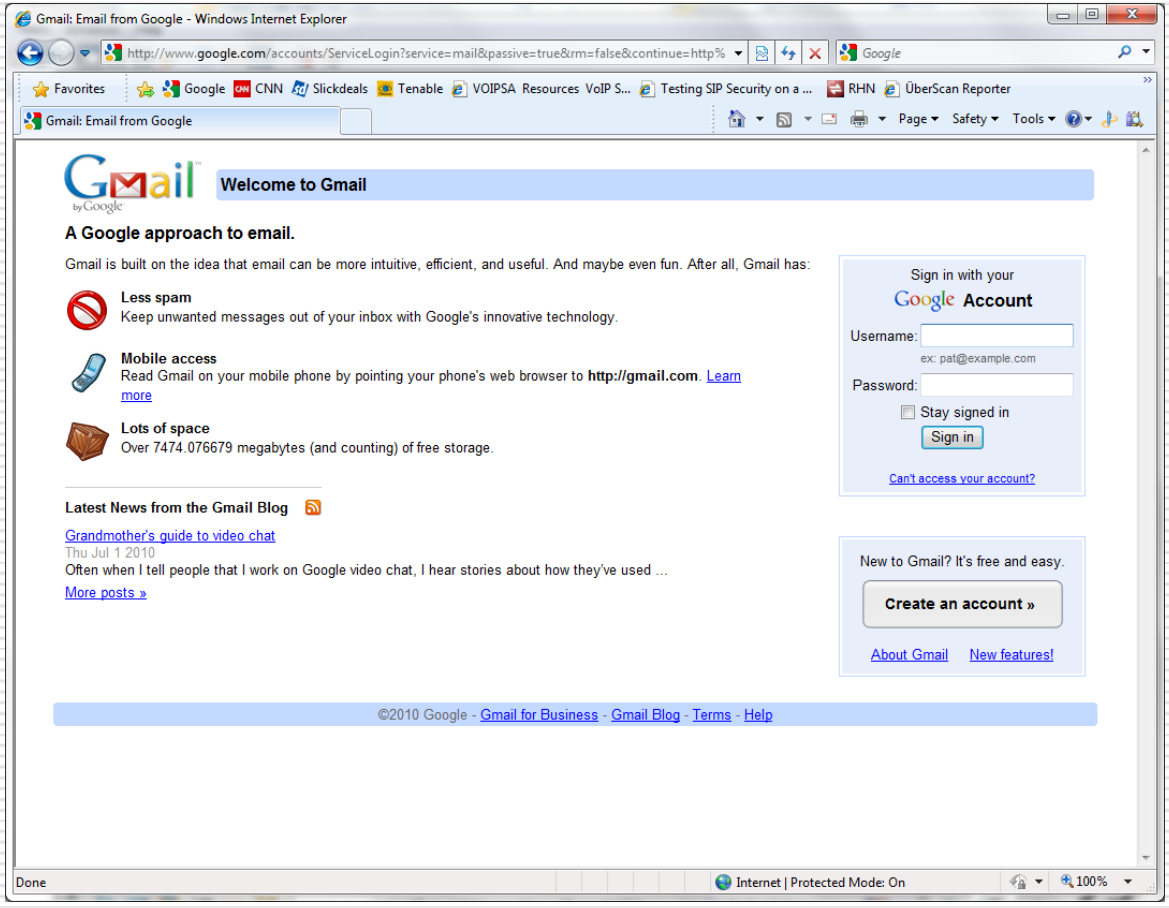
Hijacking Communication



Гмейл здорового человека

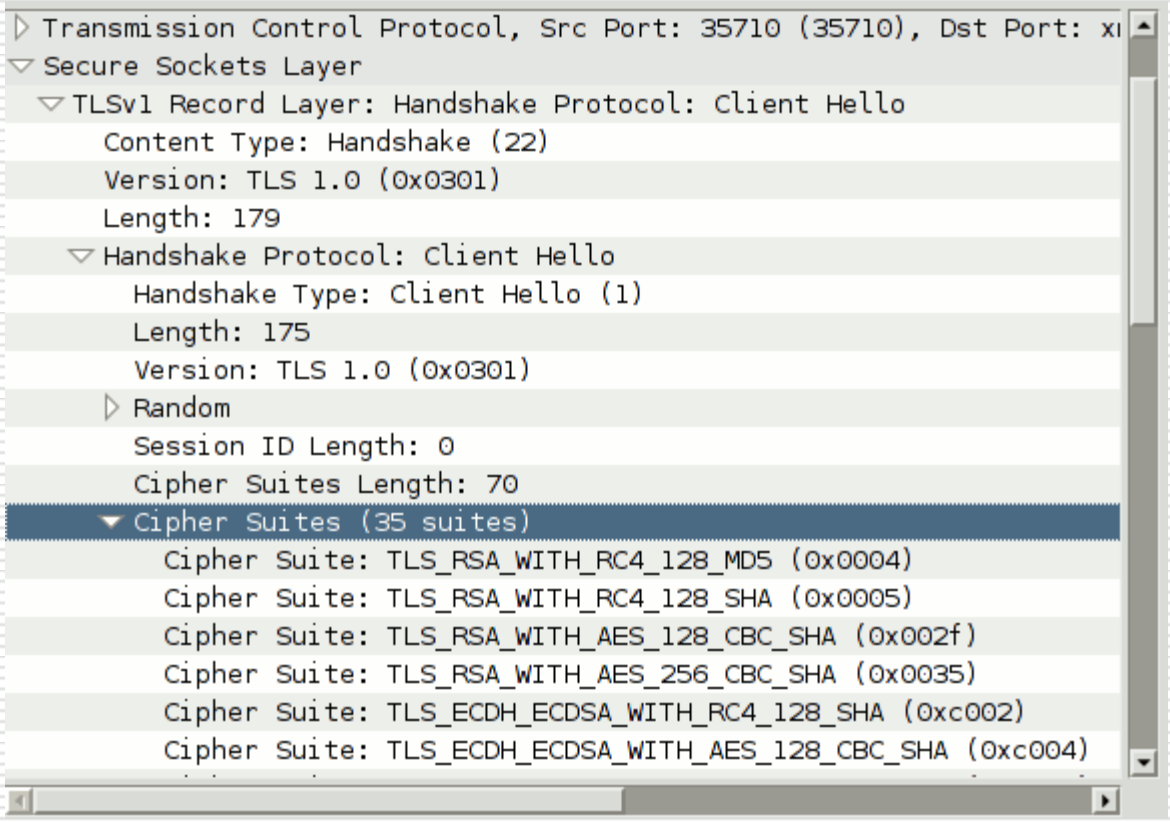


Гмейл курильщика



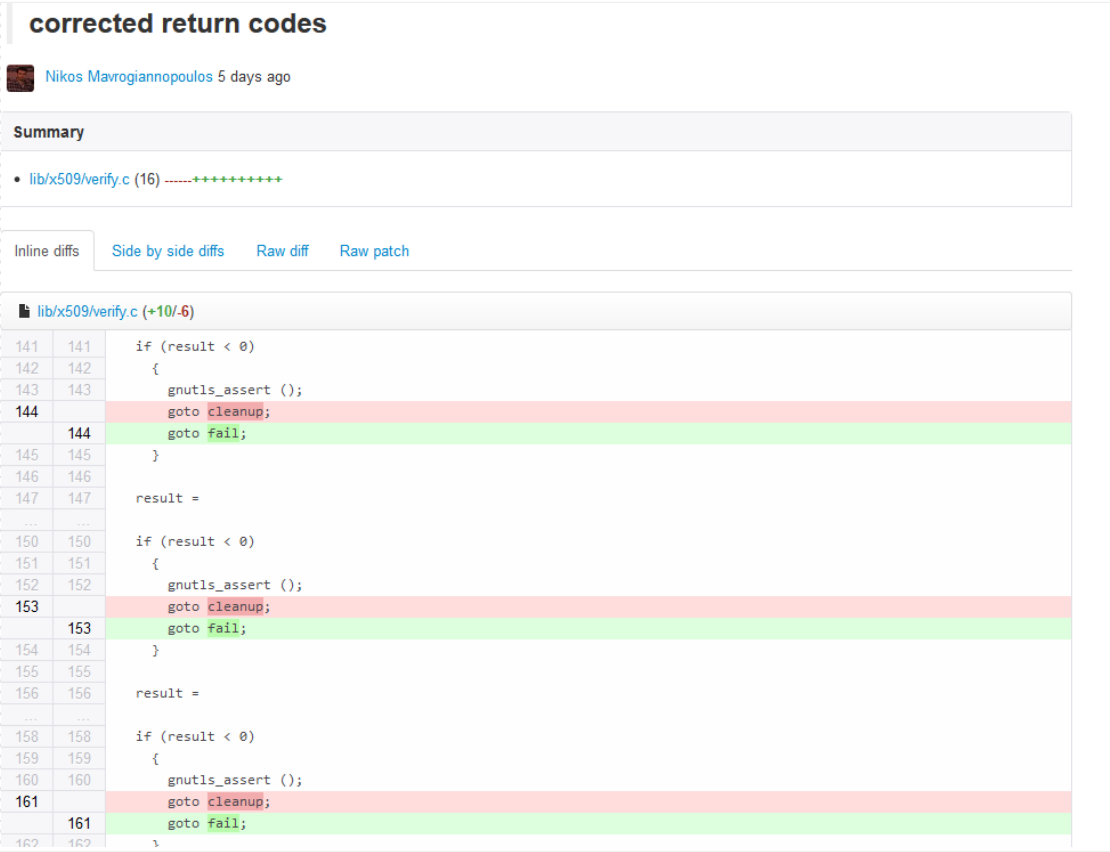
Ещё интересное

- http://op-co.de/blog/posts/android_ssl_downgrade/



GNU TLS: goto fail

- http://www.gnutls.org/security.html#GNUTLS-SA-2014-2
- https://www.gitorious.org/gnutls/gnutls/commit/6aa26f78150ccbdf0aec1878a41c17c41d358a3b



