EAP Efficient Re-authentication

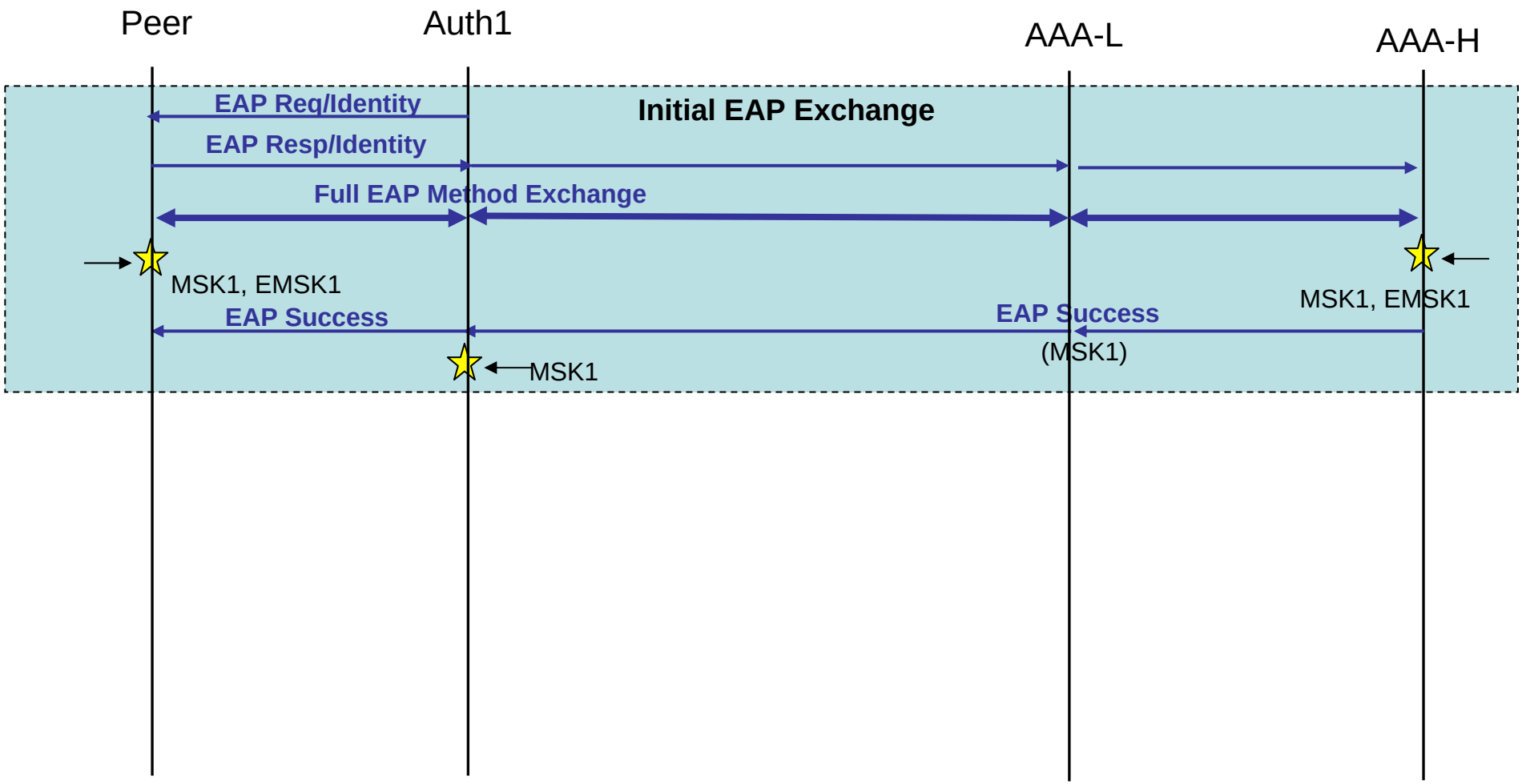
Vidya Narayanan, vidyan@qualcomm.com
Lakshminath Dondeti, ldondeti@qualcomm.com

