tcpcrypt: the case for transport-level encryption

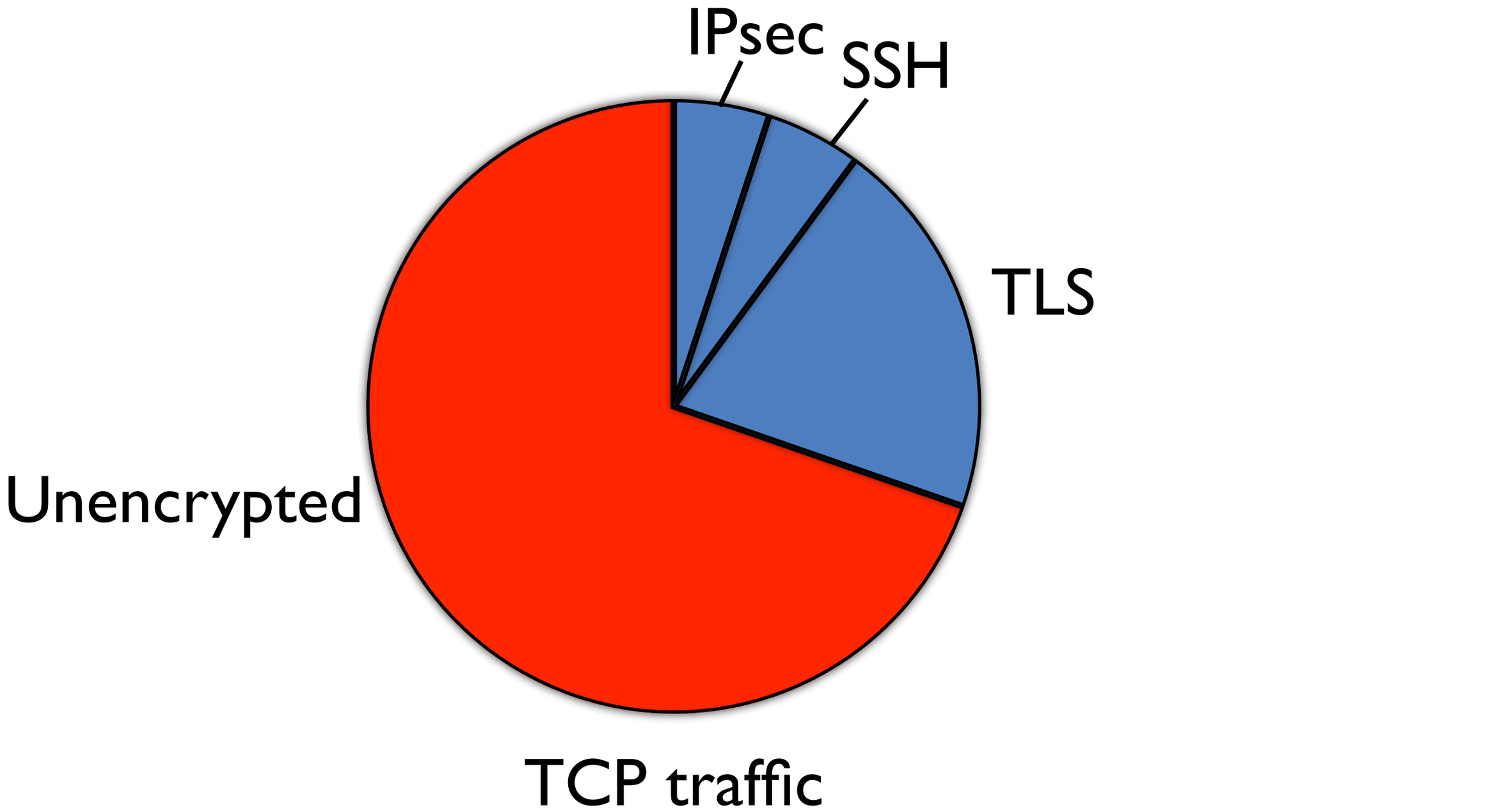
Andrea Bittau, Dan Boneh, Mike Hamburg,
Mark Handley, David Mazières, Quinn Slack

