CSE 551 TCPCrypt

Anna Kornfeld Simpson
Qiao Zhang
Danyang Zhuo
5/27/2015

Why we need web security?

Motivating Example:

Alice makes an online purchase using credit card.

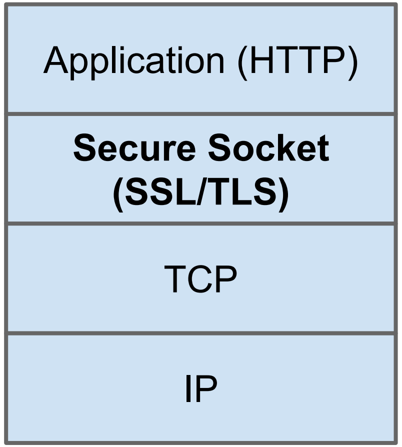
Desired Guarantees:

Confidentiality -- Mallory cannot intercept credit card info and use it to make unauthorized purchases.

Integrity -- Mallory cannot modify the transaction details.

Authentication -- Transaction is sent to the right vendor, not some other party.

Secure Transport: SSL/TLS



SSL/TLS is a user process (e.g. OpenSSL lib)

Interposes between Application and Transport Layers

modified application

unmodified TCP

Provides a secure TCP channel between two parties

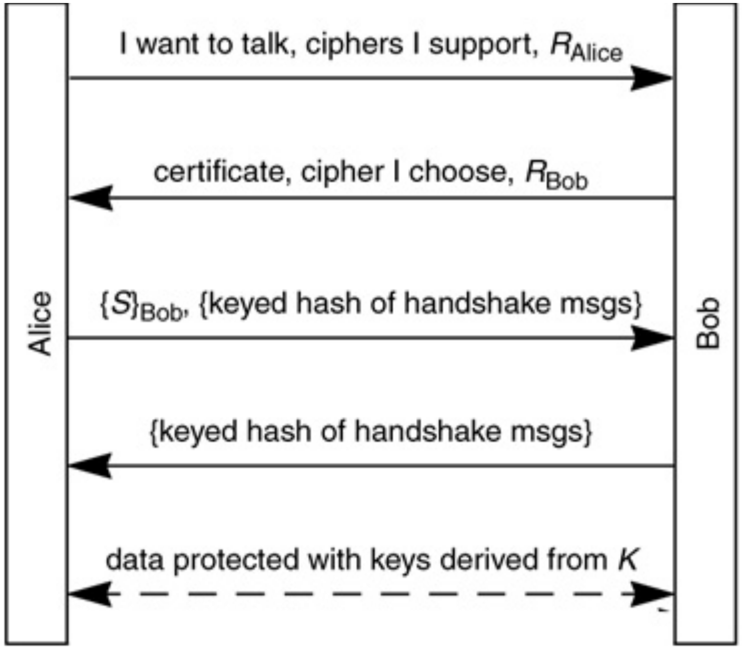
Application-level protocols: PGP, SSH etc.

SSLv3/TLS handshake

Msg1: clientHello, supported ciphers, client nonce

choose secret S, compute $K = f(S, R_{Alice}, R_{Bob})$

Msg3: secret encrypted with server's public key, keyed hash of prev msgs to ensure cipher suite not downgraded!



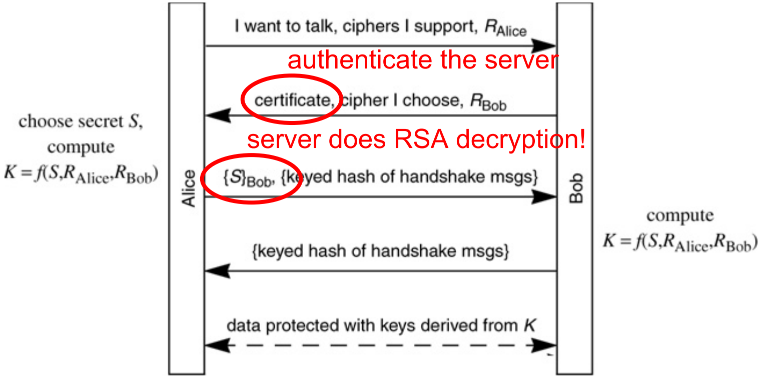
Simplified SSLv3/TLS (Kaufman's *Network Security)*

Msg2: serverHello, chain of certificates, chosen cipher, server nonce

compute $K = f(S, R_{Alice}, R_{Bob})$

Msg4:
keyed hash of prev msgs to prove
no-tempering of prev msgs
+ server has private key
+ server knows session keys derived from K

SSLv3/TLS cannot be ubiquitous

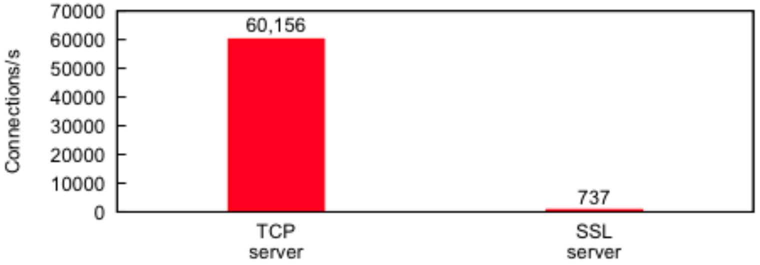


Simplified SSLv3/TLS (Kaufman's *Network Security)*

SSL/TLS achieves

1. authentication (usually only server to client)
2. data integrity through crypto hash
3. confidentiality through symmetric encryption

SSLv3/TLS cannot be ubiquitous



code snippet for openSSL?