January 2007

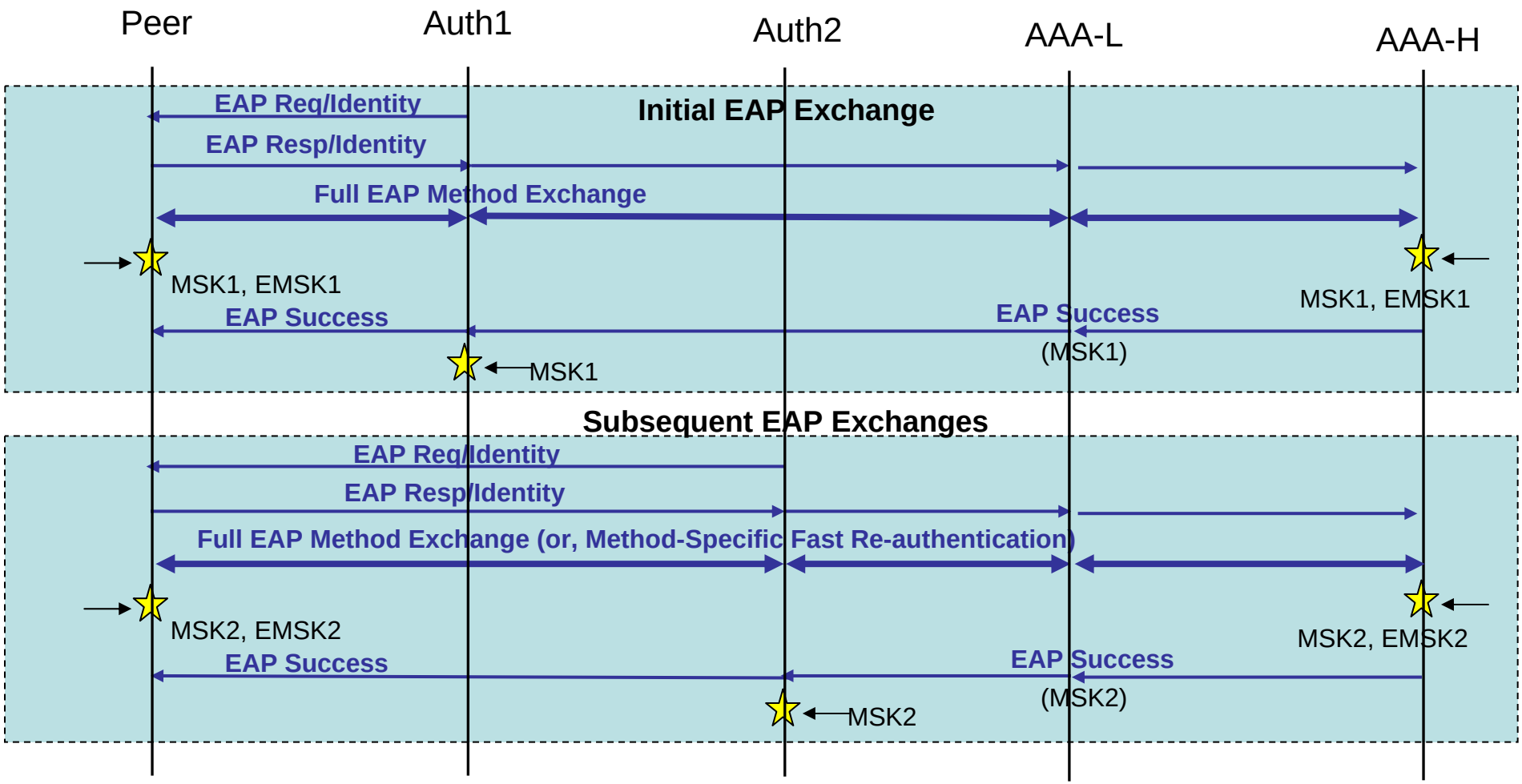
Contents

- EAP Re-authentication and Fast Re-authentication
- Requirements for low latency re-authentication
- Constraints in designing extensions to EAP
- Design Choices
  - Server vs. peer initiated
  - Re-authentication key hierarchy
    - With and without local re-auth server
  - Protocol transport
  - Lower layer requirements
- Proposed Protocol

EAP Re-authentication, as per today's standards



EAP Re-authentication, as per today's standards



Our Charter Dictates ☺

- Solutions specified by the HOKEY WG must:
  - Be responsive to handover and re-authentication latency performance objectives within a mobile wireless access network.
  - Fulfill the requirements in draft-housley-aaa-key-mgmt and draft-ietf-eap-keying.
  - Be independent of the access-technology. Any key hierarchy topology or protocol defined must be independent of EAP lower layers. The protocols may require additional support from the EAP lower layers that use it.
  - Accommodate inter-technology heterogeneous handover and roaming.
  - No changes to EAP methods. Any extensions defined to EAP must not cause changes to existing EAP methods.

Re-auth Goals

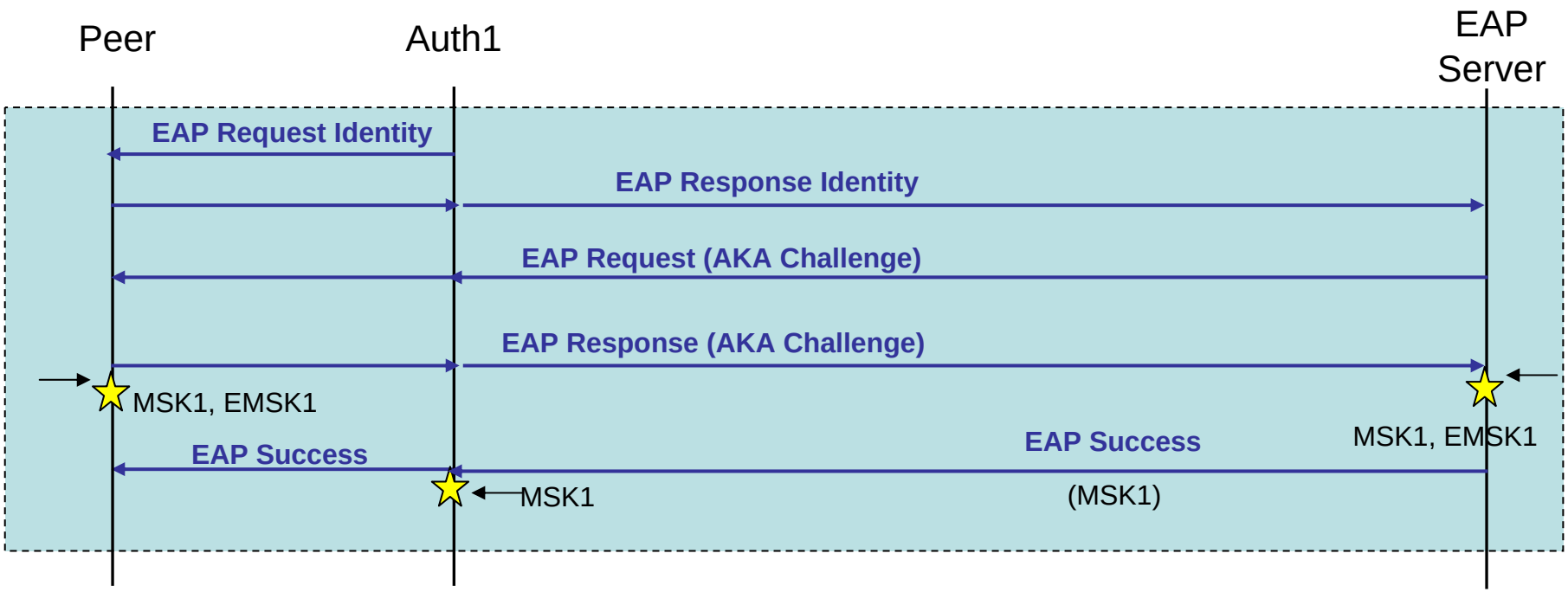
- MUST be better than full EAP authentication
  - “The protocol MUST be responsive to handover and re-authentication latency performance within a mobile access network”