Stanford, UCL

Outline

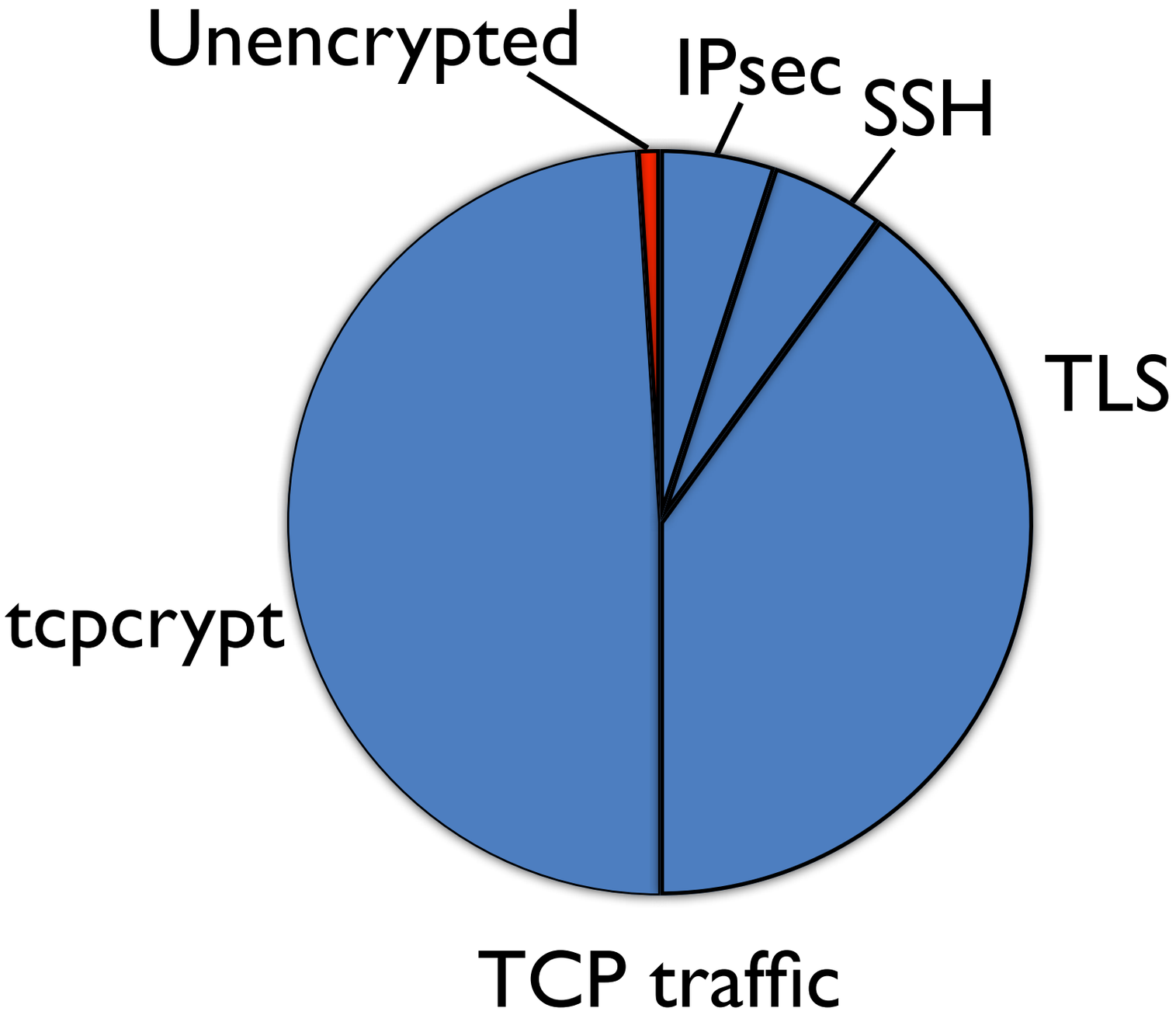
- Why the transport layer?
- Protocol overview
- Design choices
- Possible synergies

Today



Not drawn to scale

Goal: protect most traffic



Not drawn to scale

Step 1:
Maximize use of TLS

- Established solution for protecting communications.
- Depends on developers and administrators:
  - Requires changes to applications.
  - Requires changes to protocols.
  - Requires configuration (certificates).
- Can’t get to 99%

Step 2:
Reach remaining TCP traffic

- Need drop-in replacement for TCP:
  - No changes to applications.
    - No changes to application-level protocols.
    - No configuration.
- Support for incremental adoption.
- Compatibility with NATs.

Security requirements

- Forward secrecy regardless of configuration.
- Always protect against passive eavesdropping.
  - Pervasive monitoring by active attack must be detectable.
- Play well with application-level security.
  - Allow applications to prevent active attacks.
  - Be hard for application writers to misuse.
  - Make it easy to avoid double encryption.

Forward secrecy

- Definition: previously recorded communications cannot be decrypted by compromising one or both endpoints.
- Requirements:
  - No null or weak cipher options.
  - Randomness at both endpoints.
  - Short lived public keys.
- How to achieve for most TCP traffic?

Why the TCP layer?

- Ideally positioned to benefit applications that don't use TLS (and IPSec, SSH).
- Drop-in solution for unmodified applications.
- NAT friendly.
- Natural granularity for authentication.
- Conceptually encryption is as fundamental as error detection, reliability and congestion control.
  - Precedent for transport-layer crypto in TCP-AO.

tcpcrypt

- New TCP option for opportunistic encryption.
- Zero configuration, no changes to applications, simple, good performance.
- Allow end-point authentication.
  - Detect pervasive active attacks.
  - Possibility of integration with MPTCP, TCP Fast Open, DANE, ...