However, SSL/TLS is

1. too expensive due to server-side decryption
2. hard to set up and use
   a. library not easy to use
   b. certificate is a pain point
3. not suitable for all applications
4. cannot get encryption/integrity without authentication

   e.g. if no server certificate, no encryption of data

tcpcrypt: encrypt all TCP traffic

Symmetric key encryption is cheap
	=> we can feasibly encrypt all TCP traffic

Confidentiality/Integrity are general-purpose primitives
Authentication is application-specific!
	=> we should decouple confidentiality/integrity with authentication

Tcpcrypt proposes a new architecture:
	Embed encryption/integrity checking into TCP as TCP extensions
	Provide hooks to enable flexible application level authentication

tcpcrypt security guarantees

By default, all TCP traffic are encrypted

- Protects against passive eavesdropping without ANY app modification
- But can be man-in-the-middled

Tcpcrypt key exchange generates a session id on both end points

- If the session ids match, then guarantee no MitM.
- Session id can be used for authentication

tcpcrypt vs SSL

High server performance: push decryption to clients

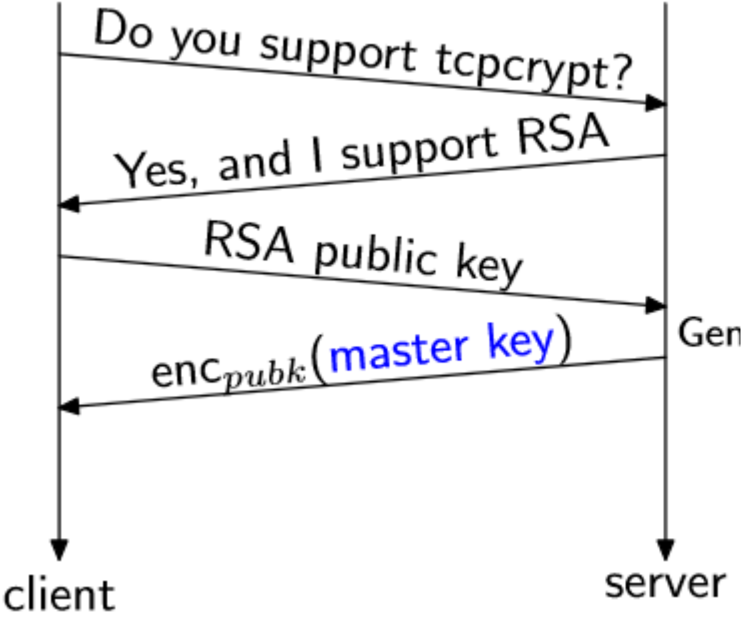
As a TCP option

- Applications use BSD socket API
- Encryption automatically enabled if both end points support tcpcrypt
- Backwards compatible, graceful fallback to vanilla TCP/SSL

