Living with postquantum cryptography

David McGrew, PhD
Cisco Fellow

April 2, 2015

Research into PQC sponsored (in part) by Cisco

- Biasi, Barreto, Misoczki, Ruggiero, *Scaling efficient code-based cryptosystems for embedded platforms,* 2012
- Bernstein, Lange, Peters, *Smaller decoding exponents: ball-collision decoding*, CRYPTO 2011
- Bernstein, Lange, Peters, *Wild McEliece Incognito*, PQC 2011
- Bernstein, *Grover vs. McEliece*, PQC 2010
- Burleson, Paar, Heyse, *Alternative Public-Key Algorithms for High-Performance Network Security*, 2011

Approach

1. Prepare for threat of practical quantum computer
2. Embrace well-known postquantum-secure algorithms
   - Well established security is paramount
3. Use systems engineering to mitigate performance issues

Approach

1. Prepare for threat of practical quantum computer
2. Embrace well-known postquantum-secure algorithms
   - Well established security is paramount
   - No Quantum Cryptography
3. Use systems engineering to mitigate performance issues

Identify opportunities and challenges, not detailed proposals

Cryptography

- Hash Based Signatures (HBS)
  - SHA-256
- Code Based Encryption (CBE)
  - McEliece/Neiderreiter encryption
    - 800KB public keys, but fast encryption/decryption
- Symmetric cryptography
  - AES, SHA-2, SHA-3

Applications of ‘systems’ approach

- HBS for authentication
- Minimize use of public key cryptography
- Optimize transmission and storage of large public keys
- Symmetric TTP key establishment

Quantum Key Distribution Is Not Needed

Minimal computational assumptions	Yes
Side channel resistance	No
Keys can be public	No
Minimal entropy requirements	No
Any device	No
High data rates	No
No range limitations	No
Point to multipoint	No
Any network, including wireless	No
Can be implemented in software	No
Simple	No

Hash Based Signatures

Hash Based Signatures

- 128-bit security level
  - 16*(265 + 20) = 1392 bytes, Key Gen time = 0.4ms * 2^20 = 7m
  - 16*(34+20) = 864 bytes, Key Gen time = 2.5ms * 2^20 = 45 m
  - Multilevel schemes improve these numbers
- Stateful signing
- Good security
- Feasible and useful

Minimize use of public key cryptography

Cryptographic services used in SSL/TLS

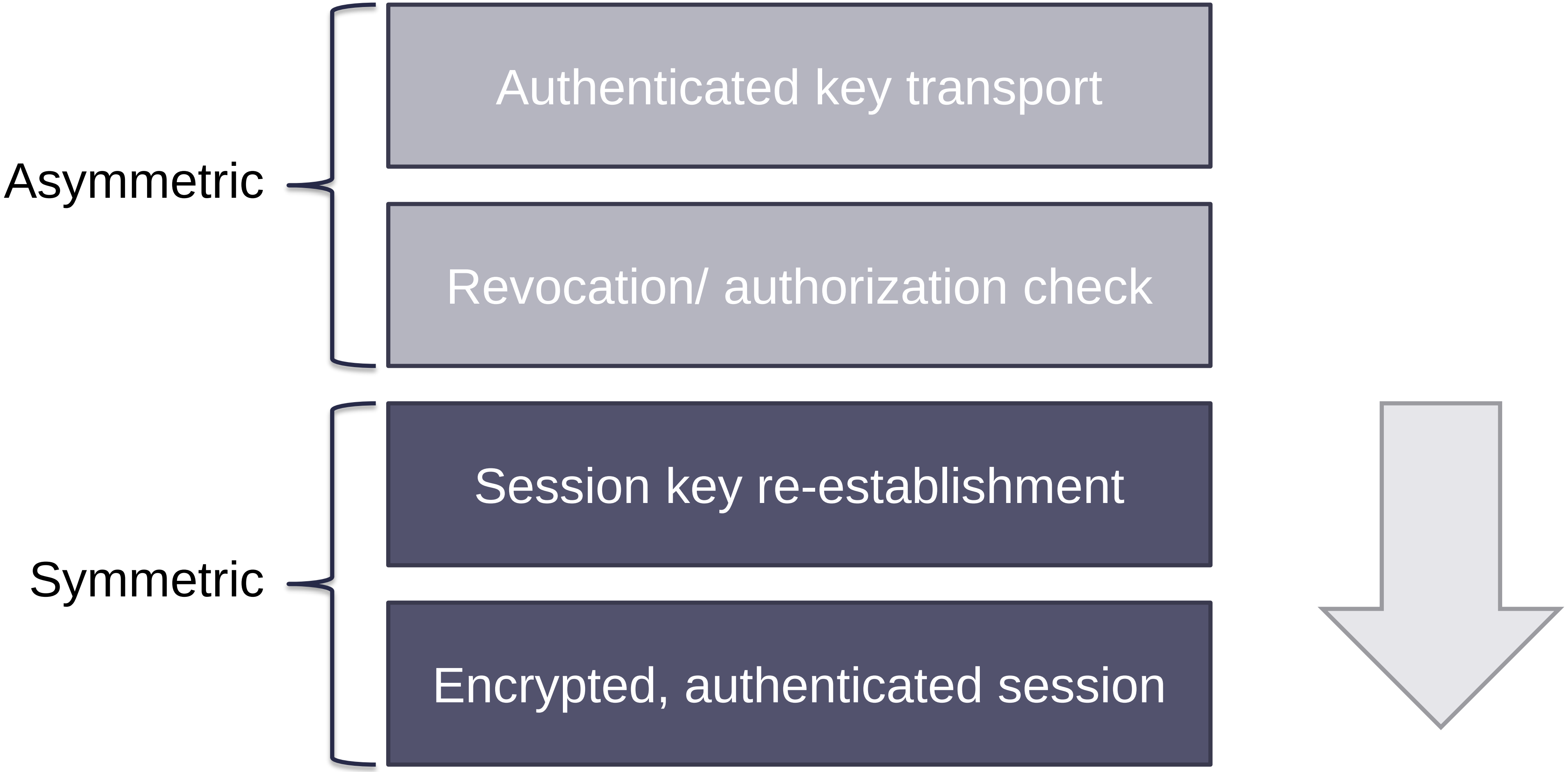
Service	Algorithm
End-entity authentication	Digital signatures PKC decryption MAC
Session secret establishment	DH PKC encryption Symmetric TTP
Session authenticated encryption	AEAD MAC encryption