The authentication problem

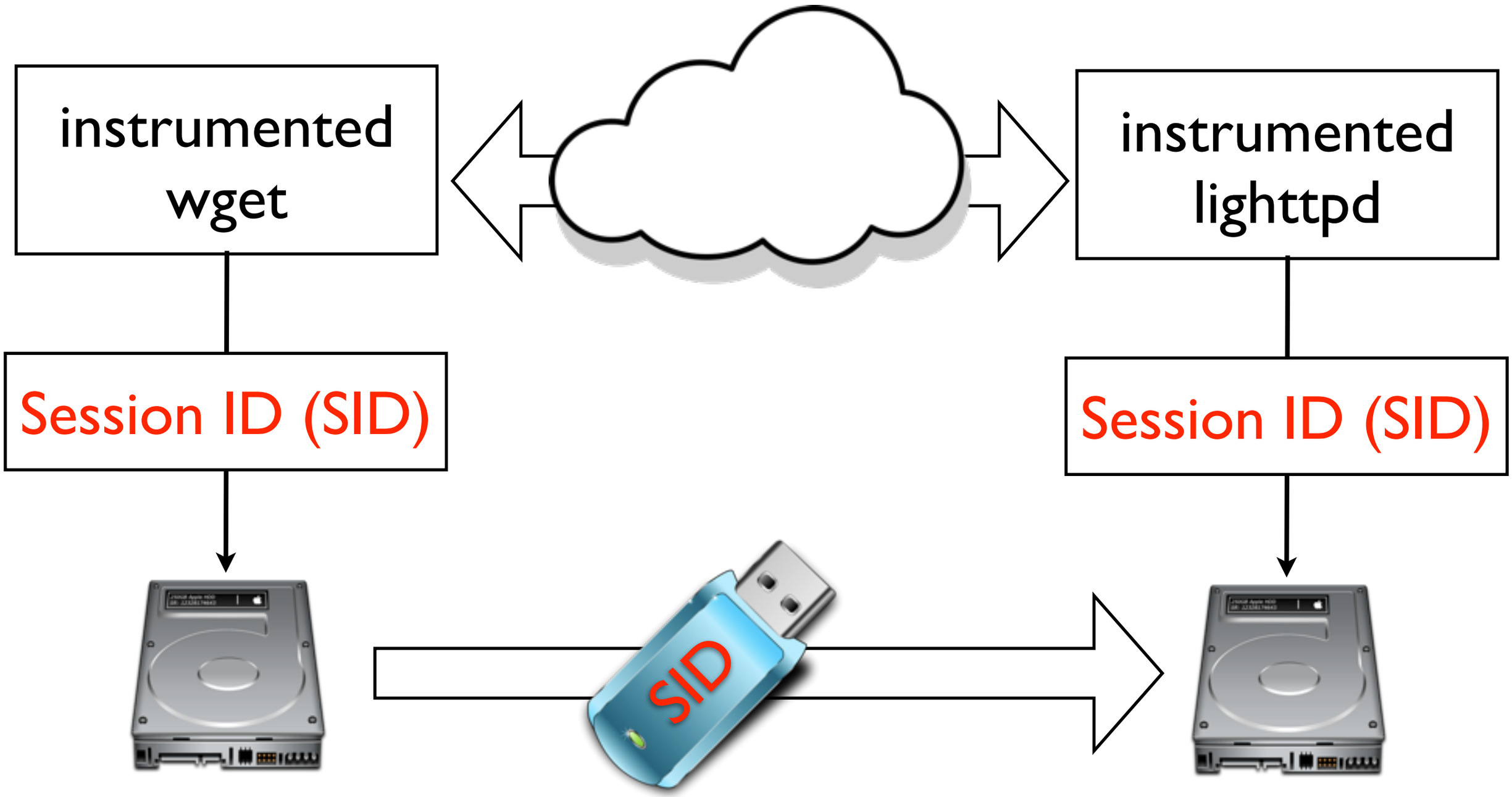
Session ID

- New getsockopt returns a connection's Session ID.
  - Returns error for non-tcpcrypt flows.
- Unique over all time with overwhelming probability.
  - Even if one end of connection is malicious.
- If equal on both ends, no man-in-the-middle.
- Without authentication, tcpcrypt secure against passive eavesdropping only.

Detecting pervasive monitoring

- Suppose tcpcrypt is widely deployed.
- Say 50% of unencrypted HTTP under active attack.
- Downgrade attacks easy to detect:
  - Session ID getsockopt returns error.
- What about man-in-the-middle?
  - Idea: instrument 100 http clients and servers to detect attacks.