- EAP lower layer independence
- EAP method independence
- AAA protocol compatibility and keying
- Co-existence with current EAP operation

What is Low Latency?

- Security becomes a burden when any latency or overhead is added to the critical handoff path ☺
  - Mobile access networks resort to insecure practices when security adds latency to handoffs
- Two aspects of latency
  - Number of roundtrips
  - Distance to the AS
- Ideally, the protocol should be executable in parallel with connection establishment
  - I.e., add 0 incremental time to L2 handoffs
- It may also be unacceptable to have to go back to the AS (EAP Server) upon every handoff
  - EAP Server may be too many hops away!

Full EAP Authentication (E.g., EAP-AKA)



EAP-AKA takes 2 Roundtrips over the infrastructure to complete; AKA fast re-authentication reduces computational expense, but takes the same number of roundtrips to complete.

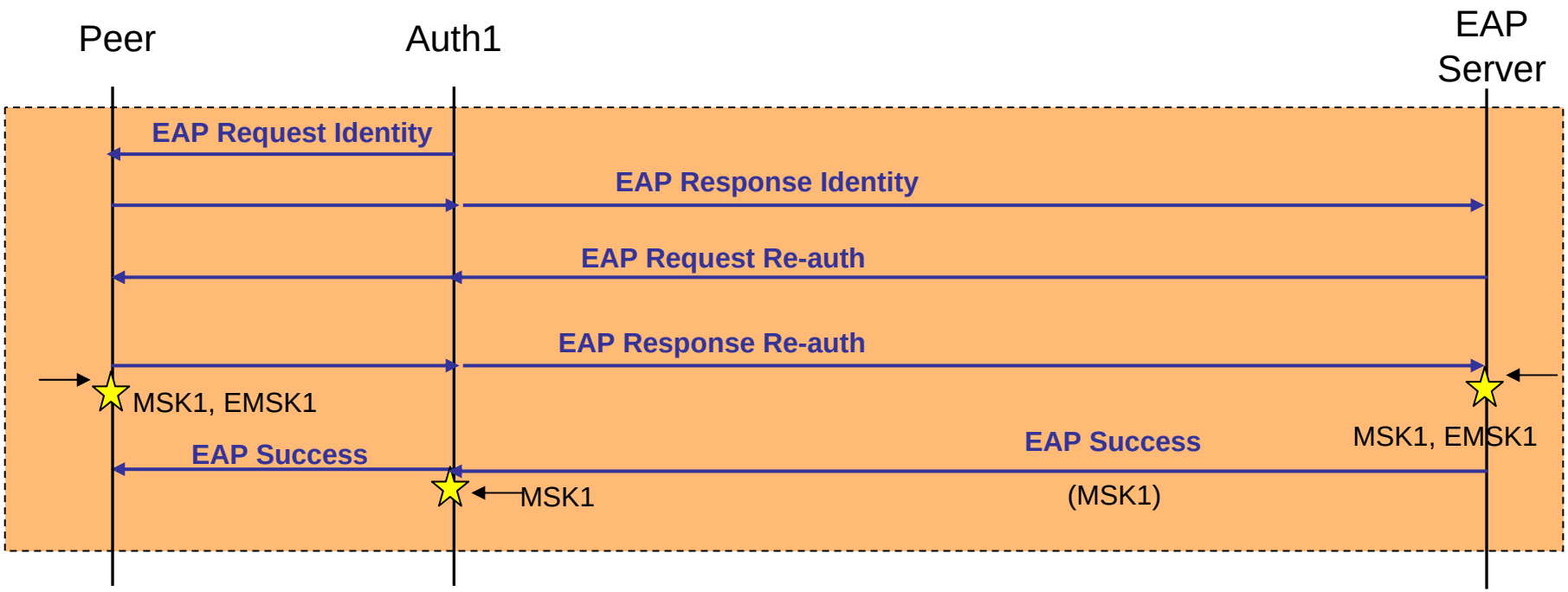
AKA is one of the most commonly used protocols for network access authentication in mobile access networks.