New getsockopt() returns session_id as hook for authentication

- certificate-based authentication
- password-based authentication

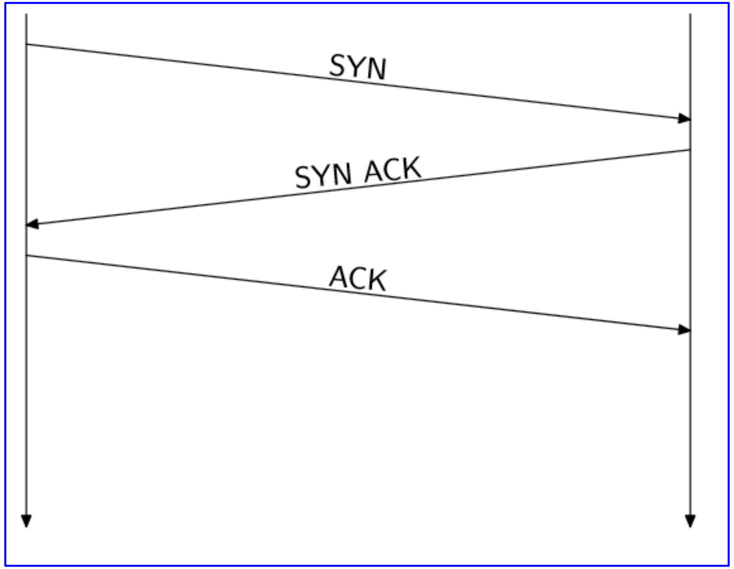
tcpcrypt handshake



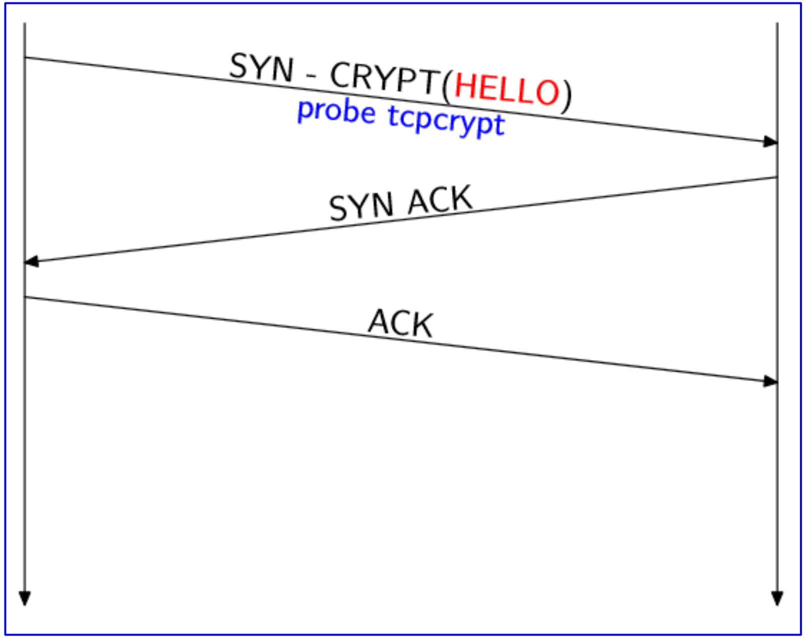
- Server sends supported public ciphers
- Client sends public key & supported symmetric ciphers
- Server generates & encrypts **symmetric** master key under client's public key
- Client decrypts and now both sides have the symmetric key

