tcpcrypt

Mark Handley

What would it take to encrypt all the traffic on the Internet, by default, all the time?

Crypto 101: Encryption without authentication is useless.

- Encryption without authentication is like meeting a stranger in a dark alley.
  - Whatever happens, there will be no witnesses.

tcpcrypt:
Opportunistic Encryption of TCP Flows

- Public key exchange in TCP handshake.
- Generate shared secret.
- Use shared secret to bootstrap encryption and MAC of TCP packets.
- Use shared secret to allow session rekeying, lightweight setup of additional sessions and session resumption from different IP addresses.

So, you like hanging about in dark alleys then?

- Did you close the curtains in your hotel room last night?

What use opportunistic encryption?

- Changes the balance of power.
  - Easy for a passive eavesdropper to listen to all of your traffic.
  - Active interception is a lot harder, and is inherently detectable.

So you support terrorists and child porn then?

- So you support identify theft?
- So you support phishing?
- So you support rate limiting of bittorrent traffic?
- So you support the great firewall of China?
- So you support government repression of freedom of speech in <*insert repressive regime of the moment*>?

What about lawful intercept?

- Whose laws?

Are we having fun yet?

What about lawful intercept?

- Opportunistic encryption prevents passive eavesdropping but is no obstacle to targetted active interception.
  - Can be man-in-the-middle.
  - Can simply downgrade to regular TCP.

OK, so much for the politics…

- What about the technical issues?

Architecture

- Why push a weak crypto solution?
  - Because it isn't weak.
  - It's just the building block upon which you build more powerful solutions.

Architecture

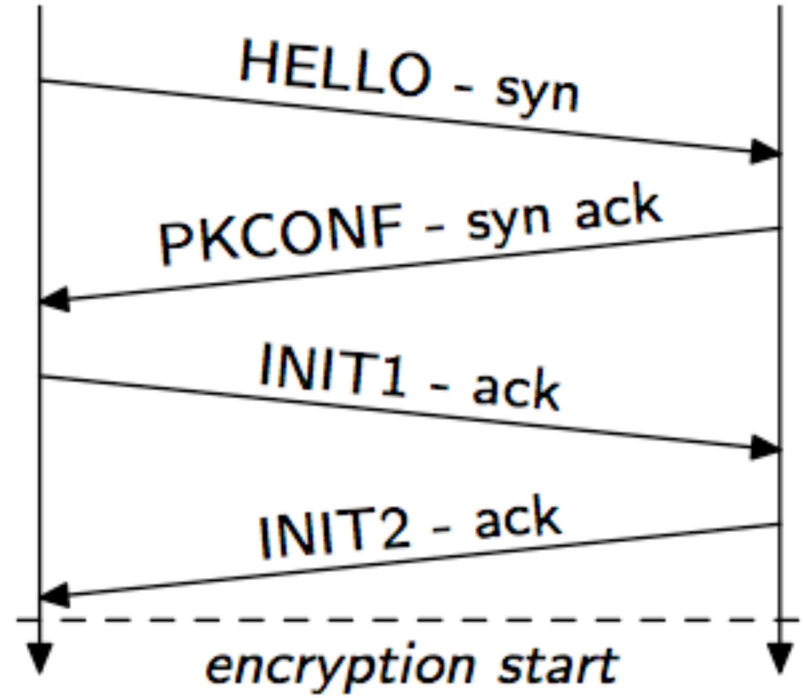
- Encryption is generic.
  - Don't need to know about the semantics of the data to keep it secret.

- Authentication is application specific.
  - Who do I trust?
  - Who is authenticating whom?
  - What identity am I authenticating?
  - How do I bootstrap identity?

Assertions

- With the right encryption building block, we can support a wide range of authentication schemes.
- We can make it go fast enough to be on by default.

Mechanism



In TCP handshake, negotiate tcpcrypt:

- C → S : HELLO
- S → C : PKCONF, pub-cipher-list
- C → S : INIT1, sym-cipher-list, NC, KC
- S → C : INIT2, sym-cipher, ENCRYPT(KC , NS)

Mechanism (2)

Generate shared secret:

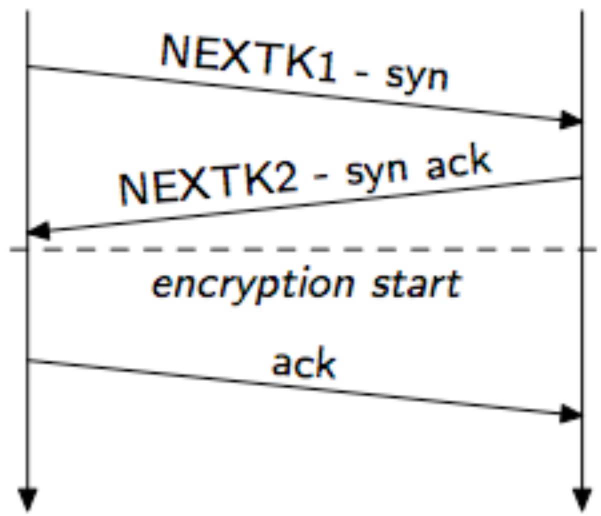
ss[0] ← HMAC (NS , {KC , NC , cipher-lists, sym-cipher})

From ss[i], use HMAC(ss[i], *x*) for various constants ***x*** to generate encryption and authentication keys for each direction.