Goal: Re-auth MUST finish in less than 2 roundtrips

Server vs Peer Initiated Re-Auth

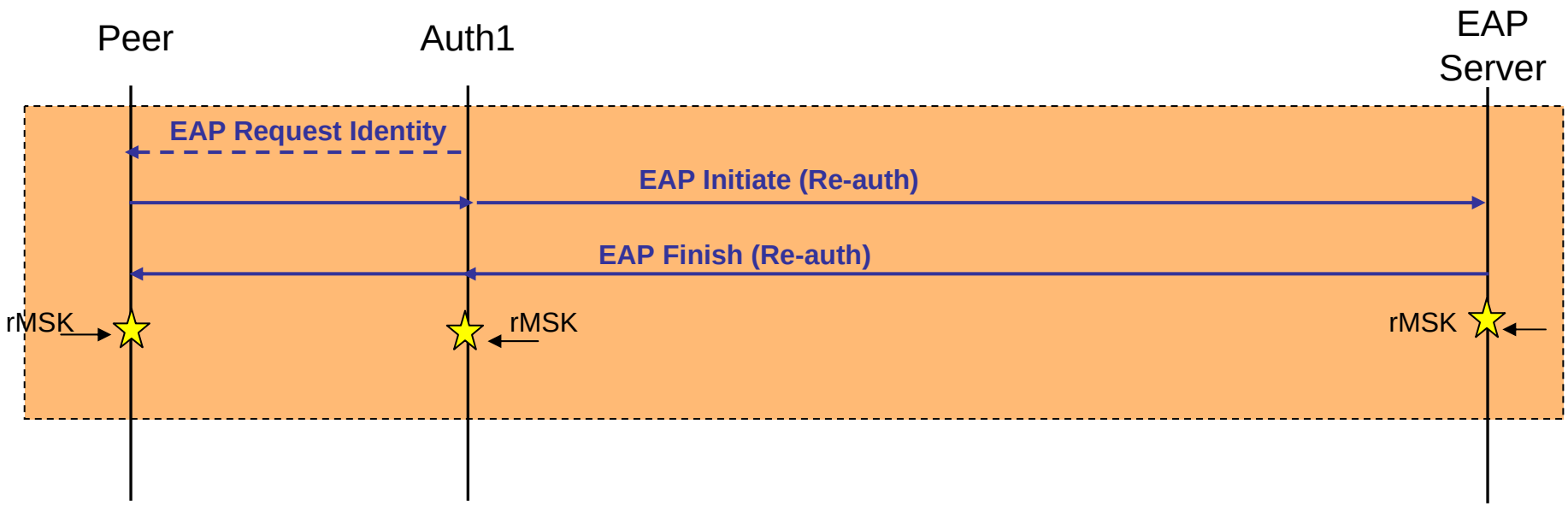
- Server-initiated re-authentication preserves the EAP model
  - Allows for similar peer operation in open/access-controlled networks
  - Only model that supports legacy authenticators
  - Needs at least 1.5 roundtrips with modifications to authenticator operation
  - Needs at least 2 roundtrips with legacy authenticators
- Peer-initiated re-authentication achieves more efficient operation
  - Can piggyback re-authentication on connection establishment on some wireless networks
  - Can finish in 1 roundtrip

Server Initiated Re-Auth With Legacy Authenticators



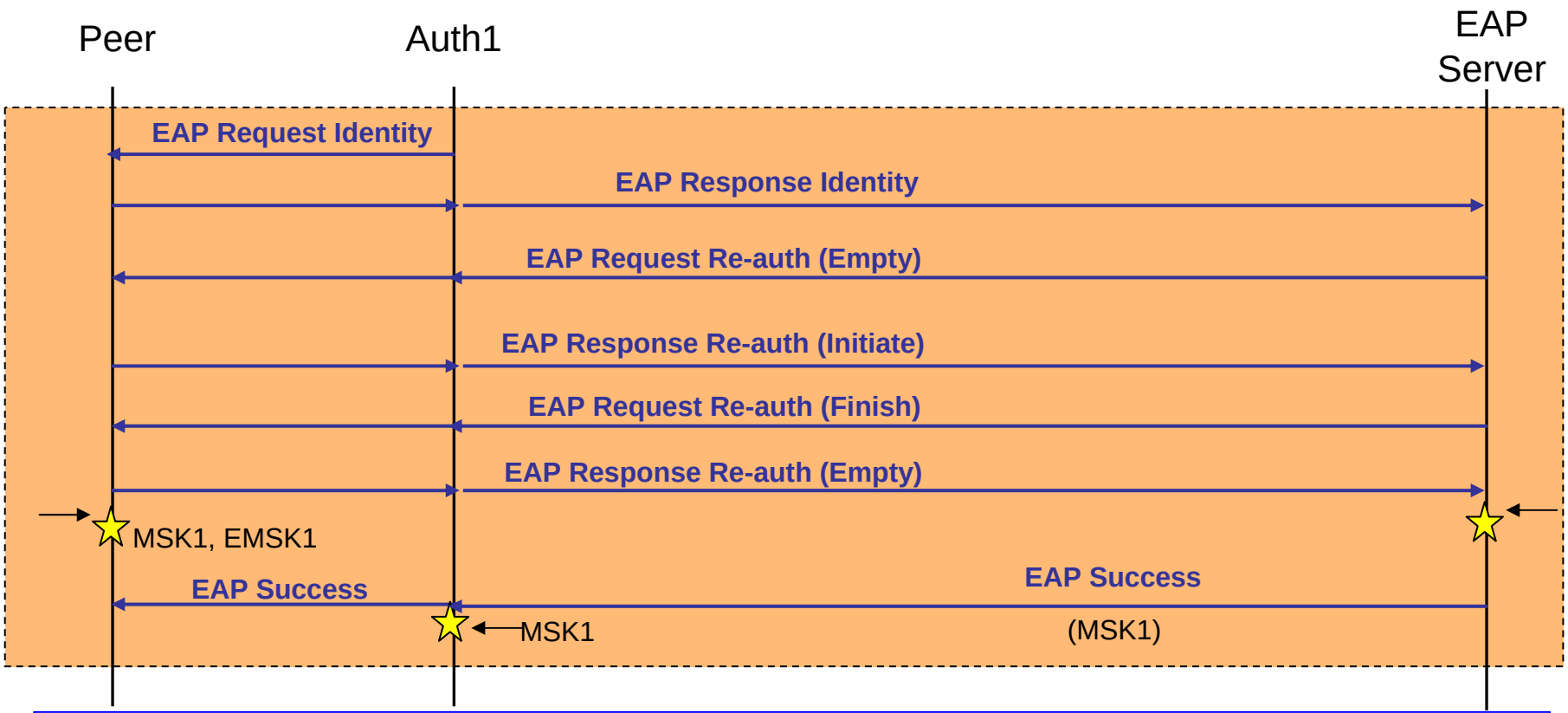
- The protocol operation is quite similar to AKA
  - No improvement over AKA in terms of latency or computational expense
- The Peer has to provide either temporary or real identity in the Response/Identity message
- EAP server has to prove possession of the key before the peer authenticates
  - Potential for DoS attacks on the EAP server