Detecting pervasive monitoring



Session IDs are generic

- Enables experiments that detect widespread man-in-the-middle.
- Easy to integrate with application-level authentication. E.g., DANE, password, ...
- Can be repurposed to name connections in related technologies: MPTCP, TCP Fast Open.

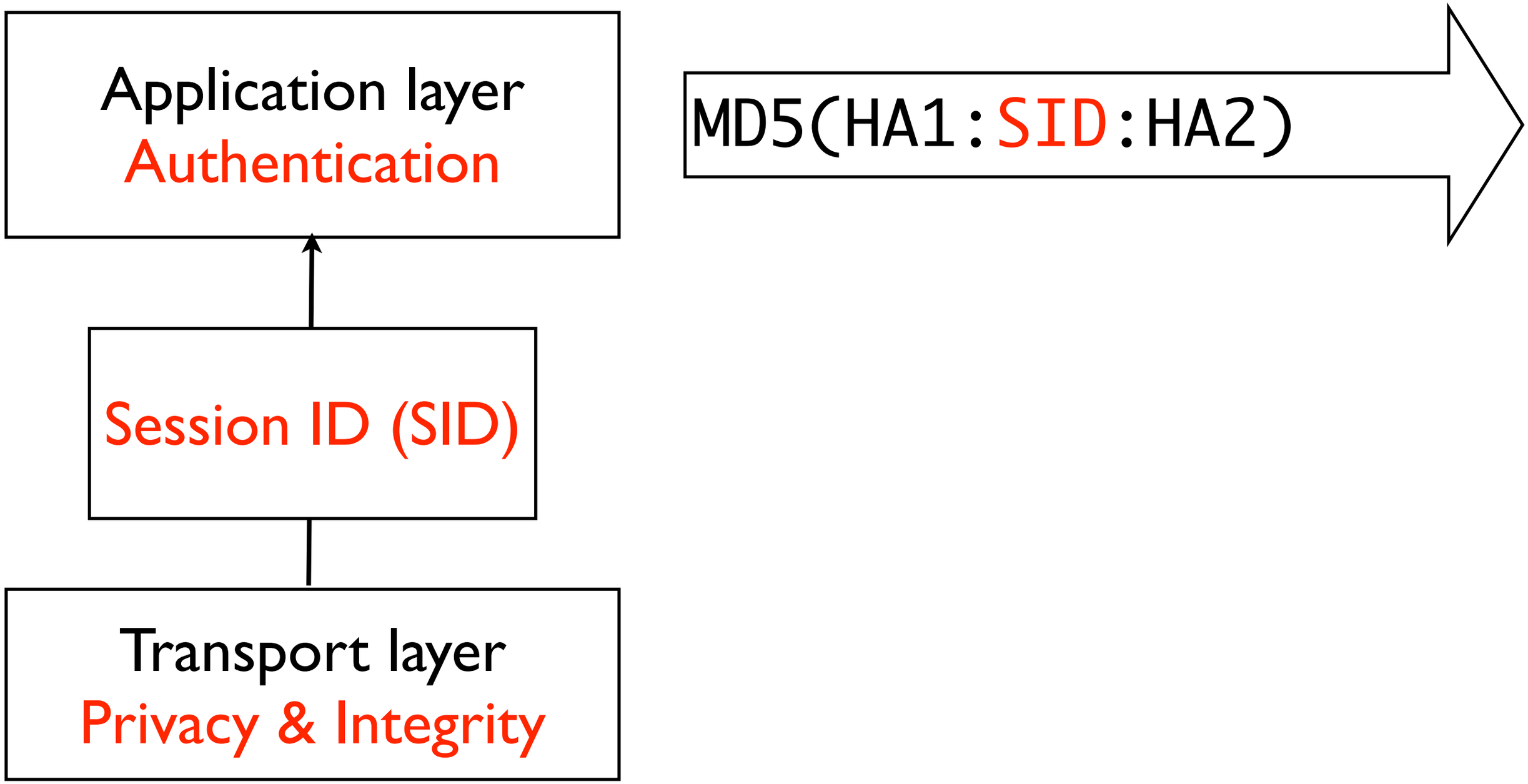
Integrating with application-level authentication

Application layer
Authentication

MD5(HA1:nonce:HA2)

