RSA Conference2015
San Francisco | April 20-24 | Moscone Center

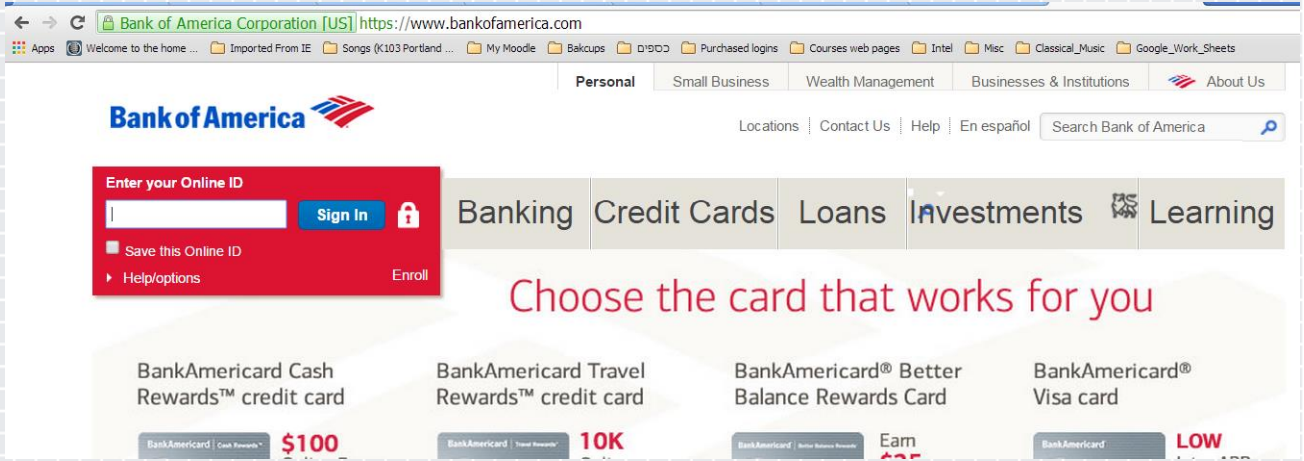
SESSION ID: DSP-F03

New Trends In Cryptographic Algorithm Suites Used For TLS Communications

Shay Gueron

Senior Principal Engineer, Intel Corporation
and Assoc. Professor, University of Haifa

What is the fate of a secure & private session if…



Someone listens & records

Or Subpoenaed ?

- And a ***year later*** the server's private key is compromised?
  Stolen, hacked, broken, leaked

Is the session still secure? It depends

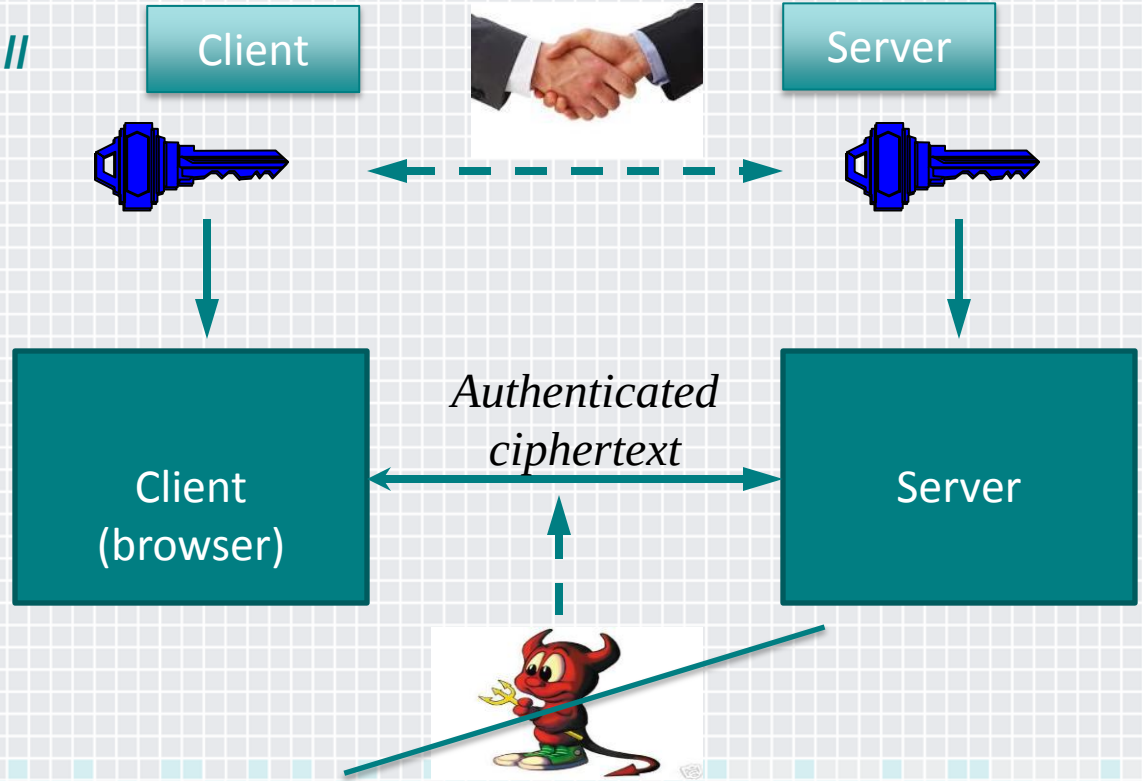
In this talk we will learn why and how

Perfect Forward Secrecy

- What is Perfect Forward Secrecy? (PFS)
- Do we have it?
- Should we care?
- What does it cost to get it?
- Is there an excuse for servers to not support PFS?