Peer/Server Initiated Re-Auth With Upgraded Authenticators



- The most optimal method of re-authentication is the peer-initiated model
  - Needs upgrades to authenticators
- Optional server-initiated model is also feasible
  - EAP Request Identity from the Authenticator to the peer serves a trigger for Re-Auth
- The Peer authenticates first
  - Uses temporary identity or a key identity for identity protection
- The Finish message contains Server's authentication and also serves the same purpose as EAP Success
- To support peer-initiated operation, changes to peer's state machine are needed
  - Peer must be able to maintain retransmission timers etc.

Peer/Server Initiated Re-Auth With Legacy Authenticators

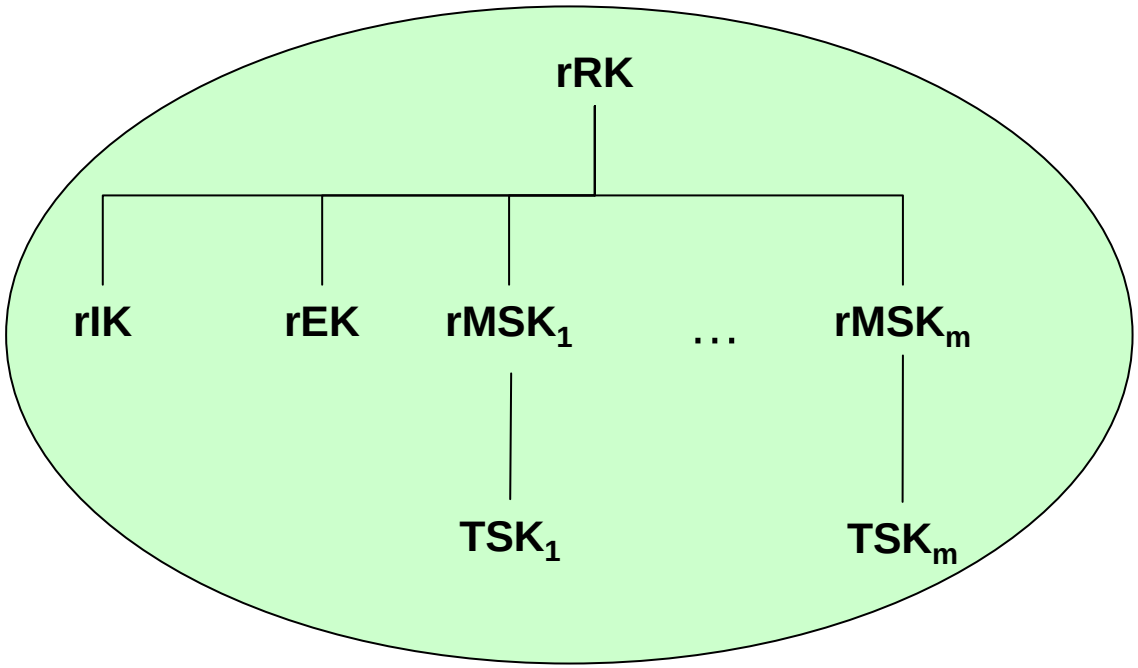


- Optional EAP type-based transport with the peer-initiated model takes more roundtrips than say, EAP-AKA operation
- Enables use of the same protocol over legacy authenticators
  - Only transport varies (code-based vs type-based)
- Peer will have to prove possession of key material before server performs any computation
- Perhaps acceptable as a transition mechanism over legacy authenticators
  - Useful for chatty EAP methods (e.g., TLS-based methods)

Local Re-auth Server

- Re-auth may still take too long if the AS is too many hops away
- Must be able to perform re-auth with a local server when handing off within a local area
- Key hierarchy must support both models
- The re-auth protocol must support some bootstrapping capability
  - Local server must be provided a key
  - Peer may need to be provided a server ID

Re-authentication Key Hierarchy



- rRK is the Re-authentication Root Key
- rIK is the Re-auth Integrity Key and used to provide proof of possession of Re-auth keys
- rEK is the Encryption Key used to encrypt any confidential data exchanged between the peer and the EAP-ER server
- rMSK is the MSK equivalent key
  - Derived based on the run of the EAP-ER protocol
  - Each Authenticator change, whether or not an Authenticator is revisited, is treated the same

Where does the rRK come from?