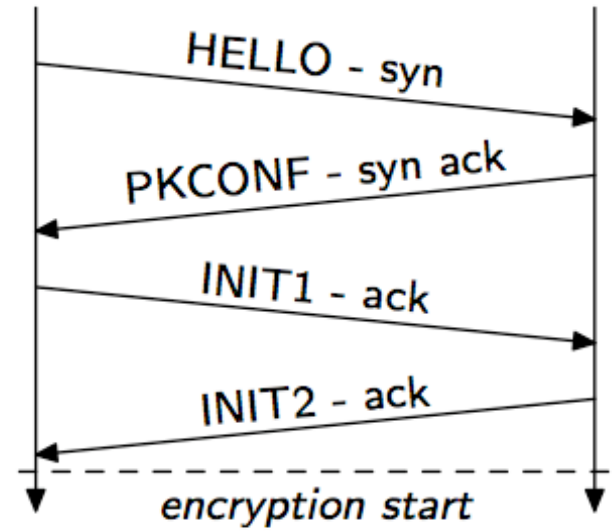
Note: KC is ephemeral: not stored to disk and regenerated frequently. Provides forward secrecy.

Mechanism (3)

- Subsequent connections can bootstrap using the shared secrets without doing public key operations:
  ss[i] ← HMAC(ss[i – 1], TAG_NEXT_KEY)



Embedding it in TCP



- HELLO and PKCONF fit in tcp optic SYN/ACK.
- INIT1 and INIT2 are too big for options.
  - Hijack the payload of first two data segments, as app can't have sent any data yet.
- Subsequent packets:
  - All include MAC option and payload is encrypted.

Authentication

tcpcrypt generates a session ID from crypto at both ends:

$$sid[i] \leftarrow HMAC(ss[i], TAG\_SESSION\_ID)$$

- Session ID is available by getsockopt.
- Guaranteed to be the same at both ends iff there is no man in the middle.

SSL-equivalent security

- Server can just sign the session ID using an SSL certificate.
  - Identical security to SSL, but also protects the TCP session from reset attacks, etc.

- Session ID is not a secret.
  - Can sign a batch of session IDs and send the batch and sig to many clients. Big speedup!

Mutual authentication using passwords

- h = H (salt, realm, password)
- C → S : HMAC(h, TAG_CLIENT || Session_ID)
- S → C : HMAC(h, TAG_SERVER || Session_ID)

- Server knows that client knows the password.
- Client knows that server also knew the password.
  - Proper mutual authentication.

- No more phishing attacks?
  - You know if you're talking directly to your bank or not because you know that they know your password.

Authentication

- Many different authentication schemes enabled by the session ID concept.

Performance

- Can be smart about using crypto.
  - Eg. single core can perform 12,243 encryptions/sec with a 2,048-bit RSA-3 key, but only 97 decryptions/sec

  Get the client to decrypt, server encrypts.

Implementation

- Andrea implemented tcpcrypt using a divert socket to a userland daemon.
  - Runs on Linux, FreeBSD, MacOS, etc.

- Not optimal performance (too many copies).
- No kernel changes needed.
- Can even run in a NAT!