Putting the Handshake in TCP

Regular TCP setup

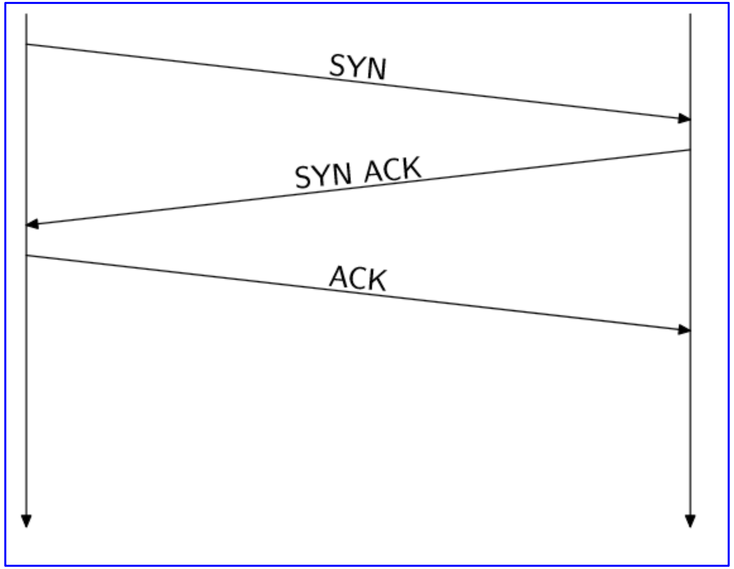


tcpcrypt part 1

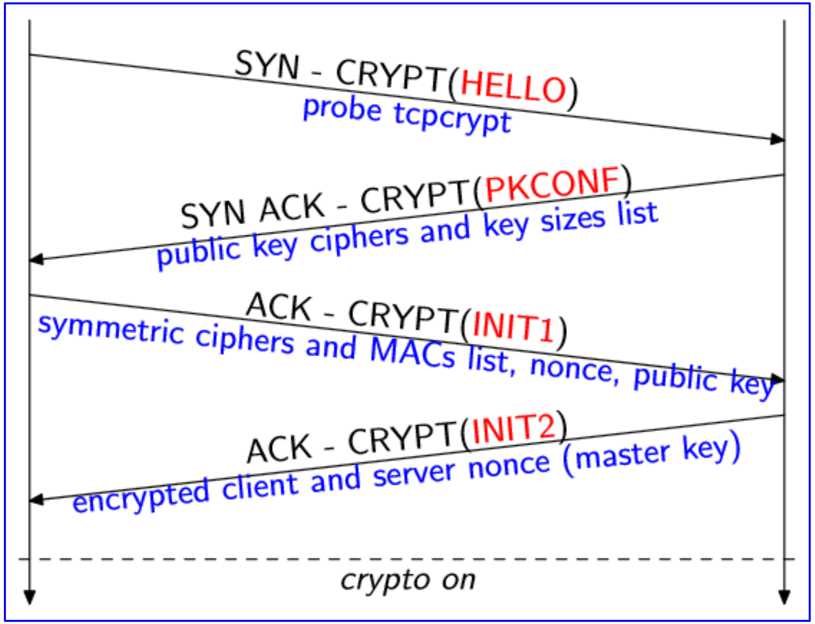


Putting the Handshake in TCP

Regular TCP setup

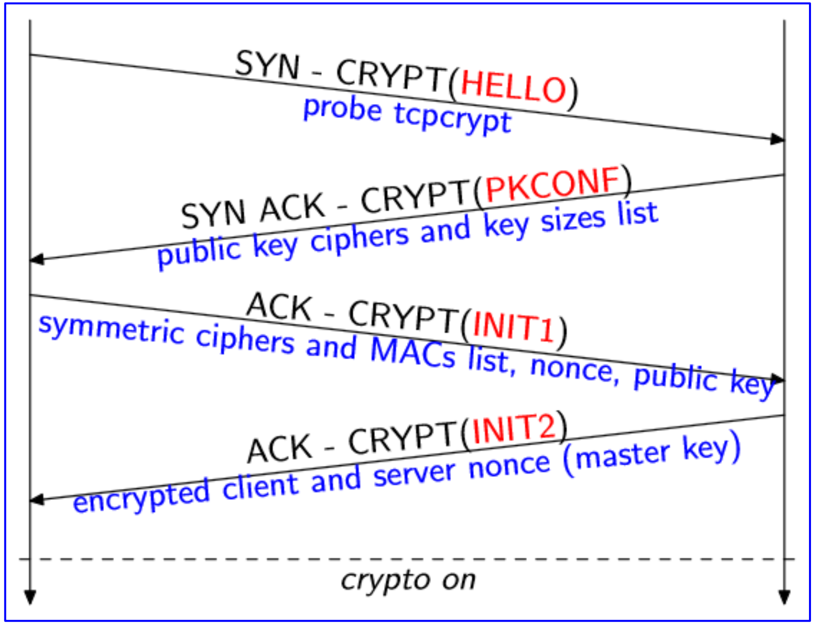


tcpcrypt full handshake



Only 1 extra message? How?

tcpcrypt full handshake



- First two messages encode info in TCP options
- INIT1 and INIT2 too large for options, so have to use application data
- Delays application data by one RTT (from 3rd message to below “crypto on”)

tcp options

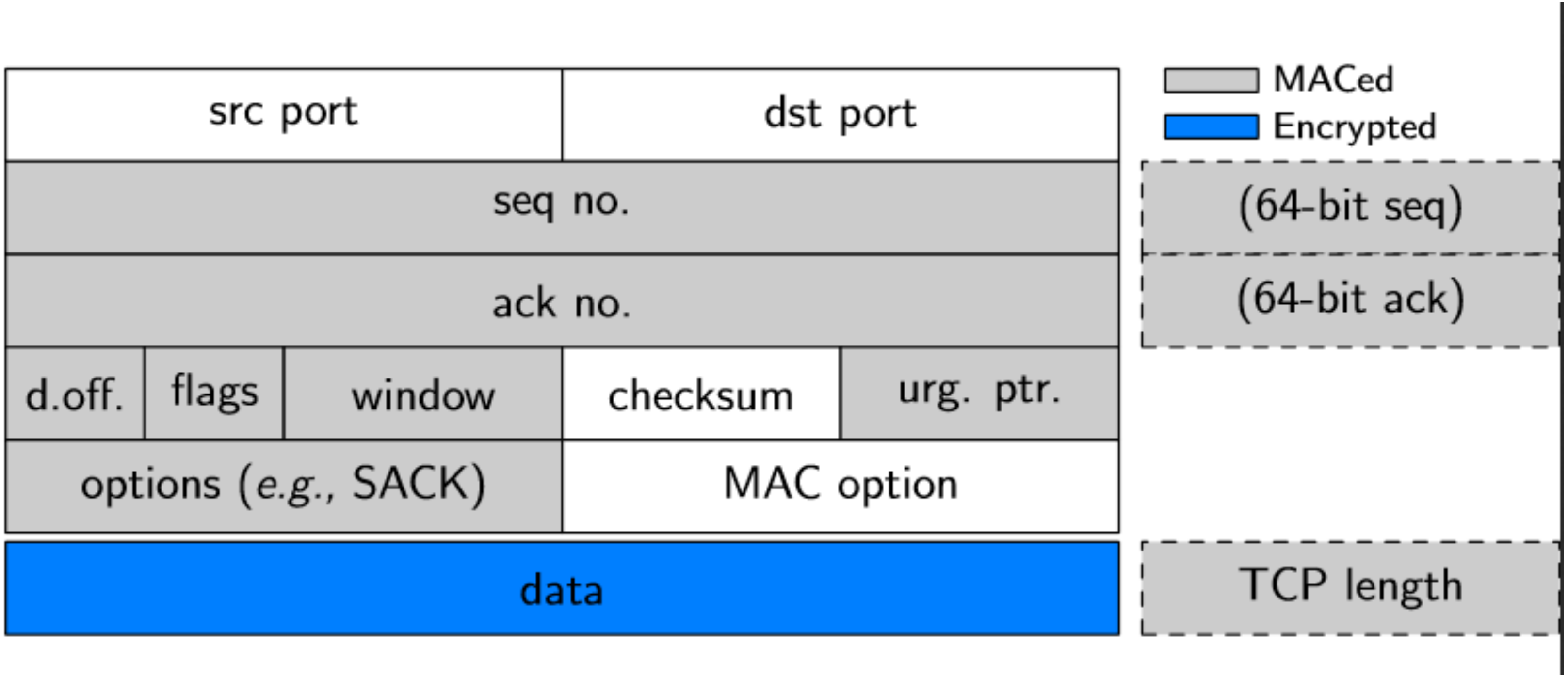
src: wikipedia

TCP Header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><th>4</th><th>32</th><td colspan="32">Sequence number</td></tr>
<tr><th>8</th><th>64</th><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><th>12</th><th>96</th><td colspan="4">Data offset</td><td colspan="3">Reserved
0 0 0</td><td>N
S</td><td>C
W
R</td><td>E
C
E</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="16">Window Size</td></tr>
<tr><th>16</th><th>128</th><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><th>20
...</th><th>160
...</th><td colspan="32">Options (if data offset > 5. Padded at the end with "0" bytes if necessary.)
...</td></tr>
</table>