SSL/TLS session establishment

Asymmetric
- Authenticated key transport
- Revocation/ authorization check

Symmetric
- Session key establishment
- Encrypted, authenticated session

SSL/TLS session establishment – session resumption



SSL/TLS long-lived sessions & session resumption

Asymmetric

- Authenticated key transport
- Revocation/ authorization check

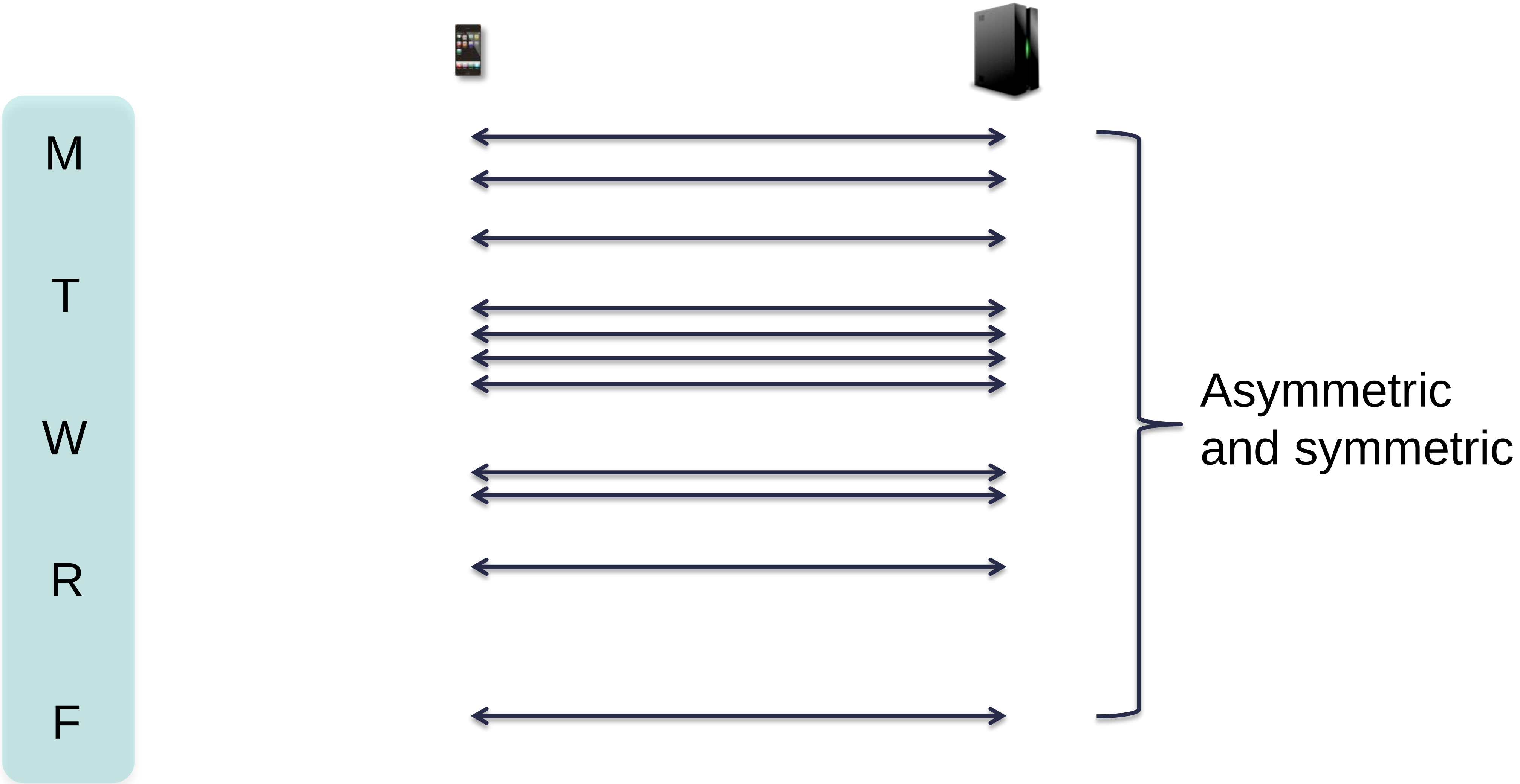
Symmetric

- Session key (re)establishment
- Encrypted, authenticated session

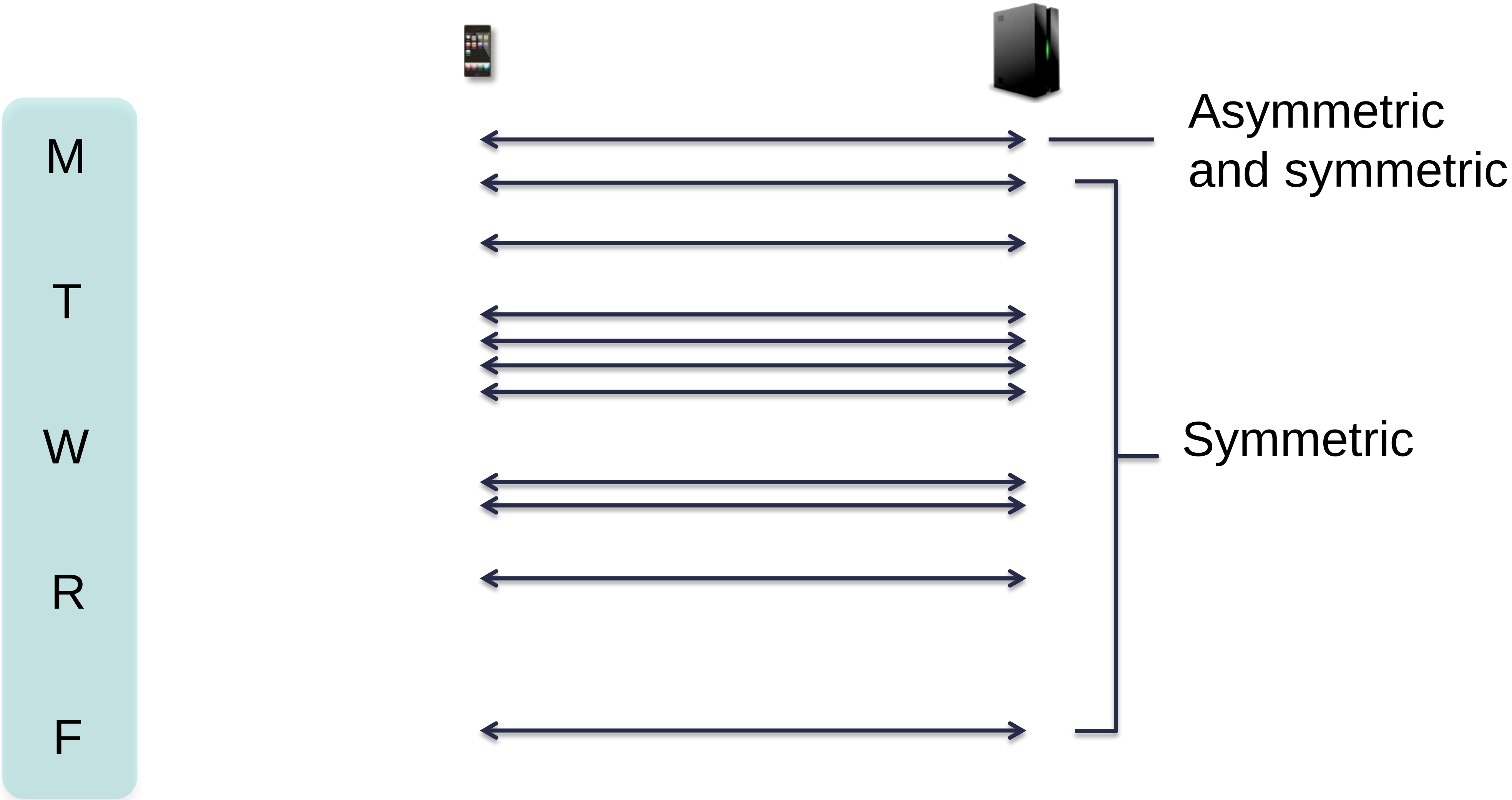
Once per peer

Once per session

TLS



TLS with Session Resumption



Issue: per-peer state

- State must be stored for each peer
  - Problematic for small devices
  - Problematic in web model
- Solution: state avoidance through encryption with local key
  - Enables server to maintain shared secret with N devices with $O(1)$ state
- RFC 5077, *TLS Session Resumption w/o Server-Side State*
  - ~ 64 bytes of state

Issues with long-lived sessions and session resumption

- Revocation check needed
  - Should use symmetric cryptography
  - Could be external to TLS
- Forward security is desirable
  - Could be achieved through use of PRF key updating function