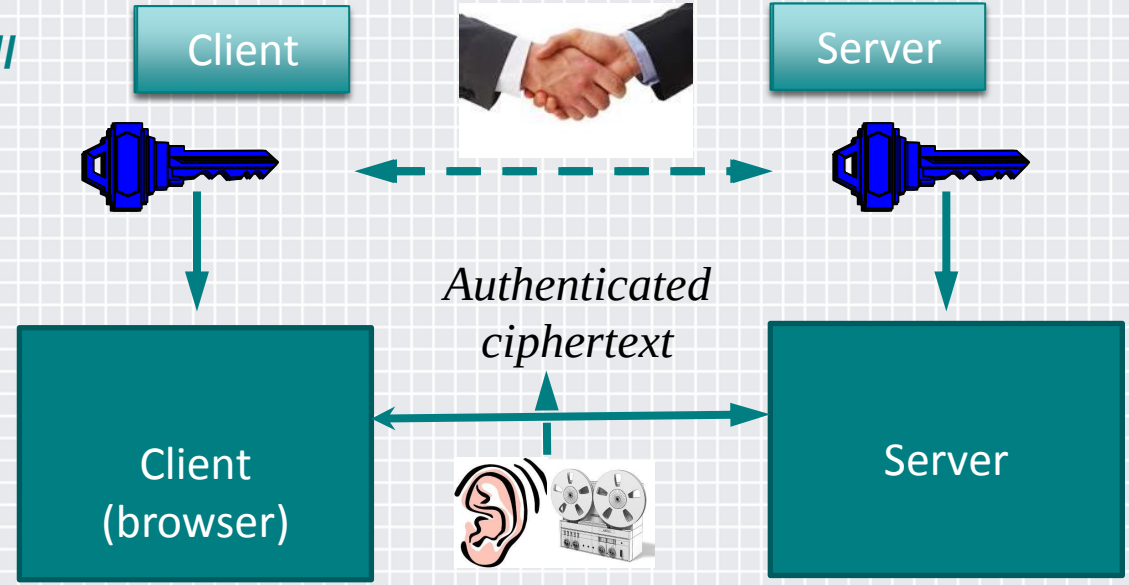
Client-server communications are secure

If we have https://



Client-server communications are eternally secure ??

If we have https://



Secure communication ***now***, does **not necessarily guarantee** the ***future*** privacy in a scenario where the server's private key could be ***compromised*** in the ***future***

Privacy? (I have nothing to hide)

- Privacy is often dismissed (or assumed, for granted, implicitly)
  - nobody is listening (?)
- But…do we want these to be observable (a-posteriori):
  - Browsing history: where we and what we searched?
  - Shopping list? Shopping preferences?
  - Salary? Bank transactions? Credit card history?
  - Addresses? Contacts?

Communication is all around…

- On a daily basis
- Multiple devices

… there is public awareness to security

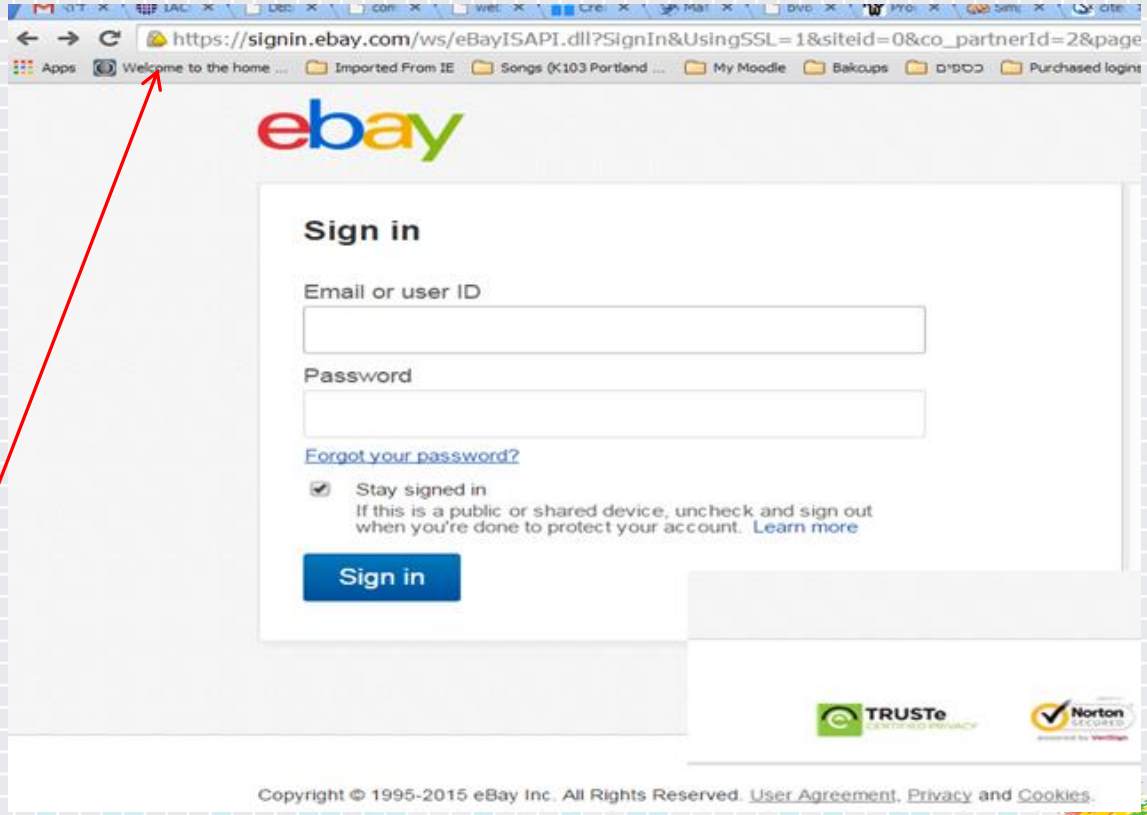
- We need security
  - We assume security
  - We expect security
- Do we also assume privacy?
  - Do we expect privacy?

Posted in the London Subway:
Only shop when the payment page has a padlock bar **://https**