Transport layer
Privacy & Integrity

Integrating with application-level authentication



Outline

- Why the transport layer?
- Protocol overview
- Design choices
- Possible synergies

tcpcrypt handshake

SYN

CRYPT<HELLO> option

SYN ACK

CRYPT<PKCONF>: public key ciphers, key sizes

ACK

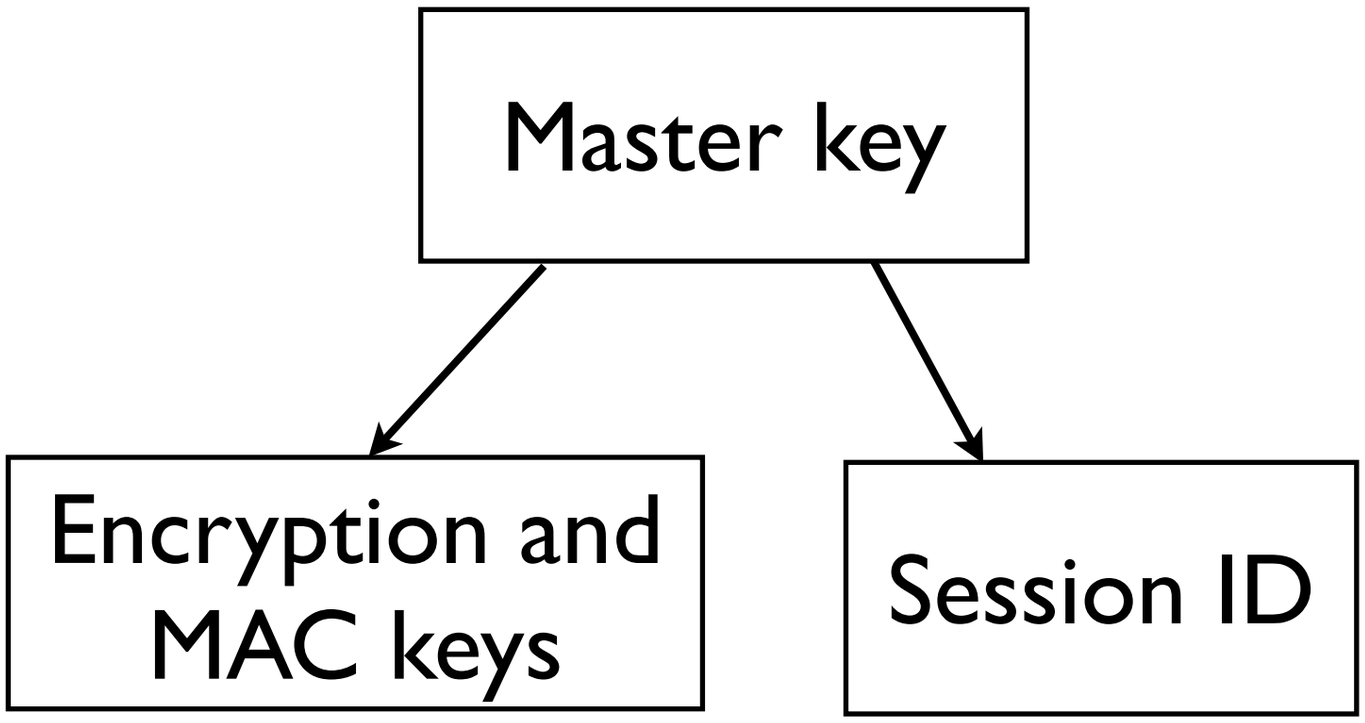
CRYPT<INIT1>: symmetric ciphers, MACs, nonce, public key

ACK

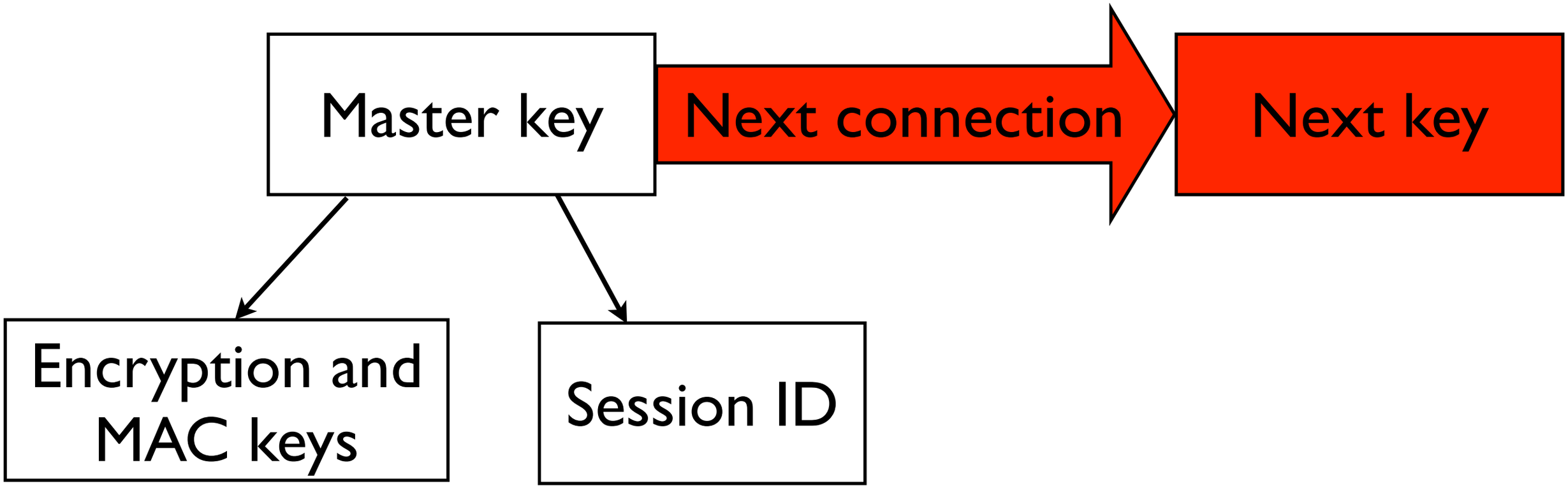
CRYPT<INIT2>: key material [RSA encrypted server+client nonce]
[DH: dh-output, server nonce]