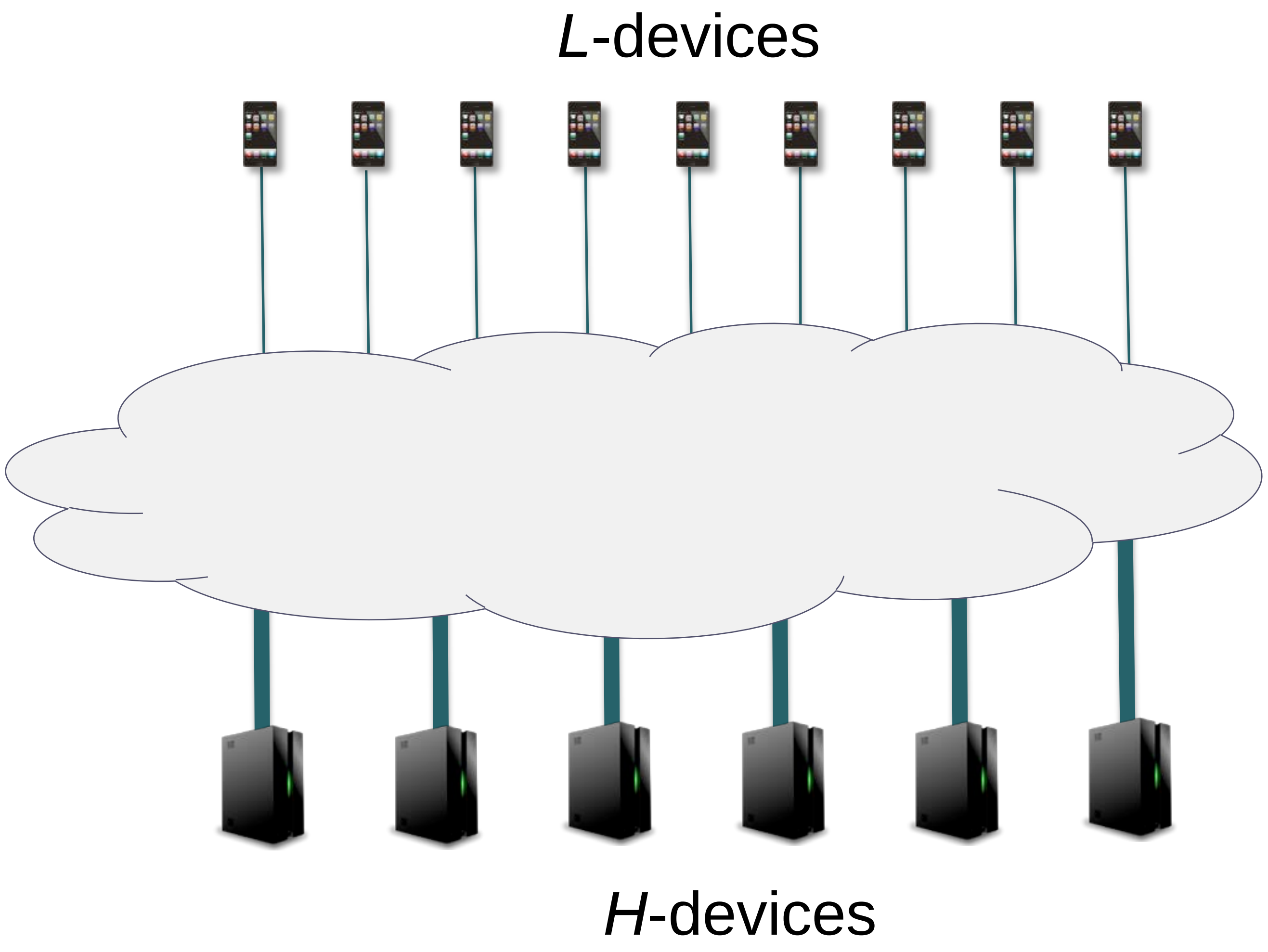
Optimize transmission and storage

Optimize transmission and storage

Low bandwidth (Mb/s)

High bandwidth (Gb/s)

High bandwidth (Gb/s)