Security is stated on shopping web sites

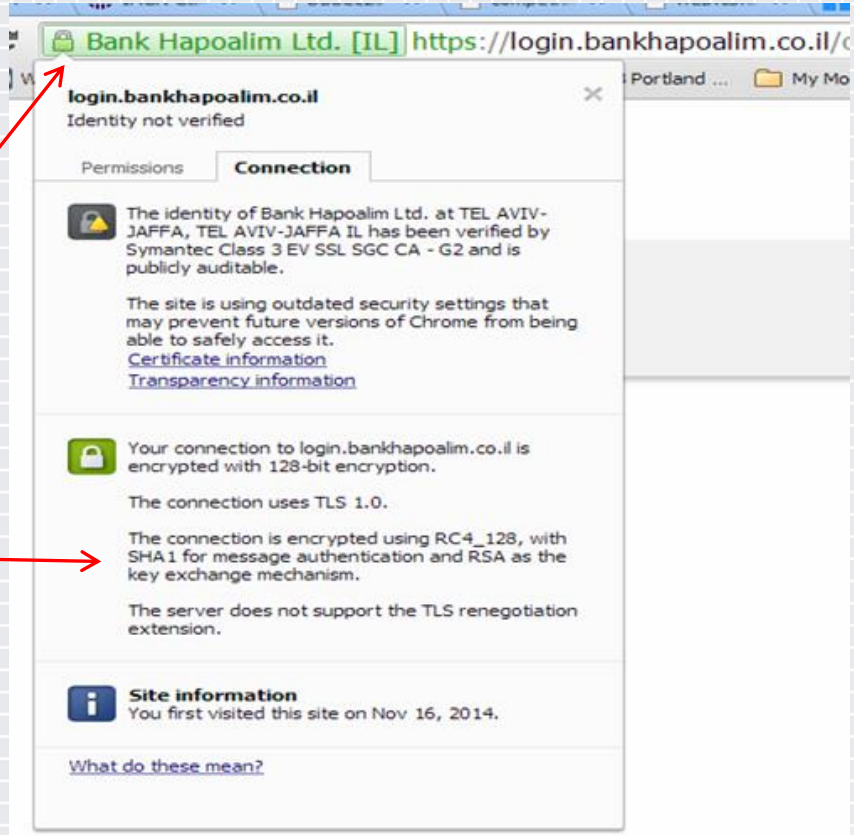
://https

… when payments are involved…



://https

… banks…


Bank Hapoalim Ltd. [IL]
https://login.bankhapoalim.co.il/
Portland …
My Mo
login.bankhapoalim.co.il
Identity not verified
Permissions
Connection
The identity of Bank Hapoalim Ltd. at TEL AVIV-JAFFA, TEL AVIV-JAFFA IL has been verified by Symantec Class 3 EV SSL SGC CA - G2 and is publicly auditable.
The site is using outdated security settings that may prevent future versions of Chrome from being able to safely access it.
Certificate information
Transparency information
Your connection to login.bankhapoalim.co.il is encrypted with 128-bit encryption.
The connection uses TLS 1.0.
The connection is encrypted using RC4_128, with SHA1 for message authentication and RSA as the key exchange mechanism.
The server does not support the TLS renegotiation extension.
Site information
You first visited this site on Nov 16, 2014.
What do these mean?

://https

If it's so good – why not encrypt everything?

://https

- Encryption is an overhead with high costs
- Large datacenter servers are extremely loaded
  - More computations ➔ higher response latencies
  - More computations ➔ more servers / higher electricity bill

Servers are aware of and are sensitive to computational overheads

TLS – Transport Layer Security (https://)

- A protocol layer that runs over a transport layer such as TCP
  - Server is identified to the client (browser)
  - privacy-&-integrity-protection to the communication
- SSL evolution:
  - (1996) SSL 3.0; (1999) TLS 1.0 ; (2006) TLS 1.1; (2008) TLS 1.2
- Known protocol weaknesses and security issues up to TLS 1.0
  - No known weaknesses for TLS 1.1 and (latest) TLS 1.2
- TSL 1.3 is being worked out (still a draft)

Recommendation is to **not use** TSL 1.0 and below; **use** TSL 1.2

A TLS session in a nutshell

- Handshake
- Pre master secret
- Master secret

- Client packets →
- ← Server packets

Server-client communications are encrypted (confidentiality) and authenticated (integrity)

The role of the handshake

- Server authenticates itself to client [via a certificate]
  - Client knows what server it is communicating with
- Client & server agree on the cipher suite to use in the session
  - Serve & client exchange a symmetric key
  - Symmetric key is used for encryption and authentication
- Handshake is based on Public Key Cryptography
  - Client needs a way to establish trust in the server's public key [certificate]

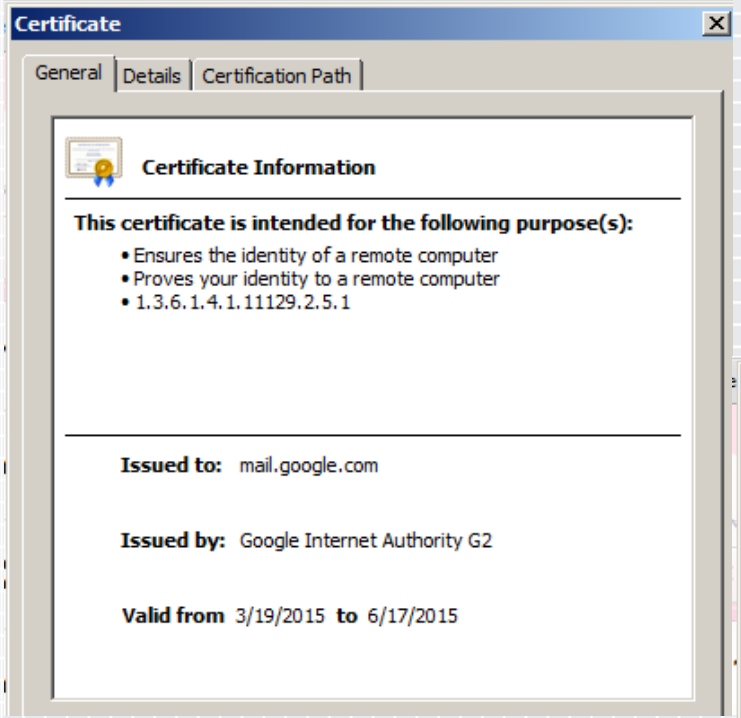
Authenticity and distribution of Public Keys

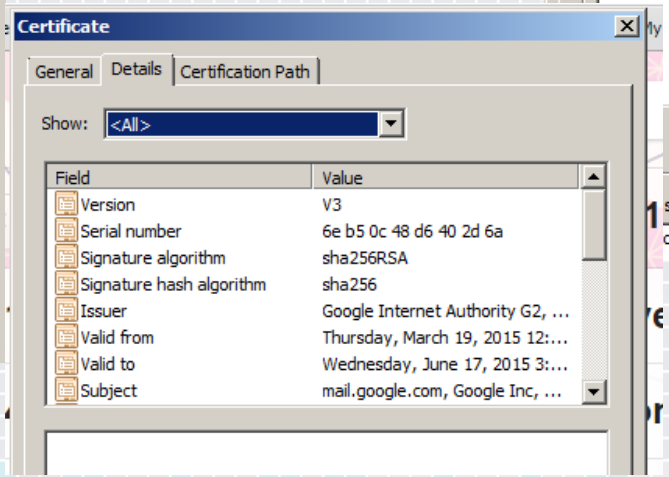
- The problem: how can a client know that the public it received during the handshake is really the public key of the server?
  - (and not of an impersonator)
- The certificate authority (CA) approach:
  - A trusted authority certifies public keys
  - Browsers are pre-configured with trusted CA's
  - Browser accepts public key of a website if certified by one of these CA's