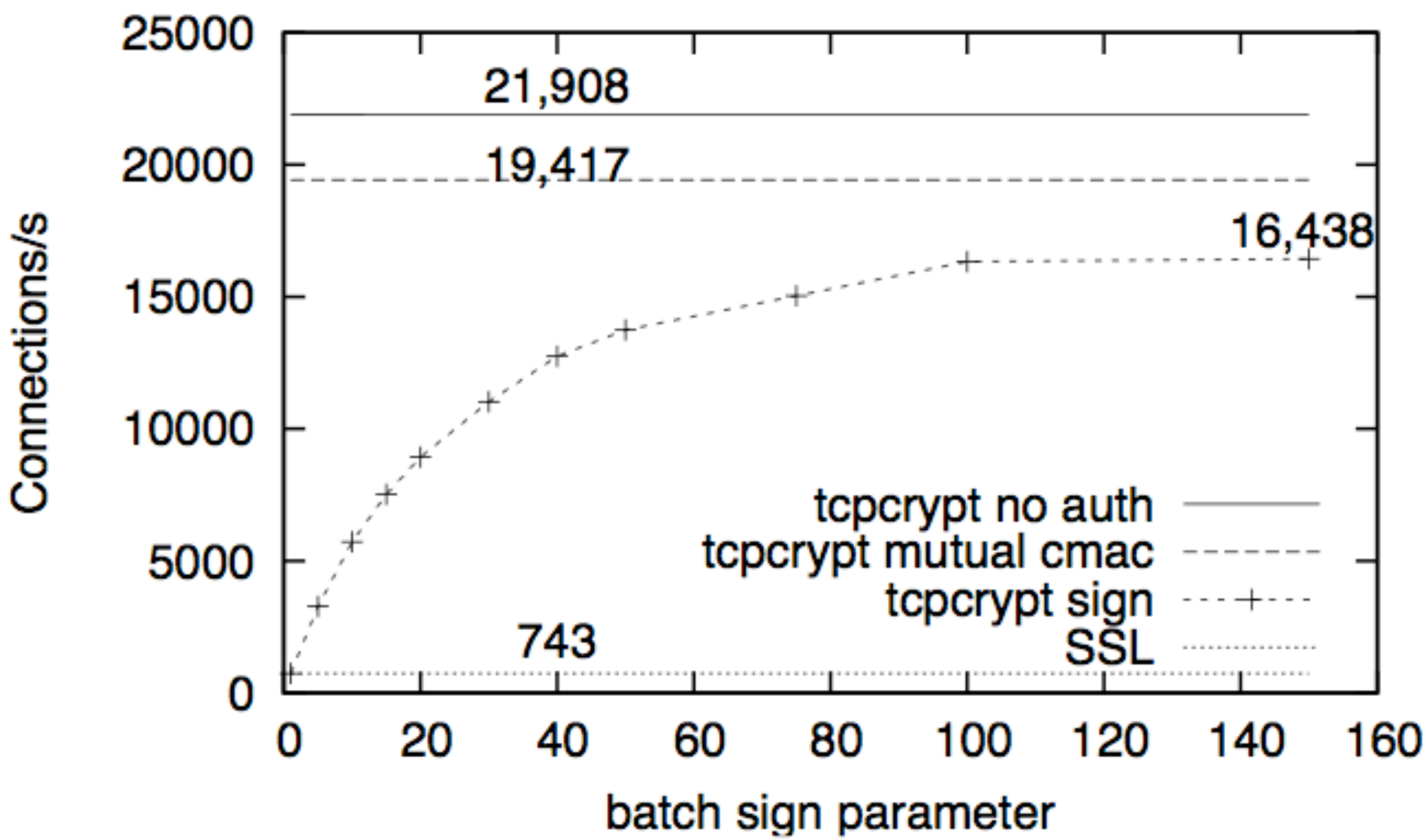
Performance (Connecton Setup)

Protocol	Connection rate (conn/s) Native	Divert
TCP server	98,434	61,515
tcpcrypt server (cached)		38,832
tcpcrypt server (uncached)		21,908
SSL server (cached)	39,785	27,348
SSL server (uncached)	754	743
tcpcrypt client (uncached)		749

Performance (Encryption)

Protocol	Transfer Throughput (Mb/s) Native	Divert
TCP	12,954	3,357
tcpcrypt AES-SHA1		1,752
tcpcrypt AES-UMAC		1,925
tcpcrypt RC4-UMAC		2,268
SSL AES-SHA1	3,692	1,939

Performance (with strong authentication)



Performance (Apache, static content)

Protocol	Apache, static content (req/s) Native	Divert
TCP	60,156	27,196
tcpcrypt (cached)		20,034
tcpcrypt (uncached)		14,215
SSL (cached)	19,787	12,063
SSL (uncached)	737	705

Performance (Apache, dynamic content)

- 10 connections per second
  - Wordpress sucked so badly, couldn’t see any different between plaintext, SSL and tcpcrypt.

MP-TCP (first connection to server)

- First subflow does handshake, bootstraps crypto.
  - Optionally, app-level auth.
  - Can do >>10,000 connections per second.
- Additional subflows use NEXTKEY.
  - No public key operations.
  - Crypto protects against hijacking.

MP-TCP (subsequent connections to server)

- First subflow uses NEXTKEY.
  - No public key operations.
- Subsequent subflows use NEXTKEY.

  No public key operations.

Summary

- tcpcrypt is not specific to MP-TCP.
  - Protects session integrity.
  - Provides auth framework.
  - Provides privacy against passive eavesdroppers.
  - Provides forward secrecy.

- tcpcrypt is well suited for MP-TCP
  - Protects subflow setup from hijacking attacks.
  - Hides content, so middleboxes don’t play guessing games with partial content.