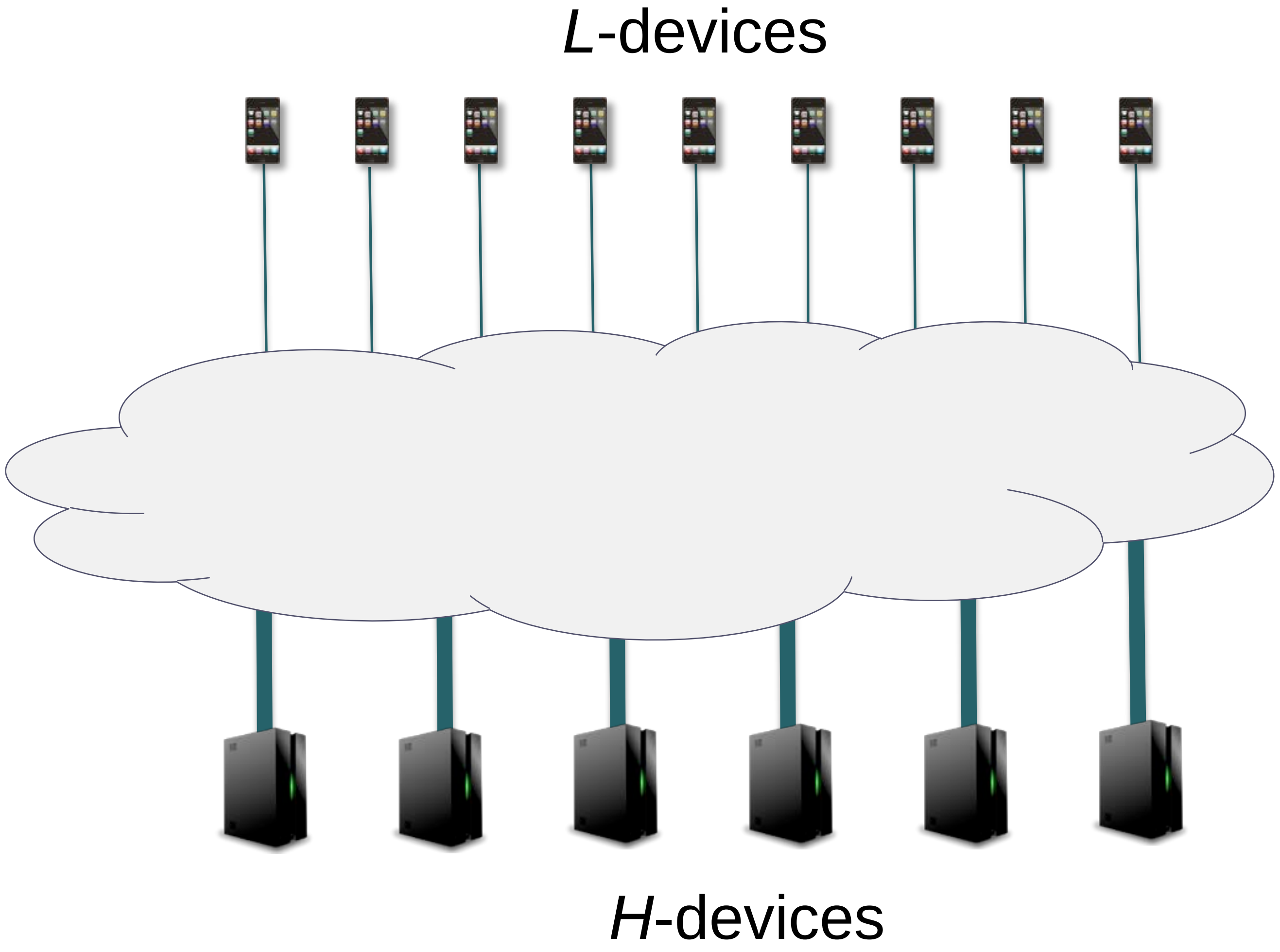
Time to send 800KB key

1 Mb/s 6.25s

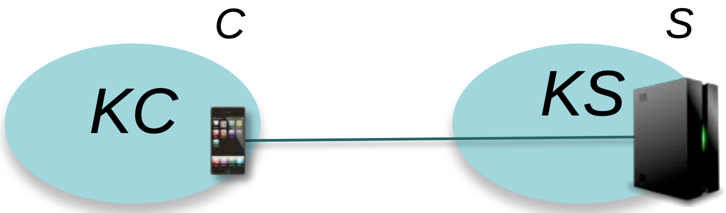
40 Gb/s

0.00015s

40 Gb/s



Using large public keys in TLS



$N_C \longrightarrow$

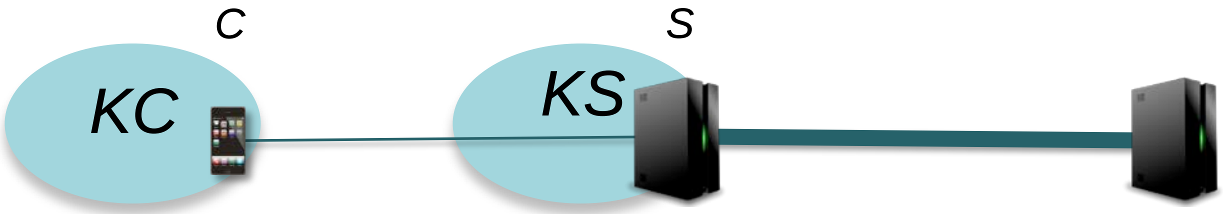
$N_S, KS \longleftarrow$

$E_{KS}(PMK), \mathrm{Sig}_{KC}(M_1), \{M_2\}_K \longrightarrow$

$\{M_3\}_K \longleftarrow$

Simplified TLS – Protocol 4.24, Boyd and Mathuria, *PFAKM*

Using large public keys in TLS



N_C →

N_S, KS ←

Revocation Check ↔

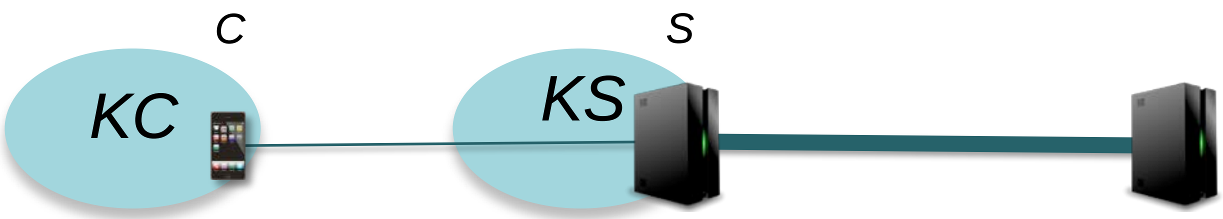
$E_{KS}(PMK)$, $\mathrm{Sig}_{KC}(M_1)$, $\{M_2\}_K$ →

$\{M_3\}_K$ ←

Simplified TLS – Protocol 4.24, Boyd and Mathuria, *PFAKM*

Using large public keys in TLS

Large key, slow link!