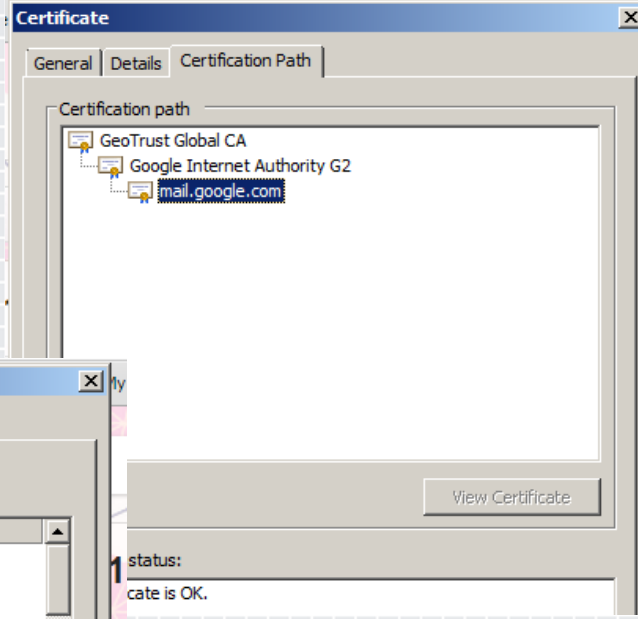
Certificate Authority hierarchy

- Browsers (and OS's) have trusted Root Certificate Authority
  - Hundreds of Root CA's
- Chain of trust:
  - Root CA signs certificates for intermediate CA's
  - Intermediate CA's sign certificates for lower-level CA's

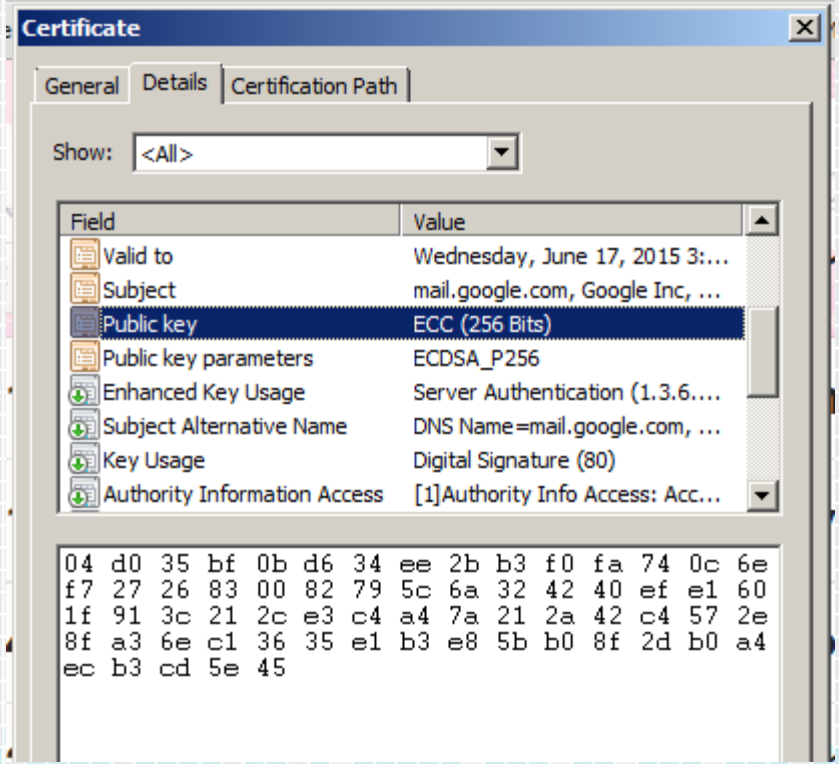
Example of a certificate







Example of a certificate



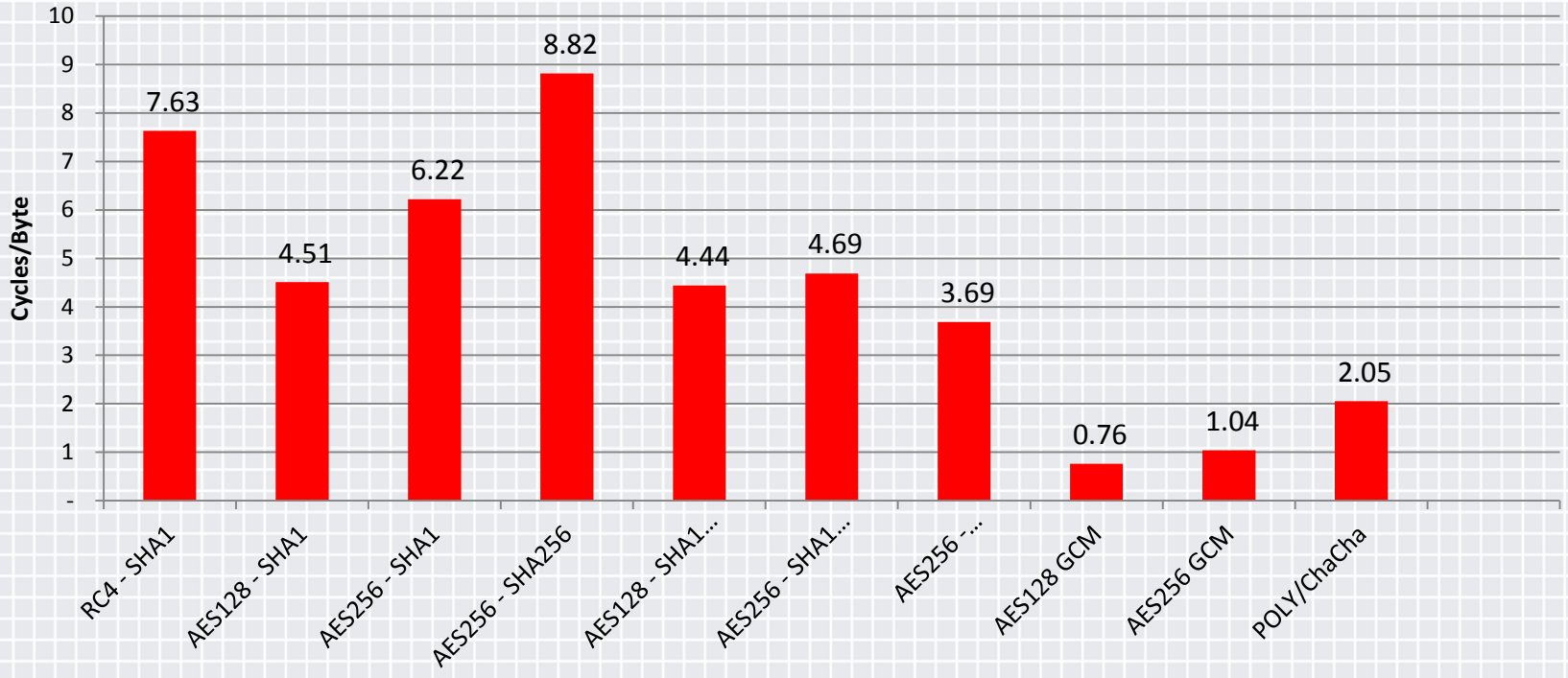
Some currently available/used cipher suites