- There are at least two candidates for the parent key for the rRK
  - f(EMSK, “Re-authentication Root Key”)
    - The EMSK is managed by the EAP server
      - EAP server may be too many hops away from the Peer
  - f(Local MSK, “Re-authentication Root Key”)
    - Local MSK is a root key associated with an EAP-ER server in the Authenticator’s domain
    - Local MSK derivation itself is out of scope of the re-authentication solution

Protocol Transport

- Protocol transport considerations
  - EAP Code-based and Type-based transport
    - EAP Code-based implies authenticator support for new codes
    - EAP Type-based can work with current EAP authenticators
  - One option is to allow both
    - Re-auth messages may be carried in EAP Request/Response or EAP Initiate/Finish messages

- Can re-auth be run over a protocol other than EAP?
  - Claimed benefit is to prevent any changes to EAP implementation
    - Peer and server implementations may treat re-auth as a new protocol for all practical purposes
  - At the authenticator, interactions with EAP are needed irrespective of the transport protocol used for re-auth
    - In many networks, access control enforcement is based on successfully finishing EAP authentication
    - Port control enforcement is contingent on EAP Success, MSK to TSK derivation and use
    - The goal of Re-Auth is also to derive the TSK eventually – port must be enabled after successful re-authentication

Benefits of EAP-based Transport

- If a new protocol were to be used to transport Re-Auth messages
  - Authenticators would have two different protocols and state machines installing SAs that enable controlled access
- EAP-based transport allows:
  - Integration of state machines for initial and re-authentication
  - Specification benefits:
    - Will largely re-use:
      - RFC3748
      - EAP keying framework
      - RFC3579
      - RFC4072
      - The list goes on…
  - Allows re-auth to be triggered by EAP Request Identity

Co-existence with Vanilla EAP

- Peers may roam in and out of networks that support re-authentication
- Support for re-auth may be indicated by the lower layer for optimal operation
- Alternatively, the peer may attempt re-auth
  - Upon a timeout/failure, the peer may do full authentication

Lower-layer Requirements

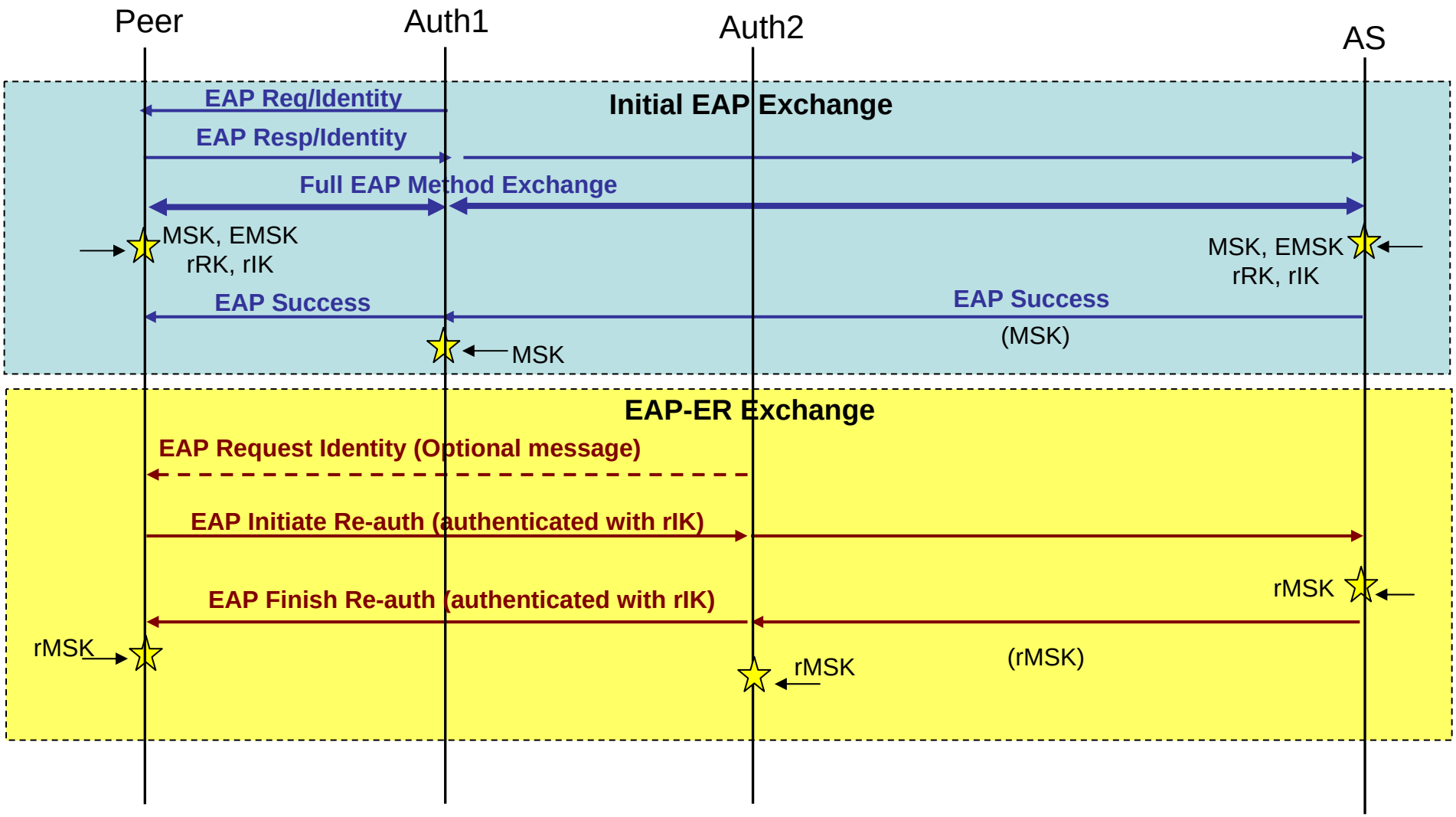
- For optimal operation, the lower layer
  - advertises re-auth capability
  - advertises a local re-auth server
    - Server ID may be obtained from the lower layer at the peer
      - Peer may not need to be “bootstrapped” at the EAP layer
- Key for the local server may be delivered along with the full EAP exchange