N_C

N_S, KS

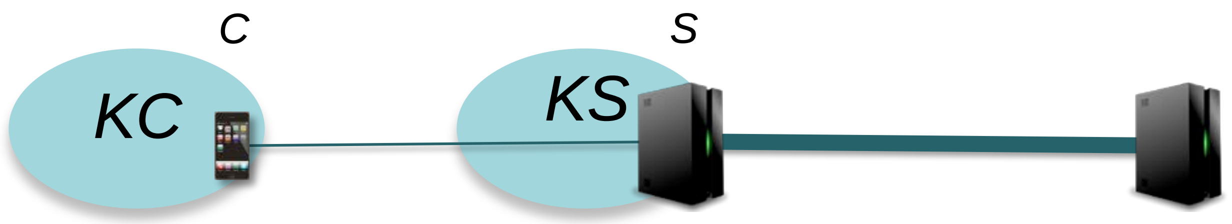
Revocation Check

$E_{KS}(PMK)$, $\mathrm{Sig}_{KC}(M_1)$, $\{M_2\}_K$

$\{M_3\}_K$

Simplified TLS – Protocol 4.24, Boyd and Mathuria, *PFAKM*

Using large public keys in ‘reversed’ TLS



R →

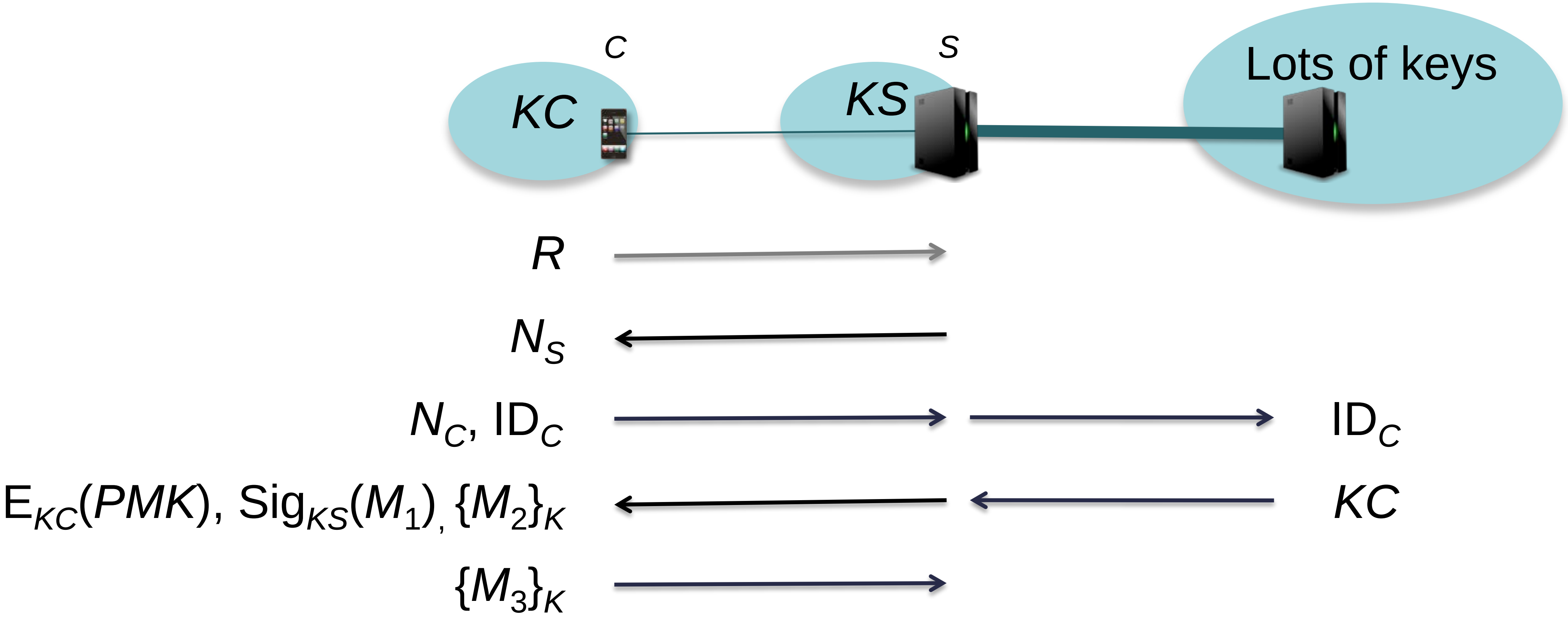
N_S ←

N_C, KC →

$E_{KC}(PMK)$, $Sig_{KS}(M_1)$, $\{M_2\}_K$ ←

$\{M_3\}_K$ →

Using large public keys in ‘reversed’ TLS

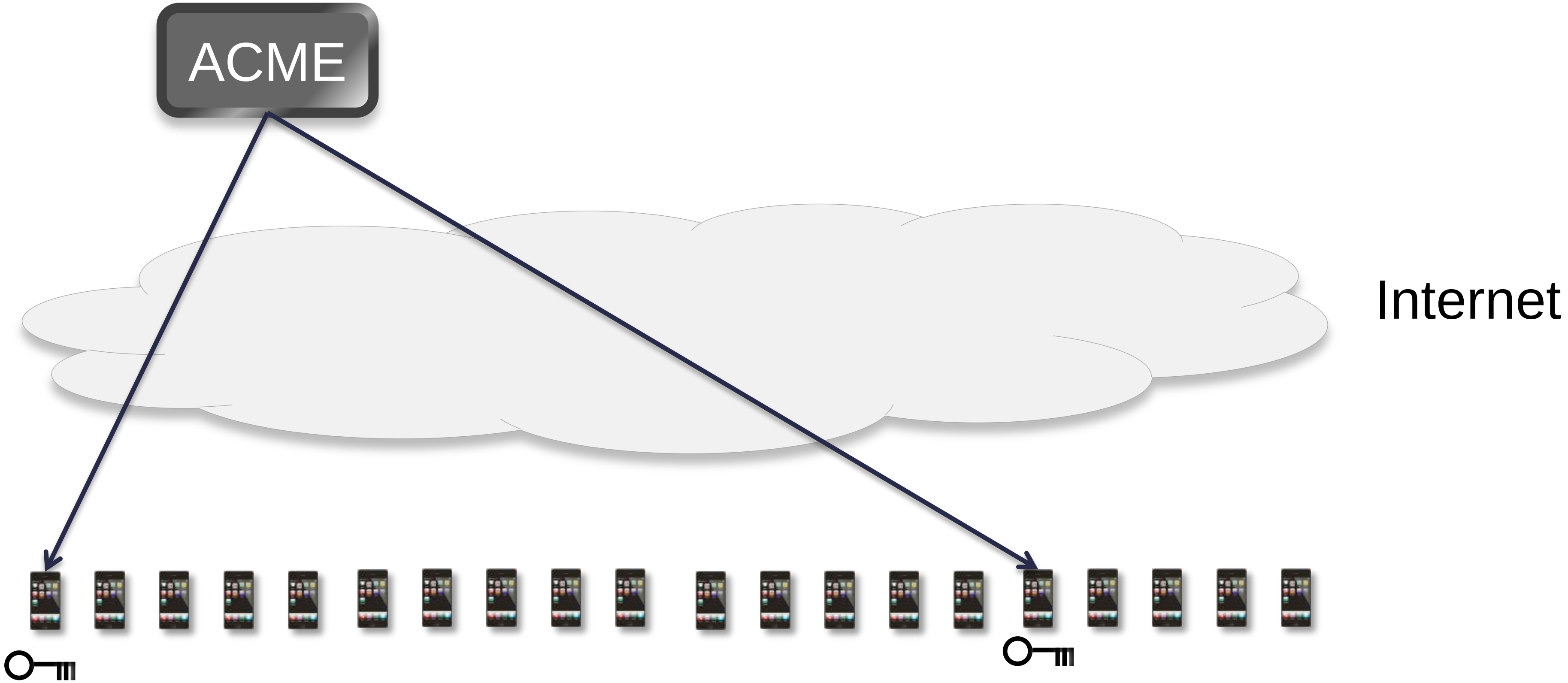


What did we achieve?

- Avoid transmitting large public keys across slow links
- Avoid storing large public keys on endpoints
- Leverage public cloud
  - Storing public keys
  - Revocation service