Multiple options

- AES128-SHA — Cipher = AES128 CBC; MAC = HMAC SHA-1; Kxchg = RSA
- AES256-SHA — Cipher = AES256 CBC; MAC = HMAC SHA-1; Kxchg = RSA
- AES256-SHA256 — Cipher = AES256 CBC; MAC = HMAC SHA-256; Kxchg = RSA
- EDH-RSA-AES128-SHA — Cipher=AES128 CBC;MAC=HMAC SHA-256;Kxchg=EDH;Sign=RSA
- EECDH-RSA-AES128-GCM-SHA256 — AEAD Cipher = AES128 GCM; Kxchg = EECDH; Sign = RSA
- EECDH-ECDSA-AES128-GCM-SHA256 — AEAD Cipher = AES128 GCM; Kxchg = EECDH; Sign = ECDSA
- EECDH-RSA-CHACHA20-POLY1305 — AEAD Cipher = Poly1305-Chacha20;Kxchg = EECDH;Sign = RSA
- EECDH-ECDSA-CHACHA20-POLY1305 — AEAD Cipher=Poly1305-Chacha20; Kxchg=EECDH;Sign=ECDSA

Not all choices have the same efficiency

Performance of some Authenticated Encryption choices



Measured for 8KB message, on Architecture Codename “Broadwell”

RSA-based “classical” TLS handshake

- Client Hello
  - client random
  - client cipher suites preferences

- Server Hello
  - server random
  - selected cipher suite
- Server Certificate
  - server certificate
  - all relevant certificates in the chain

- Server Hello Done

- Client Key Exchange
  - **Use RSA to encrypted secret**

- Finished

Ephemeral Key-exchange TLS handshake

- Client Hello
  - client random
  - client cipher suites preferences

- Server Hello
  - server random
  - selected cipher suite

- Server Key Exchange
  - parameters (DHE parameters /ECDHE curve)
  - signed by the server

- Server Hello Done

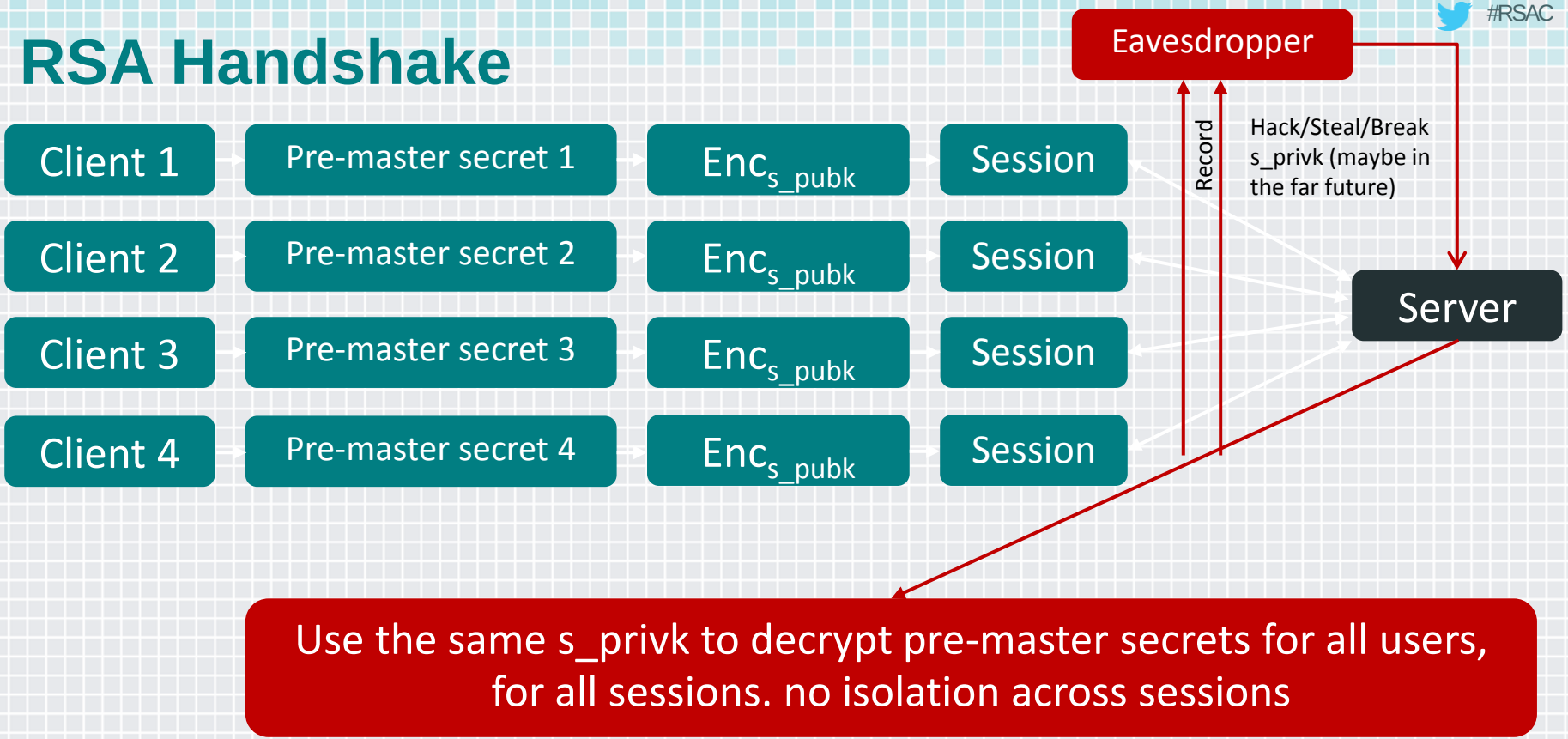
- Client Key Exchange
  - **DHE/ECDHE: public key**

- Finished

What is the difference in the handshakes?

- [Classical] RSA handshake:
  - Master secret is generated by client
  - Client encrypts the secret using server's public key from a certificate
  - Server decrypts the secret using server's private key
  - Session keys derived by applying a PRF to secret & server/client random
- DHE/ECDHE handshake:
  - Session key is agreed via (ephemeral) key exchange algorithm
  - Server's private key used only for signing the key exchange parameters

RSA Handshake



DH/ECDH Handshake

s_privk is used for authentication and not for secrecy;
it cannot be used for decrypting recorded traffic