INIT1/2 don't fit in SYN / ACK: sent as data invisible to apps.

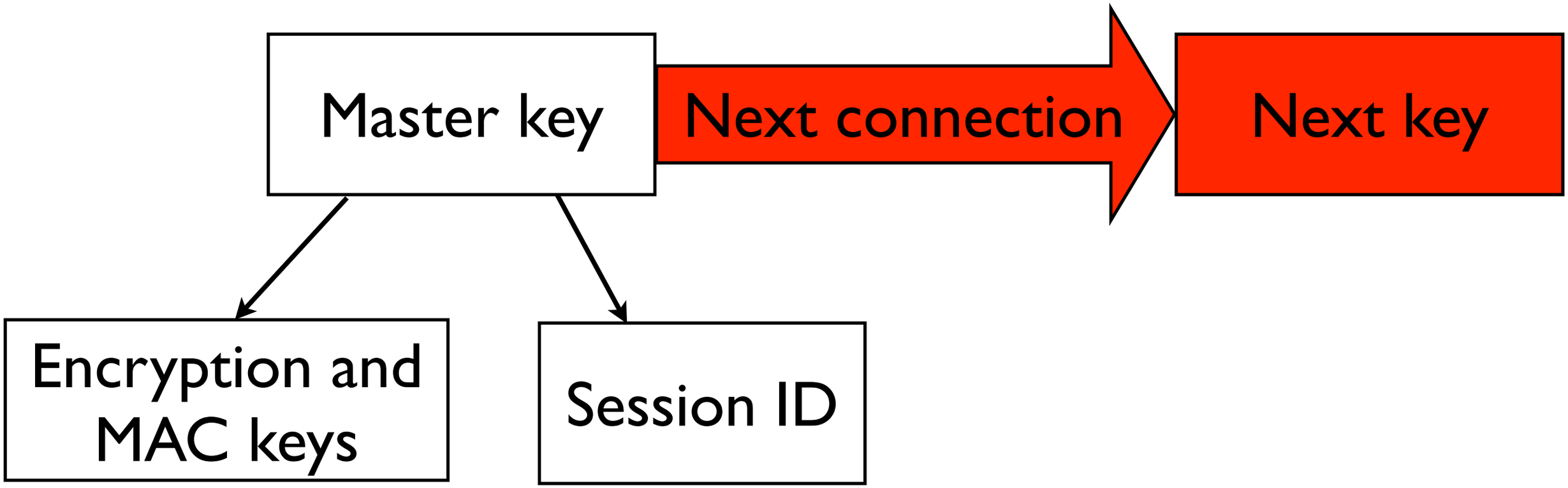
Session cached handshake



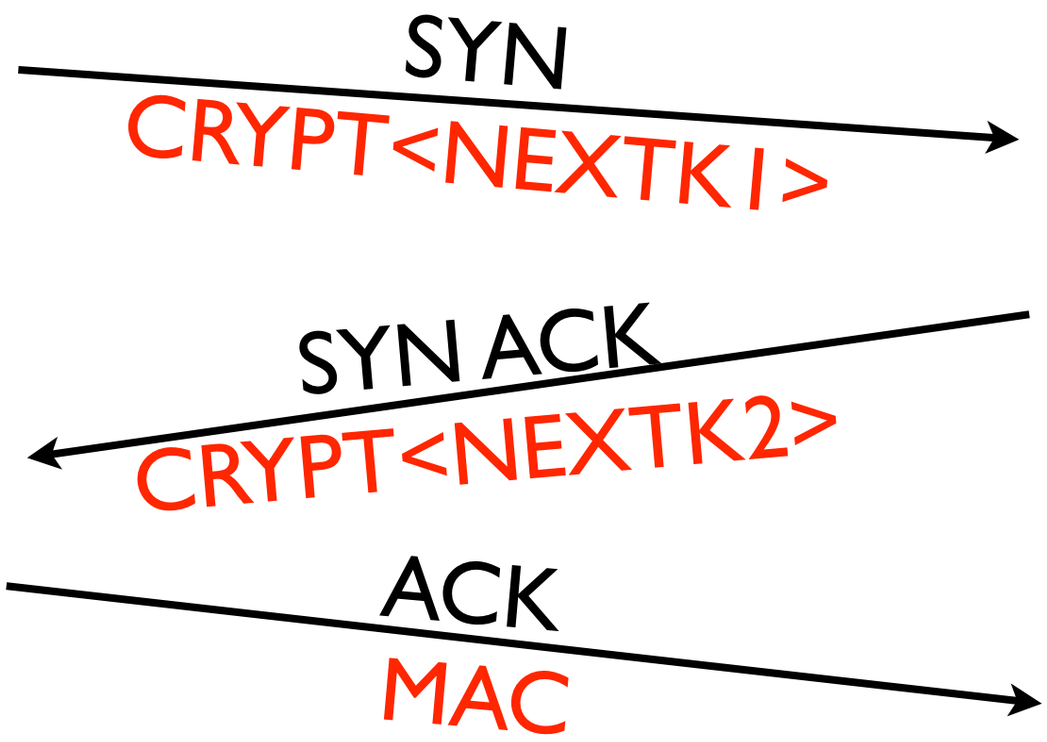
Session cached handshake



Session cached handshake



Low latency!



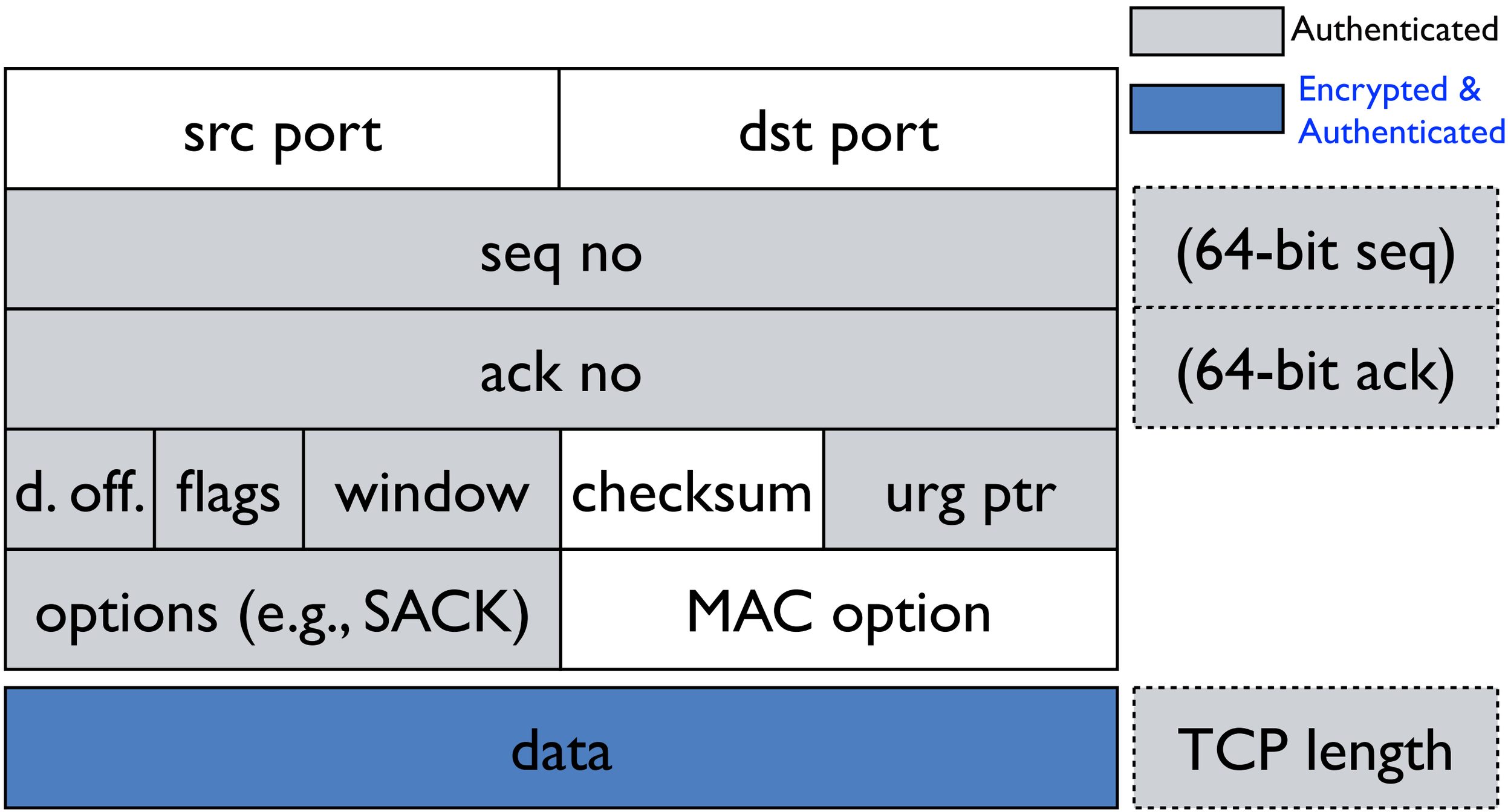
Backwards compatibility

- Fallback to TCP if:
  - One end point does not support tcpcrypt.
  - Middlebox strips CRYPT option.
- Different CRYPT suboptions signal application-level awareness of tcpcrypt out of band.
  - Specified using tcpcrypt mode socket option.
  - E.g., can be used to avoid double encryption.