corrected return codes

Nikos Mavrogiannopoulos 5 days ago

Summary

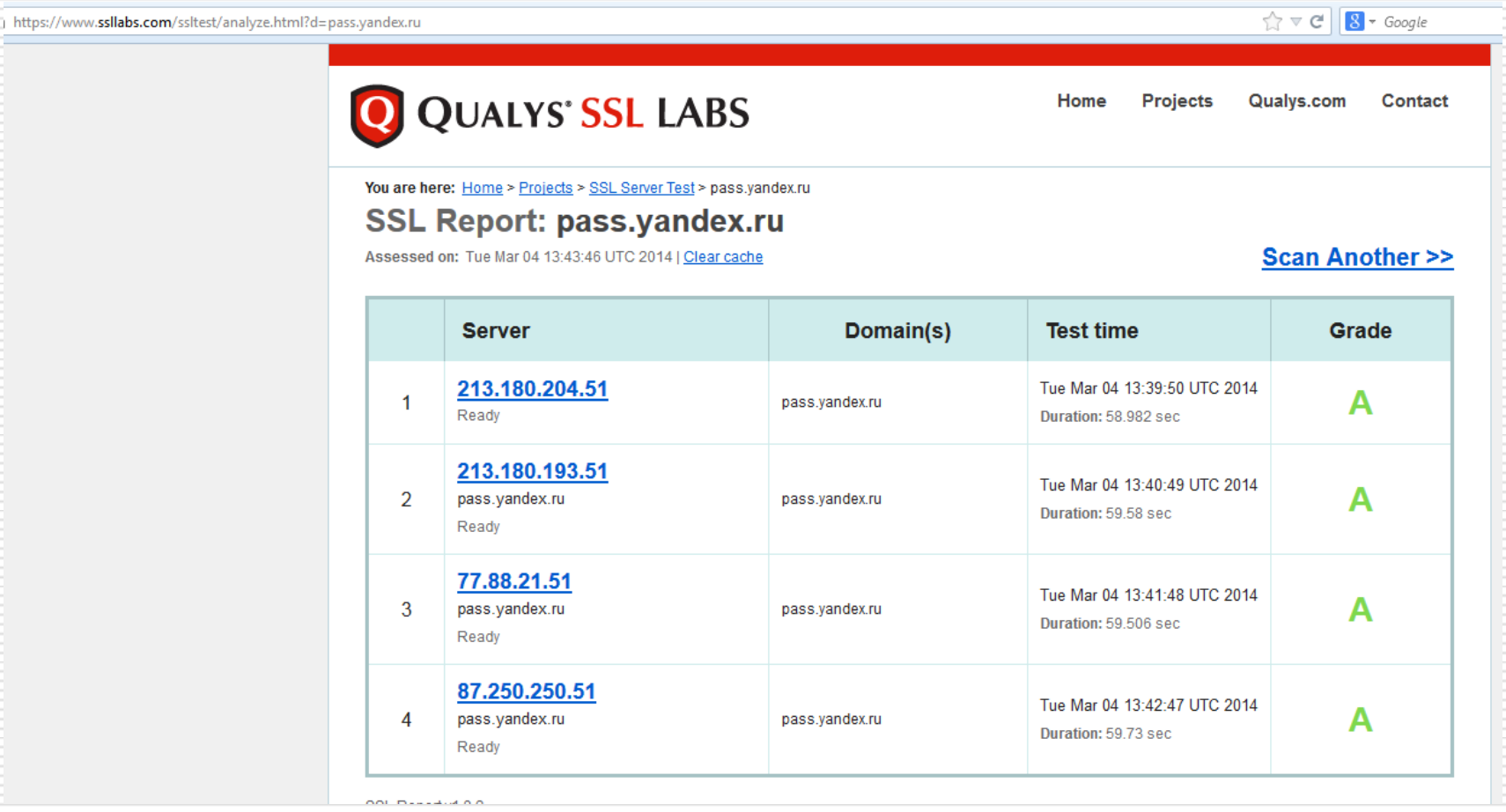
- lib/x509/verify.c (16) ------++++++++++