Perfect Forward Secrecy

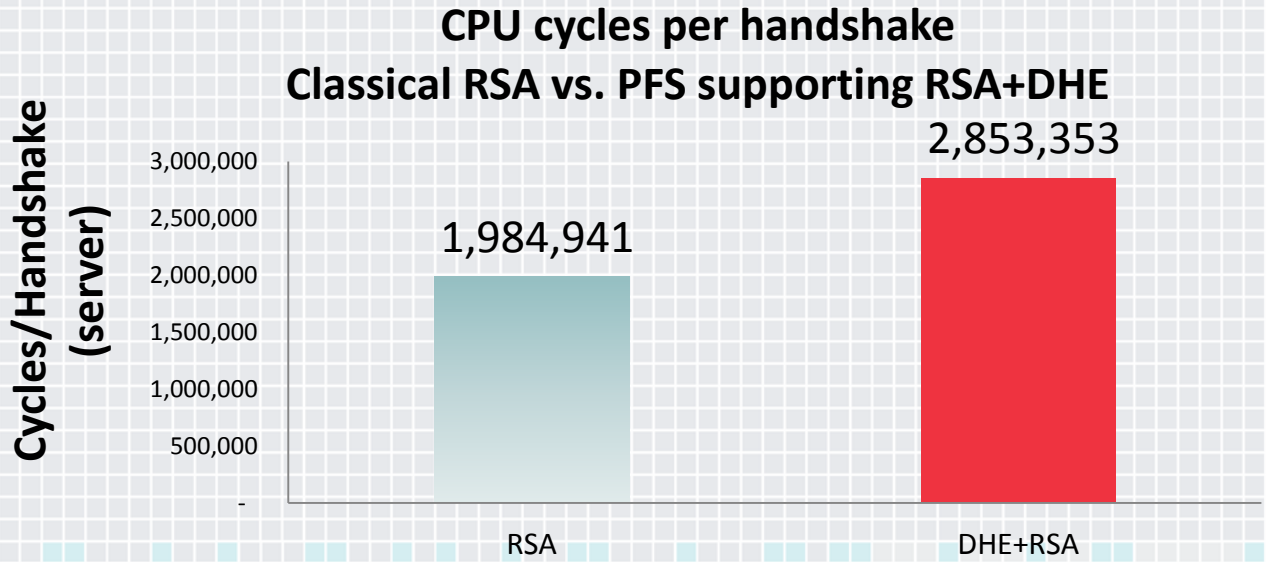
Fate of past secret traffic if the server's private key is compromised?

- With RSA key exchange: all past secrets are compromised
  - Privacy is lost for all users
- With DHE/ECDHE:
  - A man-in-the-middle attack possible (impersonating a server)
    - Can be mitigated (going forward)
  - But session keys used in past connections are not exposed
    - Each connection is protected by a unique and ephemeral session key

Perfect Forward Secrecy (PFS) protects the users' privacy
in a compromised server key scenario

Servers of the world, support PFS now! (?)

- Problem: simple move from “Classical RSA” to RSA+DHE to support PFS incurs ~50% performance penalty

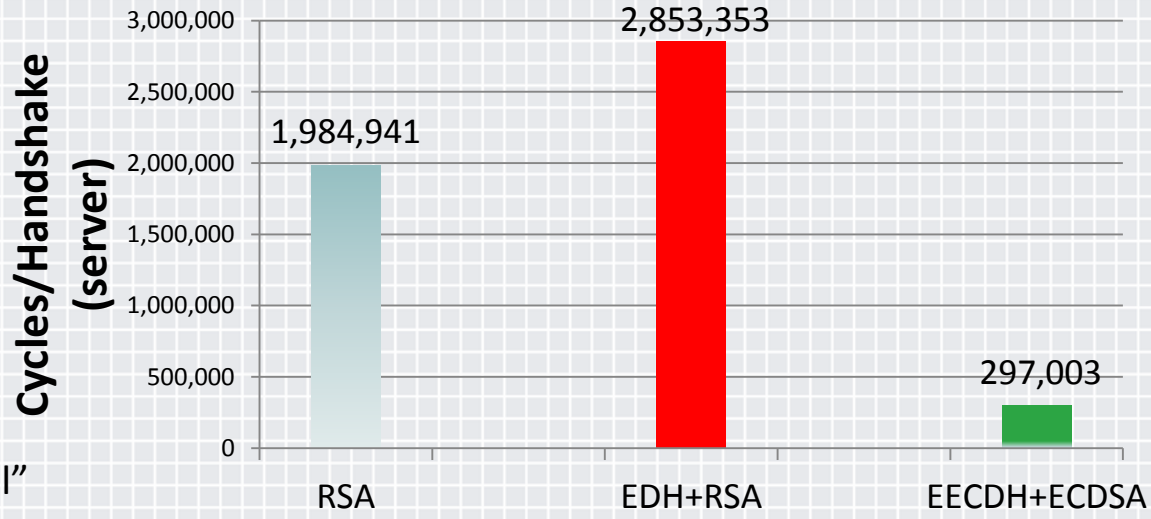


Reducing the overheads

- Elliptic Curves Cryptography (ECC) is faster than “RSA + DHE”
  - ECDH (for the key exchange)
  - ECDSA (for signatures)
- Recent algorithms improve ECC computations
- Easy migration to ECC based TLS?
  - Server needs an ECC certificate for ECDSA
  - But can use the existing RSA certificate for “RSA + ECDH”

Some new optimizations

- New optimizations speed up ECC computation by a factor of 3x
  - Software is free: integrated into OpenSSL 1.0.2
  - Download and update
- Surprising situation:
  New PFS supporting
  is ~6.7 times faster than
  the old non-PFS code



Measured on
Architecture Codename “Broadwell”

So, is there still an excuse for servers not supporting Perfect Forward Secrecy?

New trends: TLS 1.1 → TLS 1.2

(partial list)

- Getting rid of MD5:
- No backward compatibility with obsolete SSL (no SSL2.0)
- Client/server ability to specify acceptable hash/sign algorithms
- AEAD (authenticated encryption): AES-GCM & AES-CCM

New trends TLS 1.2 → (coming) TLS 1.3 (draft)

(partial list)

- Removing unused /unsafe features
  - SSL negotiation for backwards compatibility; compression.
  - Re-negotiation
  - RC4 and AES-CBC in MAC-then-Encrypt mode; Custom DHE groups (uses predefined groups)
- Improve privacy: Encrypt more of the handshake
- Improve latency: 0-RTT handshake for repeat connections

New trends TLS 1.2 → (coming) TLS 1.3

(3/2015 Draft)

(partial list)

- Mandatory PFS:
  - Removed support for static RSA and DH key exchange.
  - Can use RSA certificates - but with ECDHE or DHE (ECDHE minimizes performance hit)
- Removed support for non-AEAD ciphers.
  - Current AEAD ciphers for TLS: AES-GCM, AES-CCM, ARIA-GCM, Camellia-GCM, Poly/ChaCha