Symmetric TTP for encryption

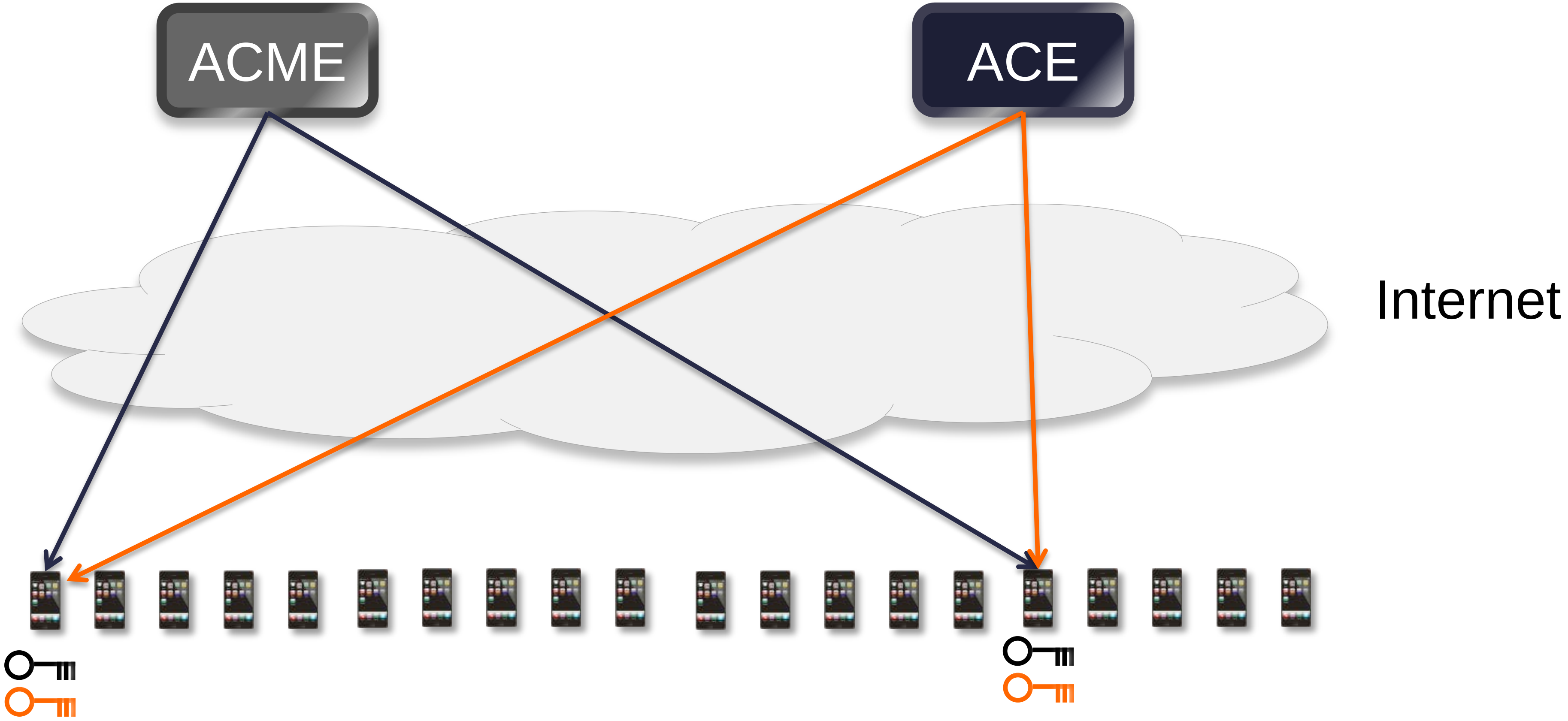
Trusted Third Party Key Establishment



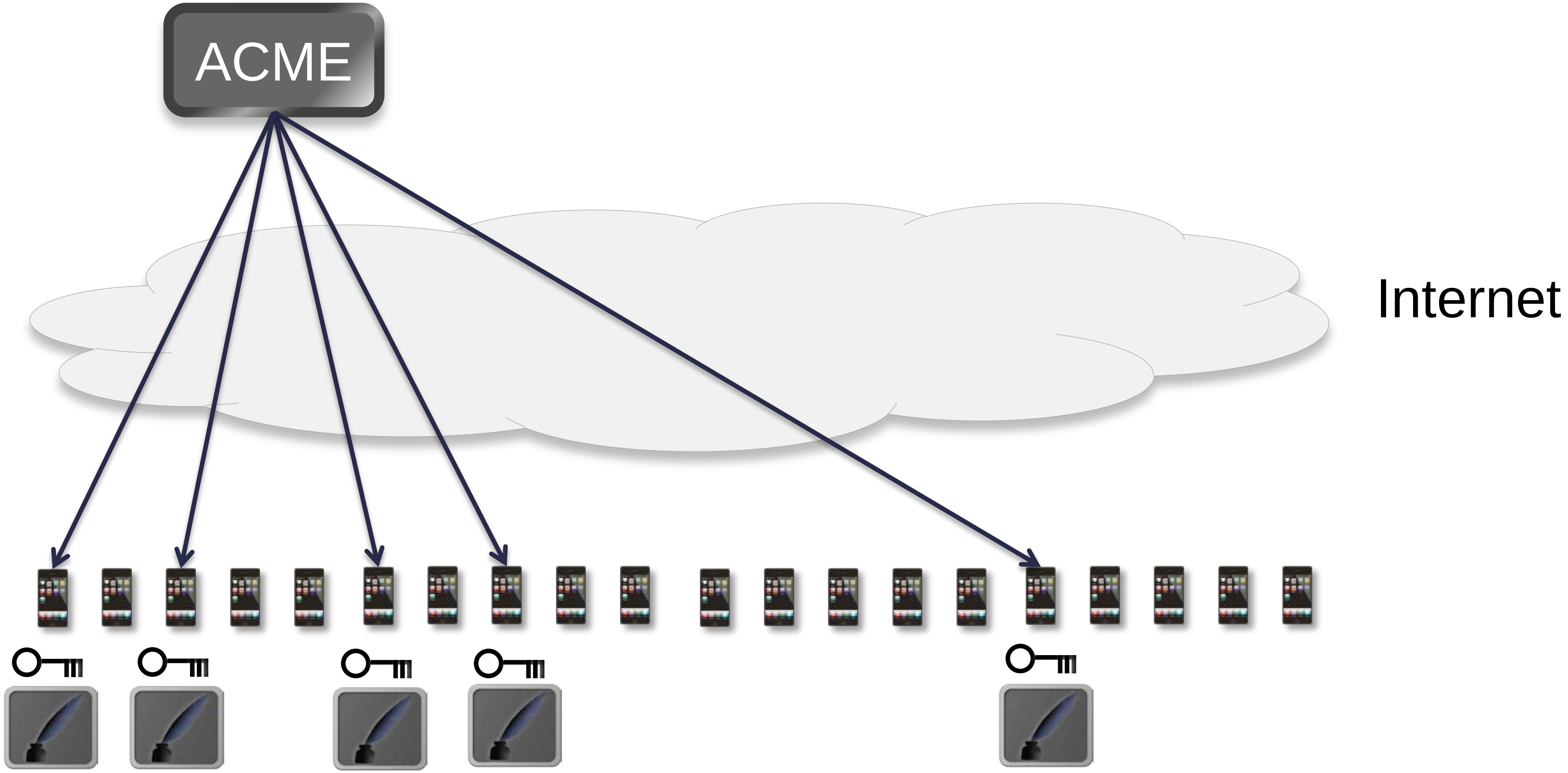
Trusted Third Party key management

- Easily postquantum secure
- Can use standards like krb5
- Can use server state avoidance to minimize storage cost

Threshold Trusted Third Party Key Establishment



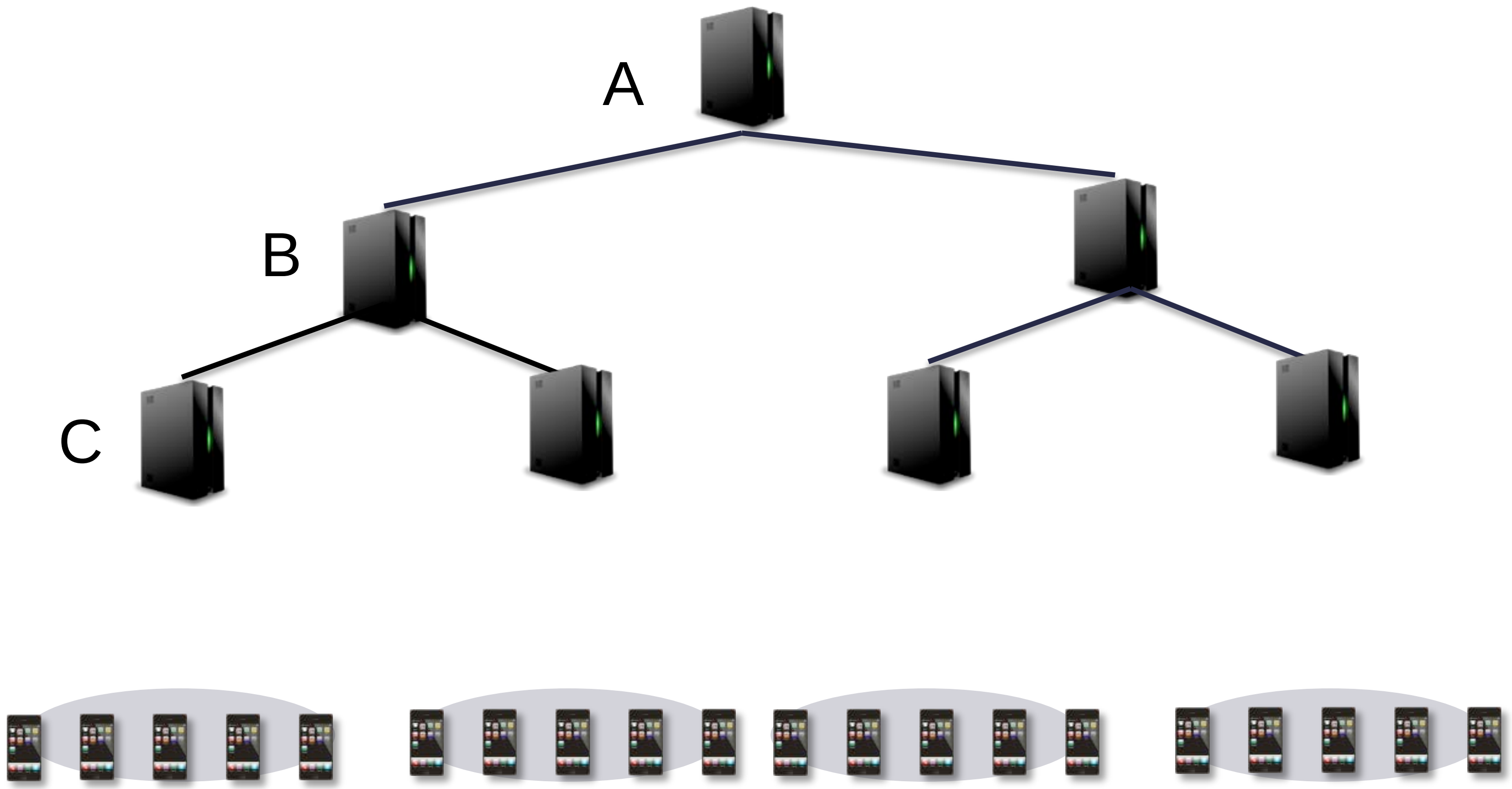
Group Keys for Encryption with Hash-based signatures