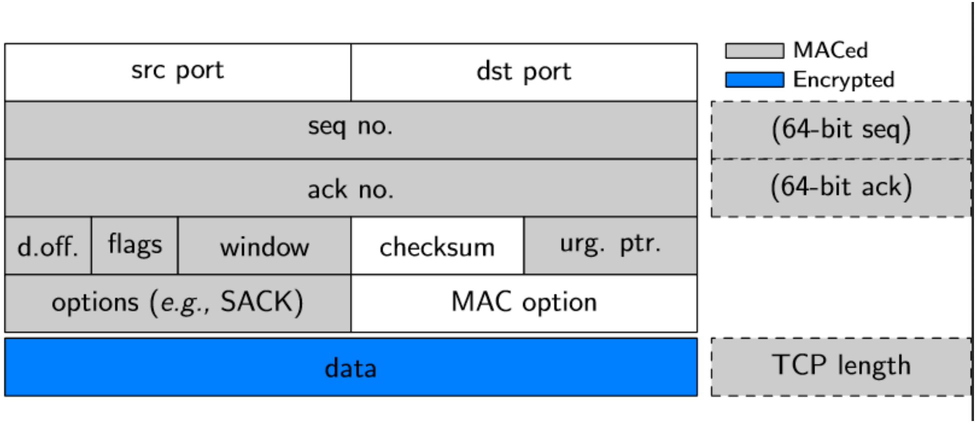
Options can have up to 40 bytes.

A tcpcrypt encrypted packet



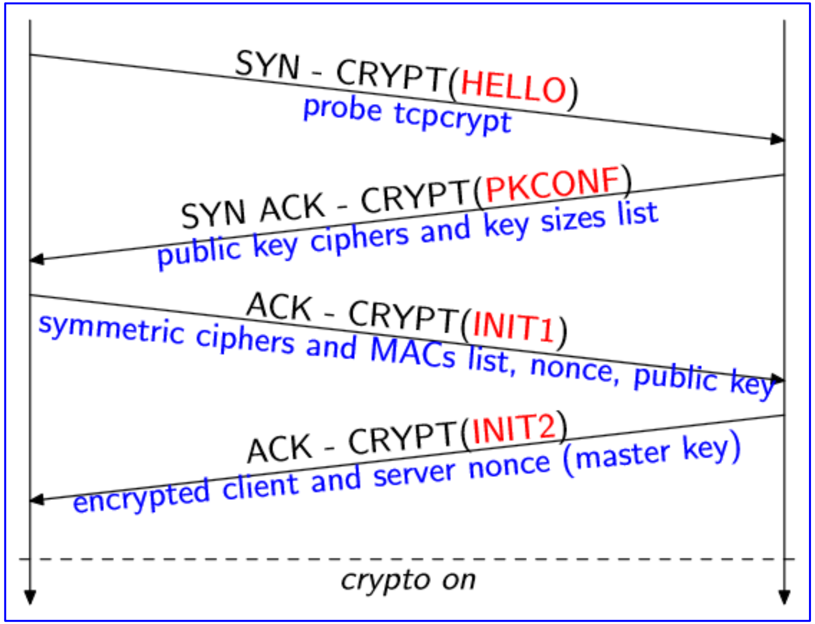
A tcpcrypt encrypted packet

- Data is encrypted (confidentiality)
- Headers + data are MAC'd (integrity)
- Also MAC dashed items (not in header)
- Don't MAC ports, checksum so NATs still work

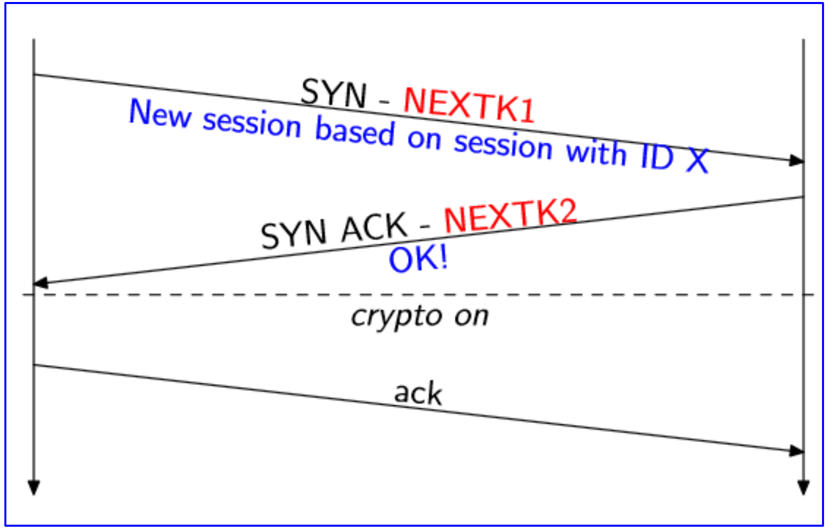


Session resumption - no latency!