EAP-ER Summary

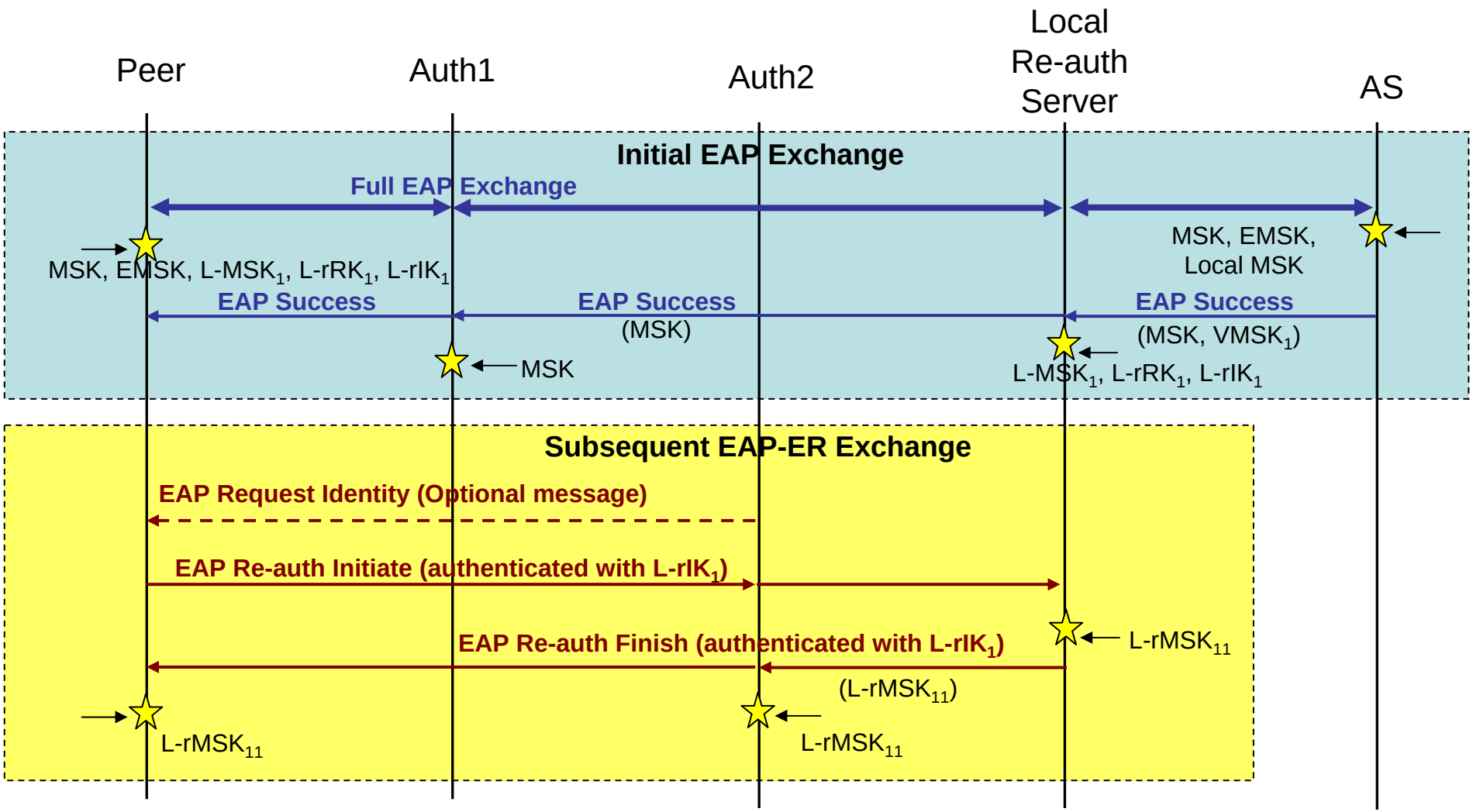
- Method-independent protocol for efficient re-authentication
  - EAP-ER is a single roundtrip re-authentication protocol
  - Access agnostic; can be used for inter-technology handoffs
  - Proof of possession of key material of an earlier authentication
  - EAP-ER execution with a local server