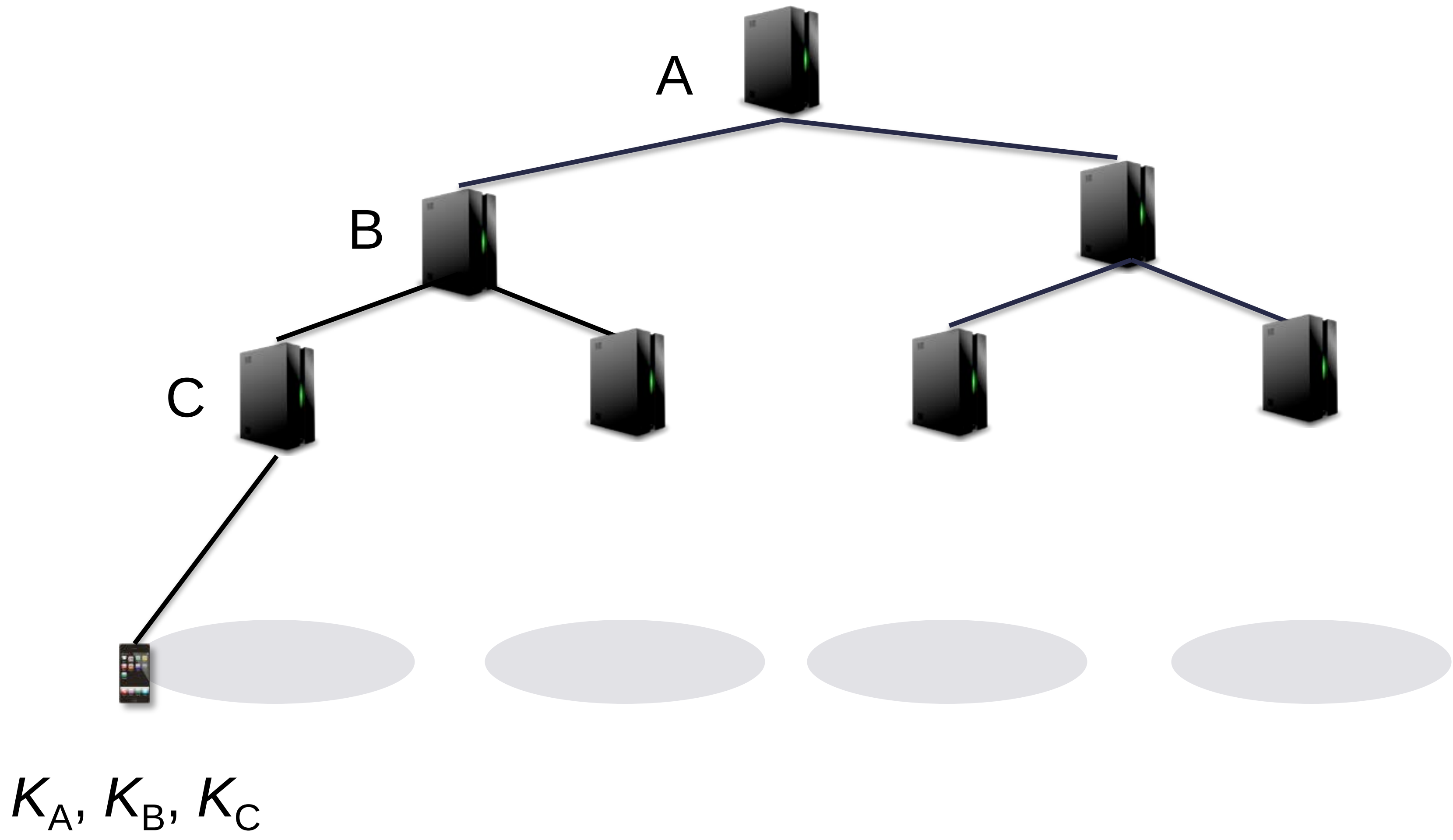
Trusted Third Party key management - issues

- TTP is high-risk target
  - Could use key sharing / threshold to mitigate risk
- Scalability
  - State avoidance
  - Hierarchical TTP