tcpcrypt init handshake



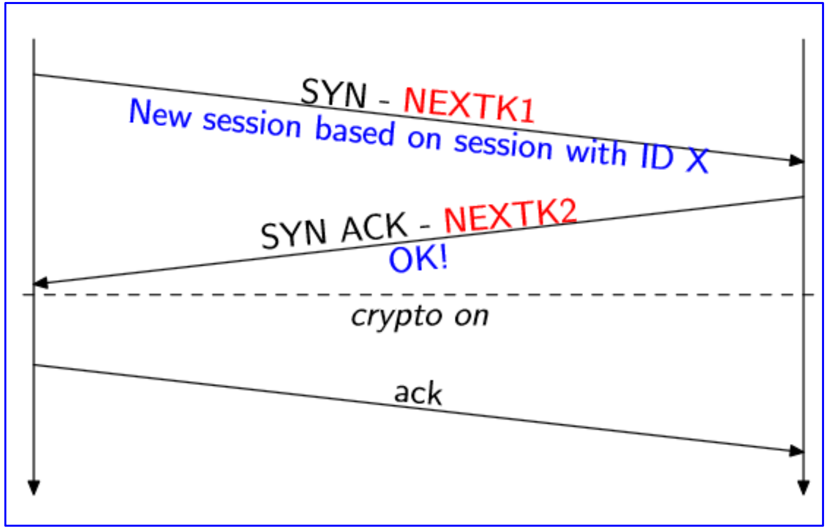
Session Resume



Great, how does that work?

- Session caching and session ID like in SSL
- NEXTK1 = 9 bytes of next session ID
- Fall back to full handshake if cache miss

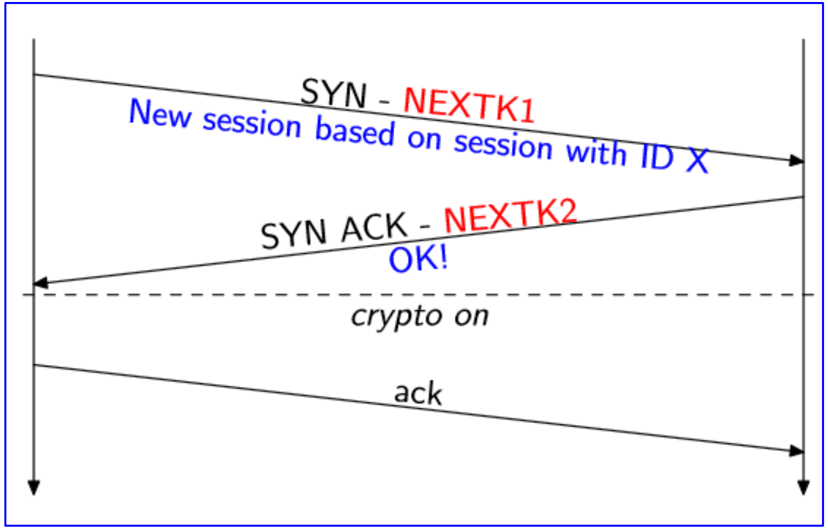
Session Resume



Great, how does that work?

- Session ID calculated by HMAC'ing a value with the session secret
- Session secret built during init by HMAC'ing the initial parameters and then iterate over time
- HMAC is a keyed hashing algorithm.

Session Resume



Certificate-based Authentication

- Server signs the shared Session ID with its private key
- Batch Sign
  - SIGN requires encryption using server’s private key which is as expensive as standard SSL
  - Server can sign a batch of Session IDs to amortize of the cost of SIGN function
  - Session ID is not secret

Weak Password Authentication

- Client and server share a secret
- C->S
  - MAC(hash(salt, realm, secret), TAG_C || Session ID)
- S->C
  - MAC(hash(salt, realm, secret), TAG_S || Session ID)
- Client and server can verify if the other end knows hash(salt, realm, secret) or not

Strong Password Authentication

- In Weak Password Authentication, adversary can impersonate the server to get the h = hash(salt, realm, secret) and then use a dictionary to guess the secret
- Use Diffie-Hellman problem to generate a shared key
  - h0 = H0(password, user name, server name)
  - h1 = H1(password, user name, server name)
  - g is a generator of group G (order of G is a prime number q)
  - U,V are randomly chosen in G

Strong Password Authentication

Server:

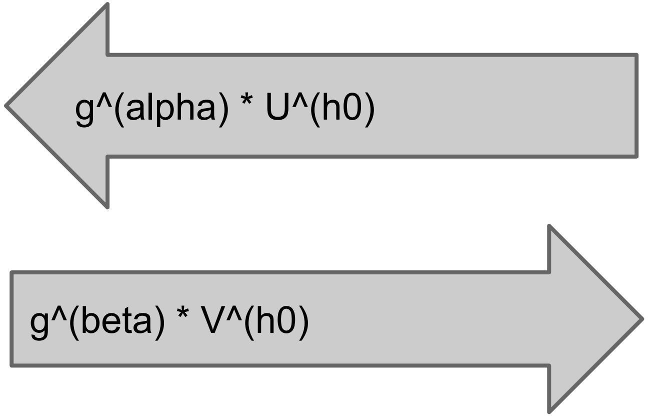
g, h0, g^h1, U, V, beta

Client:

g, h0, h1, U, V, alpha

Strong Password Authentication

Server:	Client:
g, h0, g^h1, U, V, beta	g, h0, h1, U, V, alpha



Strong Password Authentication

Server:

g, h0, g^h1, U, V, beta

g^(alpha) * U^(h0)

Client:

g, h0, h1, U, V, alpha

g^(beta) * V^(h0)

Strong Password Authentication

Server:

g, h0, g^h1, U, V, beta

g^(alpha)

Client:

g, h0, h1, U, V, alpha

g^(beta)

Strong Password Authentication

Server:

g, h0, g^h1, U, V, beta

g^(alpha)

g^(alpha*beta)

g^(beta*h1)

Client:

g, h0, h1, U, V, alpha

g^(beta)

g^(alpha*beta)

g^(beta*h1)

Strong Password Authentication

Server:

g, h0, g^h1, U, V, beta

g^(alpha)

Client:

g, h0, h1, U, V, alpha

g^(beta)

h = H(h0, g^(alpha), g^(beta), g^(alpha*beta), g(beta*h1))

C->S : MAC(h, TAG_C || Session ID)

S->C : MAC(h, TAG_S || Session ID)

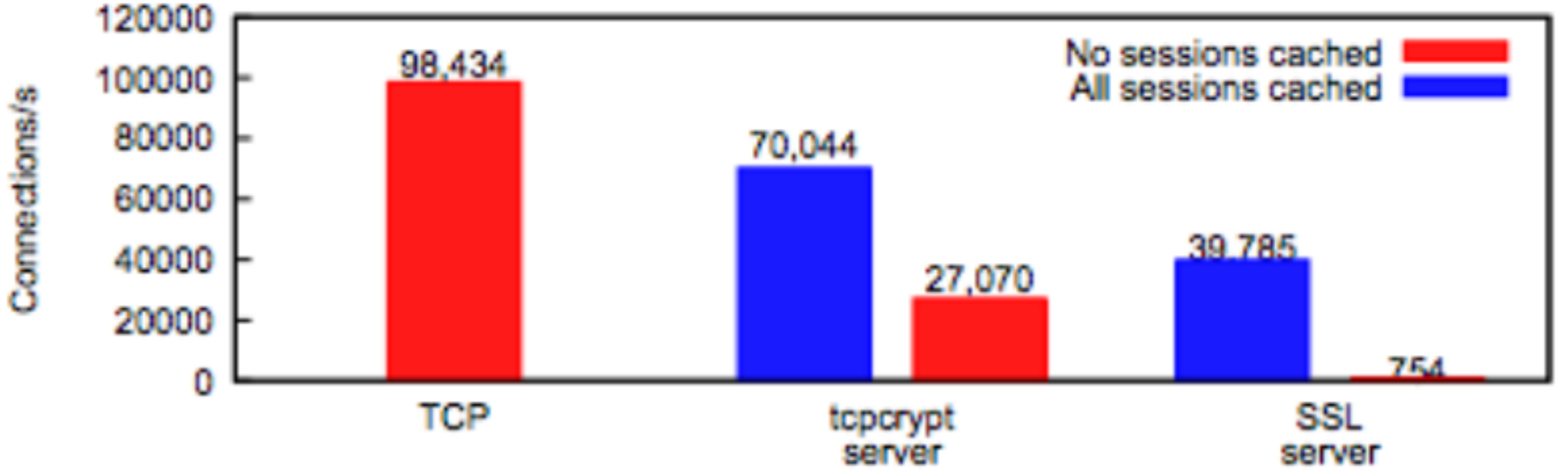
Implementation

- Linux Kernel Implementation
  - Port OpenSSL into kernel for RSA support
  - Incompatible with TCP segment offloading
- Userspace Implementation
  - Use divert socket to access TCP packets
  - Track connections, calculate checksum, rewrite sequence number,...
  - IPC call for getsockopt