Inline diffs | Side by side diffs | Raw diff | Raw patch

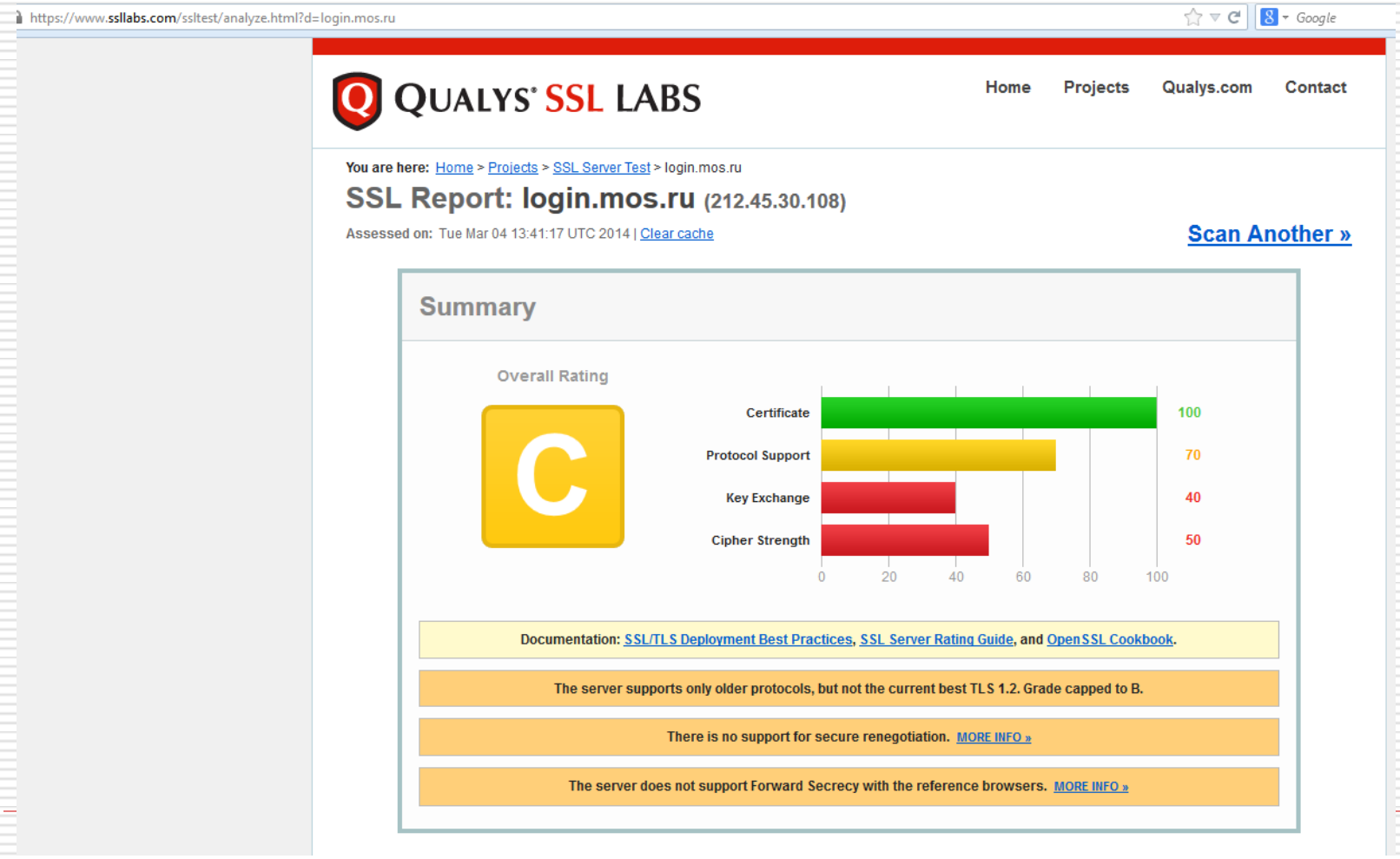
lib/x509/verify.c (+10/-6)

```
141 141   if (result < 0)
142 142     {
143 143       gnutls_assert ();
144         -     goto cleanup;
    144     +     goto fail;
145 145     }
146 146
147 147   result =
...
150 150   if (result < 0)
151 151     {
152 152       gnutls_assert ();
153         -     goto cleanup;
    153     +     goto fail;
154 154     }
155 155
156 156   result =
...
158 158   if (result < 0)
159 159     {
160 160       gnutls_assert ();
161         -     goto cleanup;
    161     +     goto fail;
```

Анализ HTTPS на pass.yandex.ru



Анализ HTTPS на login.mos.ru



Задание

- Настроить раздачу статической страницы с авторизацией по клиентскому SSL сертификату:
  - Nginx или apache
  - Генерация сертификатов – openssl + easy-rsa
  - Экспорт сертификатов в пригодном для браузеров виде