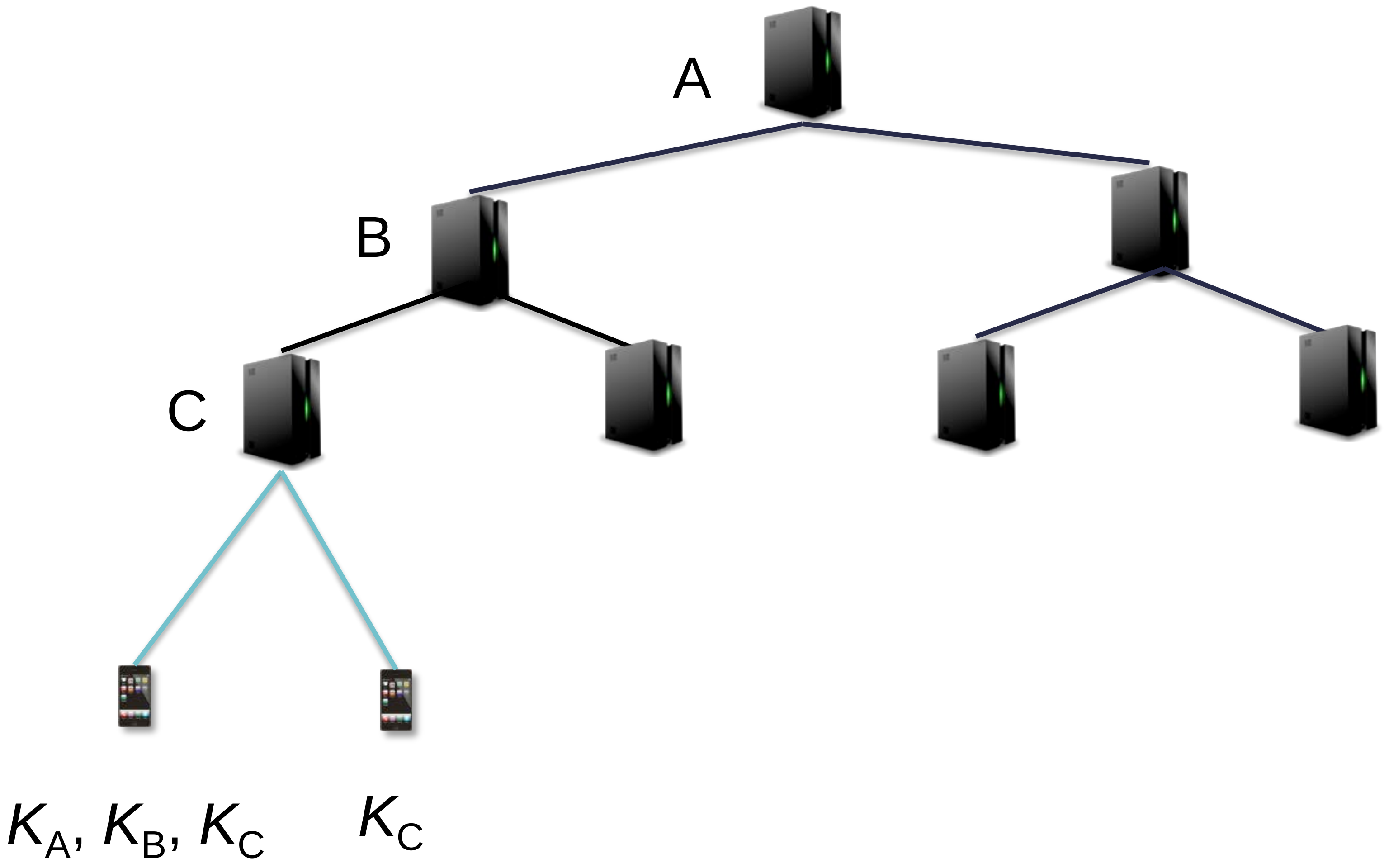
Hierarchical TTP



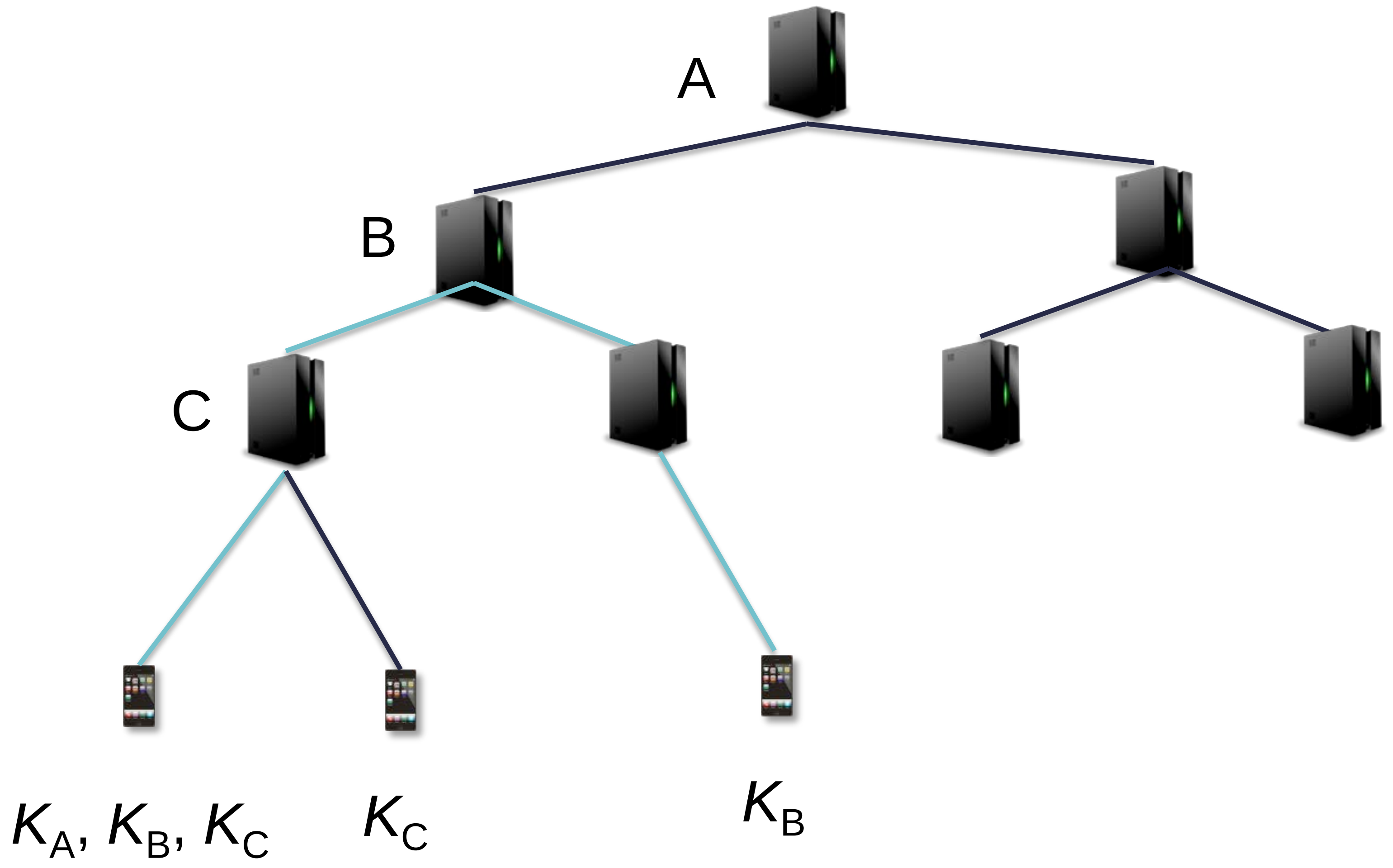
Hierarchical TTP



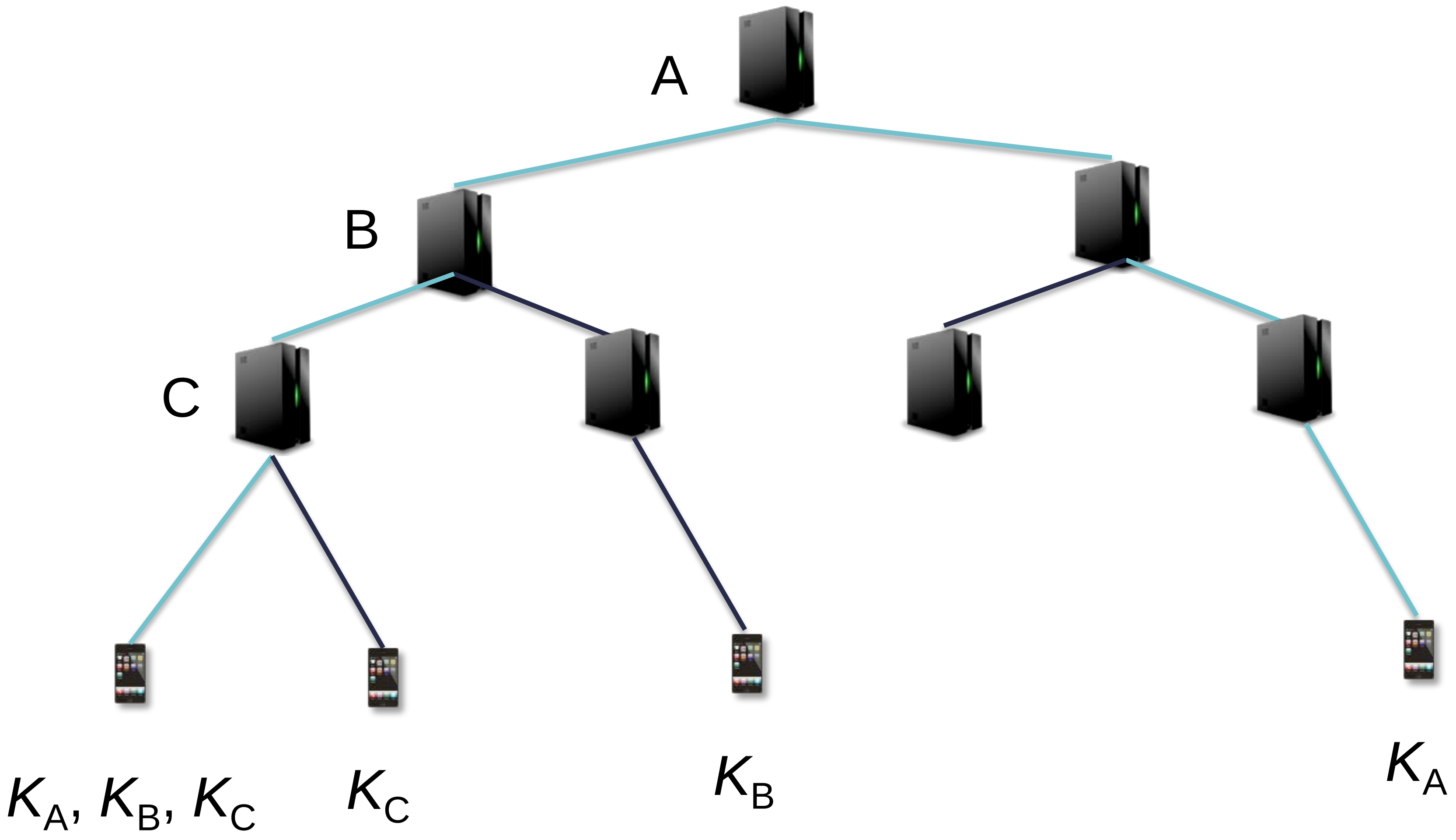
Hierarchical TTP



Hierarchical TTP



Hierarchical TTP



Conclusions

- Engineering for large keys is feasible and useful
  - We can solve many of today's Communications Security problems this way
- Best promise
  - HBS
  - Minimizing and optimizing public key use
  - Revocation using HBS or symmetric cryptography
  - TTP for encryption keys
    - Multiple TTPs
    - HBS authentication

Thank you.