- Key Generation in EAP-ER
  - rRK is the root of the hierarchy
    - May be generated from the EMSK or a local MSK
  - Re-authentication MSKs (rMSK)
    - Serves the same purpose as an MSK

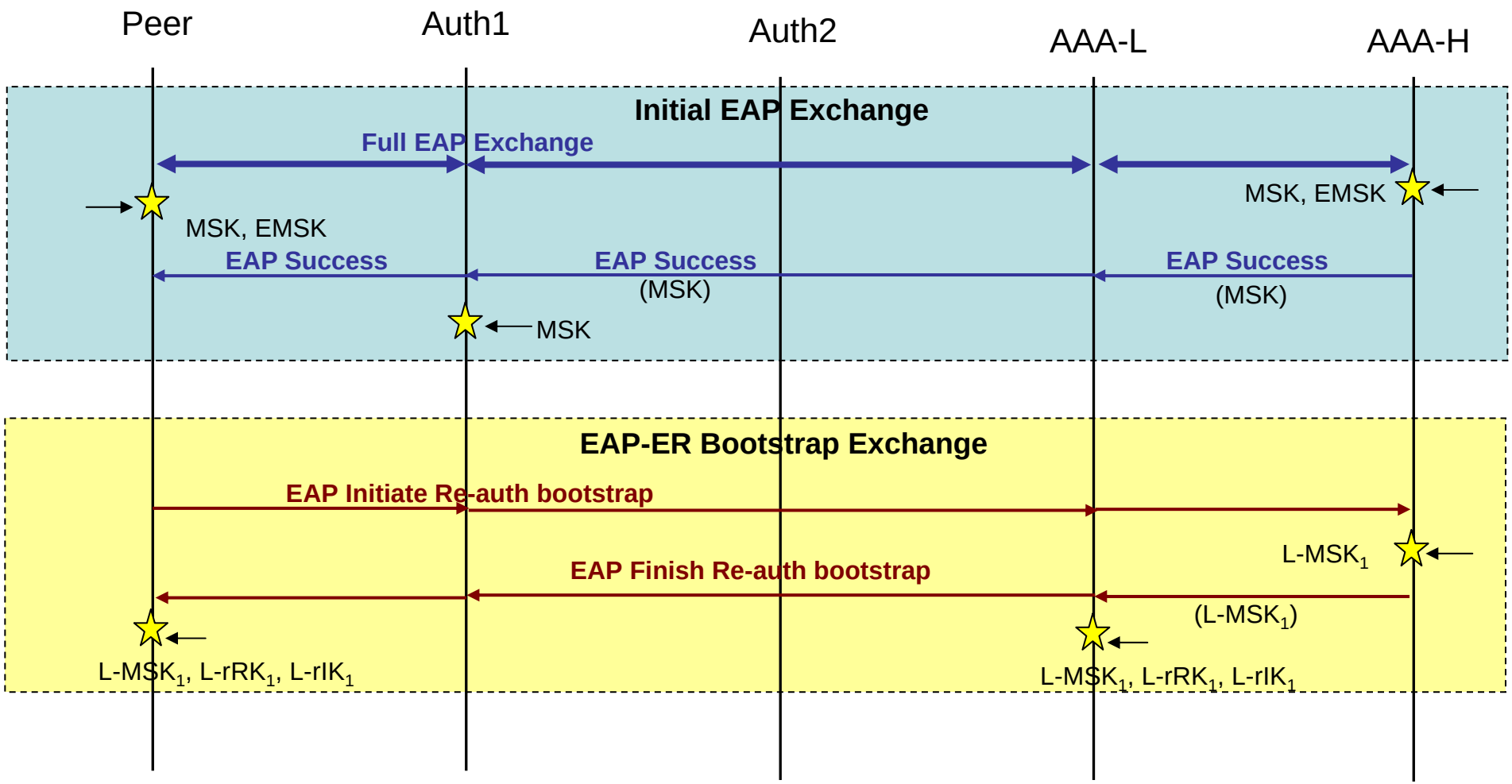
EAP-ER Exchange with AS (EAP Server)



EAP-ER Exchange with Local Re-auth Server



EAP-ER Bootstrap Exchange



Backup Slides

EAP Re-auth Packet format

Code	Identifier	Length
Type	Flags	SEQ
1 or more TVs or TLVs containing identities		
Crypto-Suite	Authentication Tag (variable)	
Authentication Tag (contd)		

Type	Length	Value (variable length)
Value (contd)		

EAP-ER attributes

- Peer sends an EAP Initiate Re-auth message with
  - rIKname for key lookup and Proof of possession verification
  - server-id (optional)
  - Peer-id, NAI (optional)
    - If neither peer-id nor server-id are present, rIKname must be in the form of an NAI
- Code indicates Initiate/Finish
- Flags indicate bootstrap or not
- SEQ for replay protection
- Crypto-suite indicates the algorithm used for integrity protection
- Authentication tag is the proof of possession of the rIK

Key derivation

- rRK = prf+ (K, S), where,
  - K = EMSK and
  - S = rRK Label
    - (“EAP Re-authentication Root Key”)
- rRK_name = NDF-64(EAP Session-ID, rRK Label)
- rIK = prf+ (rRK, "Re-authentication Integrity Key")
- rIK_name = prf-64 (rRK, "rIK Name")
- rMSK = prf+(rRK, SEQ)