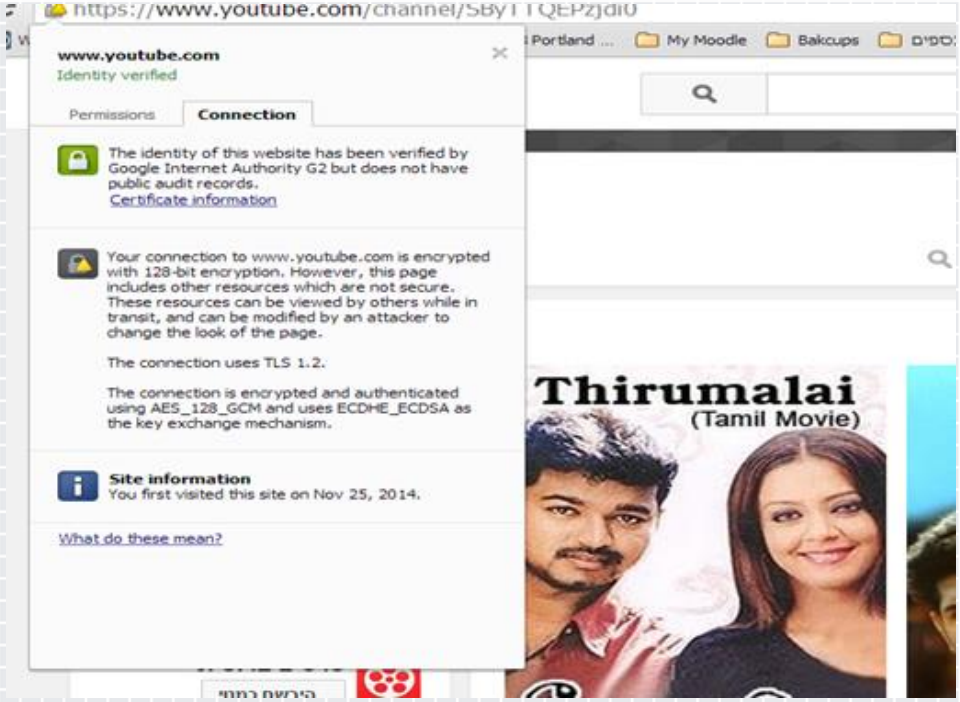
What is the situation in the field today?

Good examples

Updated TLS 1.2 and PFS support

What is the situation in the field today?

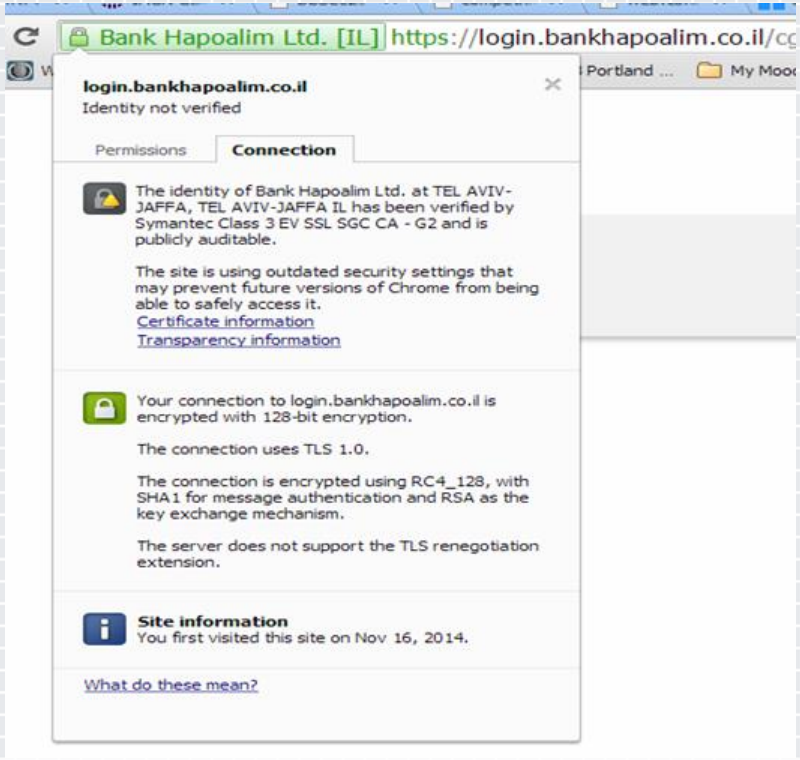
Good examples

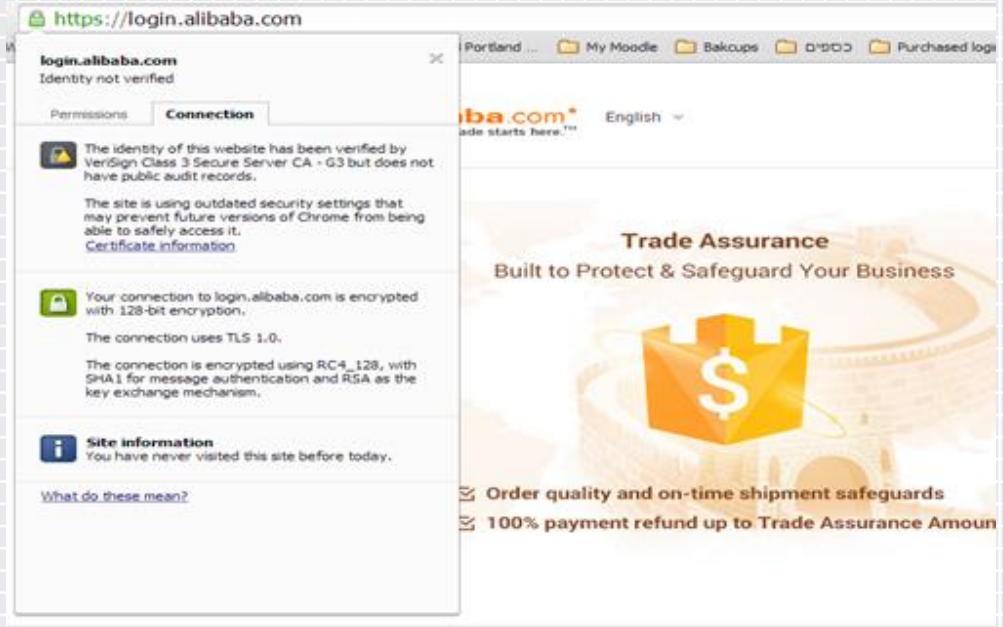


Updated TLS 1.2 and PFS support

What is the situation in the field today?