tcpcrypt modes

1. Disabled.
2. Enabled.
3. Application aware.
   - Signals application support via getsockopt on peer.
   - May be useful for DANE or future protocols.
4. Application mandatory.
   - Like 3, but disables tcpcrypt if peer in mode 1 or 2.

Authenticated encryption (Encryption + MAC)



Authenticated encryption requirements

- Ciphertext for a particular byte position must never change, even if re-encryption occurs after coalescing and retransmission.
- Authentication must occur only on fields not modified by middleboxes.
- Updating the authentication tag must be cheap when only ACK numbers change.
  - Compute two MACs: one for the ACK number, and one for the rest. XOR the two.

Outline

- Why the transport layer?
- Protocol overview
- Design choices
- Possible synergies

INIT1/INIT2 in data

- Problem: can't fit key exchange into TCP option space (40 bytes).
- Encode INIT1 and INIT2 options in TCP payload invisible to applications, starting on third leg of handshake.
  - This never again happens throughout connection. Only at the very start.

MAC as TCP option

- Con: incompatible with TSO, middleboxes that coalesce packets.
- Alternative: MAC in data.
  - Would require retransmitting pure ACKs.
  - Can't check MAC of packets that arrive out of order.

Outline

- Why the transport layer?
- Protocol overview
- Design choices
- Possible synergies

Use with DANE

- DANE stores server certificate in DNS (uses DNSSEC).
- Can be used for server authentication: sign session ID with DANE certificate.
- Requires application changes,
  - But out-of-band application-aware signal can maintain backwards compatibility.

Other possible synergies

- Multipath TCP: need to ensure that multiple connections belong to same user.
  - Integrates well with tcpcrypt's session caching.
- Fast open TCP: need to ensure that future connections come from past users.
- In both cases: need proof that endpoints previously spoke. tcpcrypt can provide that.

Open Issues

- TCP Segmentation Offloading (TSO)
- Size of MAC option (e.g., 10 bytes).
  Especially when used with Multipath TCP.
- Reduce protocol complexity: rekey?
- Middleboxes: firewalls drop non-HTTP traffic on port 80?

Conclusion

- Large scale interception poses threat to all traffic.
- Tackle problem from both ends:
  - Increase TLS adoption wherever possible.
  - Slip tcpcrypt under applications that don't adapt.
- Can detect/prevent active attacks using Session ID.
- Out-of-band application-aware signaling may allow backwards compatible use of Session ID in future.

http://tcpcrypt.org

Backup

- Applications often mess up:
  - E.g., poor random seeds, failing to check replay, writing session keys to swap, sign message with insufficient context, ...

Implementation benefit: transport layer often in kernel

- Fewer implementations, easier to get right.
- Memory protection against application bugs.
- Keys never leave kernel.
- Tighter integration with sources of entropy.
- Prevents developers from tweaking internals.

tcpcrypt makes security pitfalls less likely

- Because application programmers do not modify the transport layer:
  - Applications can't disable forward secrecy.
  - Applications can't use inadequate entropy.
  - Applications can't leak session keys.
- By putting Session IDs into authentication:
  - No need to check for replay.
  - Signed messages cannot be taken out of context.

Use IPSec everywhere?

- Drop-in solution for unmodified applications.
- Hard to configure.
- Hard to make work with NAT.
- Hard to tie in with application-level authentication.
- Less widely used than TLS.

Multipath TCP integration

- Connection setup:
  - MP_CAPABLE transfers keys to generate connection tokens. Use tcpcrypt as a security algorithm and use Session ID as token.
- Adding flows:
  - MP_JOIN uses token to signal connection ID. Use tcpcrypt's Session ID and REKEY key stream.
- Steady-state:
  - DSS option up to 28 bytes. MAC option 11 bytes. 1 byte to spare! 32-bit seq nos make DSS 20 bytes, and DSS checksum not necessary with MAC (freeing 2 bytes).

TCP fast open integration

- Connection setup:
  - Server sends fast open cookie to client. Use tcpcrypt Session ID instead.
- Reconnect:
  - Client includes cookie in SYN, along with data. Use tcpcrypt Session ID as fast open cookie, and include MAC for data.