Implementation

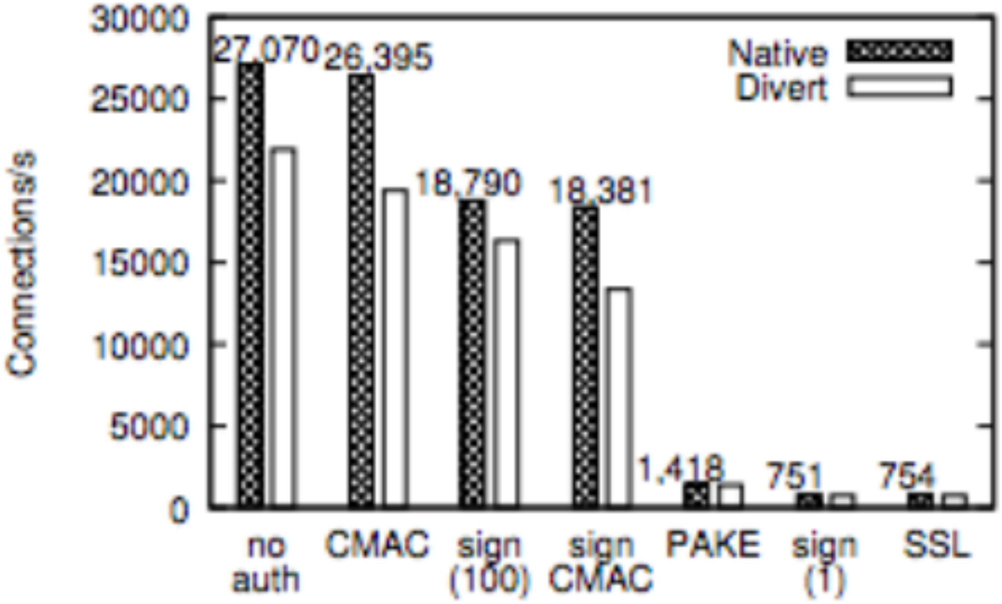
- OpenSSL
    - Modify OpenSSL’s BIO layer to leverage the shared Session ID
    - Use a single worker thread to batch sign all incoming SSL connections

Connection Rate



Main bottleneck is public key operations. RSA decryption is much slower than encryption.

Authentication Cost



Authentication can be 25x faster with batching.

Latency

Protocol	Connect time (ms) LAN	WAN
TCP	0.2	105
Tcpcrypt cached	0.3	103
Tcpcrypt not cached	11.3	219
Tcpcrypt CMAC	11.4	320
Tcpcrypt PAKE	15.2	426
SSL cached	0.7	210
SSL not cached	11.6	321

tcpcrypt has lower latency than SSL.

Demo: Wireshark + TCP options

sudo ./launch_tcpcryptd.sh

http://tcpcrypt.org/fame.php

wireshark (passive eavesdropping)
ip.addr == 171.66.3.196

graceful fallback

without "sudo ./launch_tcpcryptd.sh"

cleartext GET request

Q: Why don’t people actually deploy tcpcrypt? Is it because people don’t care about security or people are happy with SSL’s performance?

ewm87: “people get very nervous about changing security protocols”

eamullen: “Maybe if it the implementation was verified, people would be more ready to adopt it.”

jtoman: “Sure you could implement these primitives using getsockopt, but good luck doing that in PHP running on apache (or even PHP running via FCGI).”

bornholt: “It's one of those clean sheet designs that wasn't viable when current systems were being designed but that makes perfect sense now (like, say, Rust).”

Q: Why don’t people actually deploy tcpcrypt? Is it because people don’t care about security or people are happy with SSL’s performance?

vsriniv2:

1. Deploying TCP options is impossible on today's internet.
2. tcpcrypt is susceptible to attacks that dramatically restrict its utility -- downgrade and MITM attacks are possible
3. When Http is treated as a transport (albeit a not-great one, with pipelining but not split transactions), encryption and ag,uthentication are better deployed at that layer.

billzorn:

“I just don't think people care enough about security to catalyze a major shift in something as massive and boring as infrastructure. I don't see how encrypting all of my traffic would help that much when there are still major security concerns about things like browsers and OSs.”

Q: Should we have encryption in transport layer in the first place?

lijl: “I still think encryption should be done in the application level. This is definitely more a religious question, but adding encryption to the transport layer just seems to break the abstraction provided by the OSI model. And to the ubiquitous problem, is it possible to simply add a layer in the OS network stack or even in glibc to encrypt all network messages?”

antoine: “Besides the religious reasons mentioned above, I would also like to mention a more practical point: updates.”

Q: Do you agree that it makes sense to separate confidentiality/integrity from authentication? Are the first two guarantees more fundamental than the third one? Do you ever only need the first two guarantees without the third one?

wysem: “it seems to make sense to build confidentially/integrity into the network stack itself (at TCP) instead of forcing this on application developers. It adds a level of protection for everyone and allows those requiring more to add at the application level.”

“Given the increasingly mobile nature of computing, are mobile processors implementations of hardware encryption/AES sufficiently efficient to promote use of this idea? “

naveenks: This should help me remain anonymous and be certain that my traffic can't be snooped by anyone or be tampered by anyone.

Q: Does it make sense to further separate confidentiality from integrity and push one of them further down the network stack?

wysem: “it seems to make sense to build confidentially/integrity into the network stack itself (at TCP) instead of forcing this on application developers. It adds a level of protection for everyone and allows those requiring more to add at the application level.”

“Given the increasingly mobile nature of computing, are mobile processors implementations of hardware encryption/AES sufficiently efficient to promote use of this idea? “

naveenks: “ I can imagine cases where I want just integrity but not confidentiality. “

Comments:

jrw12: “This paper' analysis of what's wrong with modern crypto on the web is right on.”

naveenk: “how easy would it be to build a Tor like system on top of tcpcrypt?”