Not yet there

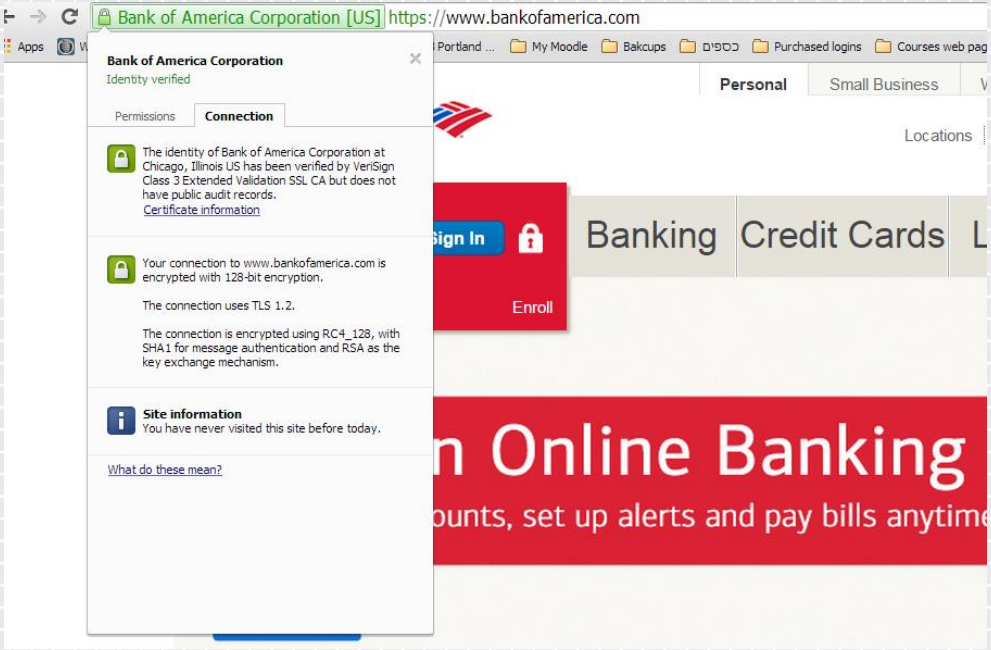


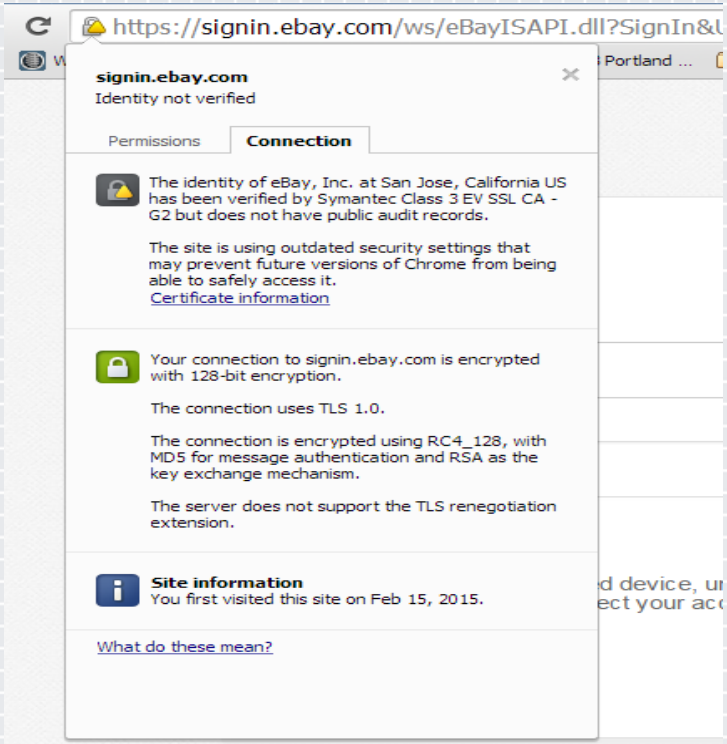


Outdated TLS and no PFS support

What is the situation in the field today?

Not yet there





Updated / Outdated TLS
no PFS support

What can I do with all this information?

- Be aware of security/privacy offered servers you access
  - Outdate TLS is not a good sign
  - Prefer providers who offer better privacy protection
- Setup your server
  - Use updated TLS (1.2) and update crypto library routinely
  - Configure server to prefer optimized ciphers and PFS support
  - Choose AES-GCM as top priority (on processors with AES-NI support)
  - Opt for ECDSA when obtaining a new certificate

Configure your server

- Set your server to the following cipher preferences (in order of preference) :
  - ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA256:DHE-DSS-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-DSS-AES128-SHA256
- Use this in Apache* configuration:

```
SSLProtocol -all +TLSv1.2
SSLCipherSuite AES128+EECDH+ECDSA:AES128+EECDH:AES128+EDH:-SSLv3
SSLHonorCipherOrder on
```

- Use this in nginx* configuration:

```
ssl_protocols TLSv1.2
ssl_ciphers AES128+EECDH+ECDSA:AES128+EECDH:AES128+EDH:-SSLv3
ssl_prefer_server_ciphers on
```

Summary

- Communication security $\not\Rightarrow$ Perfect Forward Secrecy
  - PFS property depends on client-server handshake
  - Most browsers offer a PFS handshake
  - But the server makes the call
- New solution makes PFS almost 7x faster than classical non-PFS
  - Available in free (open source) libraries
- Servers should use updated TLS
  - Avoid known vulnerabilities and enjoy bets features
- No excuse for servers to not support Perfect Forward Secrecy

Thank you for your attention

Feedback? Questions?

Contacts: Shay Gueron	
Senior Principal Engineer, Intel Corporation, Israel Development Center Shay.gueron@intel.com	Assoc. Professor, University of Haifa, Israel Dept. of Mathematics shay@math.haifa.ac.il

Reading (forward secrecy)

- https://community.qualys.com/blogs/securitylabs/2013/06/25/ssl-labs-deploying-forward-secrecy
- http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html
- https://www.eff.org/deeplinks/2013/08/pushing-perfect-forward-secrecy-important-web-privacy-protection
- http://www.perfectforwardsecrecy.com/
- https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy
- https://scotthelme.co.uk/perfect-forward-secrecy/
- http://googleonlinesecurity.blogspot.com/2011/11/protecting-data-for-long-term-with.html
- https://www.digicert.com/ssl-support/ssl-enabling-perfect-forward-secrecy.htm
- http://www.computerworld.com/article/2473792/encryption/perfect-forward-secrecy-can-block-the-nsa-from-secure-web-pages--but-no-one-uses-it.html

Reading (software optimization)

- S. Gueron, Efficient Software Implementations of Modular Exponentiation, Journal of Cryptographic Engineering 2:31-43 (2012).
- S. Gueron, V. Krasnov, Software Implementation of Modular Exponentiation, Using Advanced Vector Instructions Architectures, Proceedings of The International Workshop on the Arithmetic of Finite Fields (WAIFI 2012), LNCS 7369: 119-135 (2012).
- S. Gueron, V. Krasnov, Fast Prime Field Elliptic Curve Cryptography with 256 Bit Primes, Journal of Cryptographic Engineering (Nov. 2014).