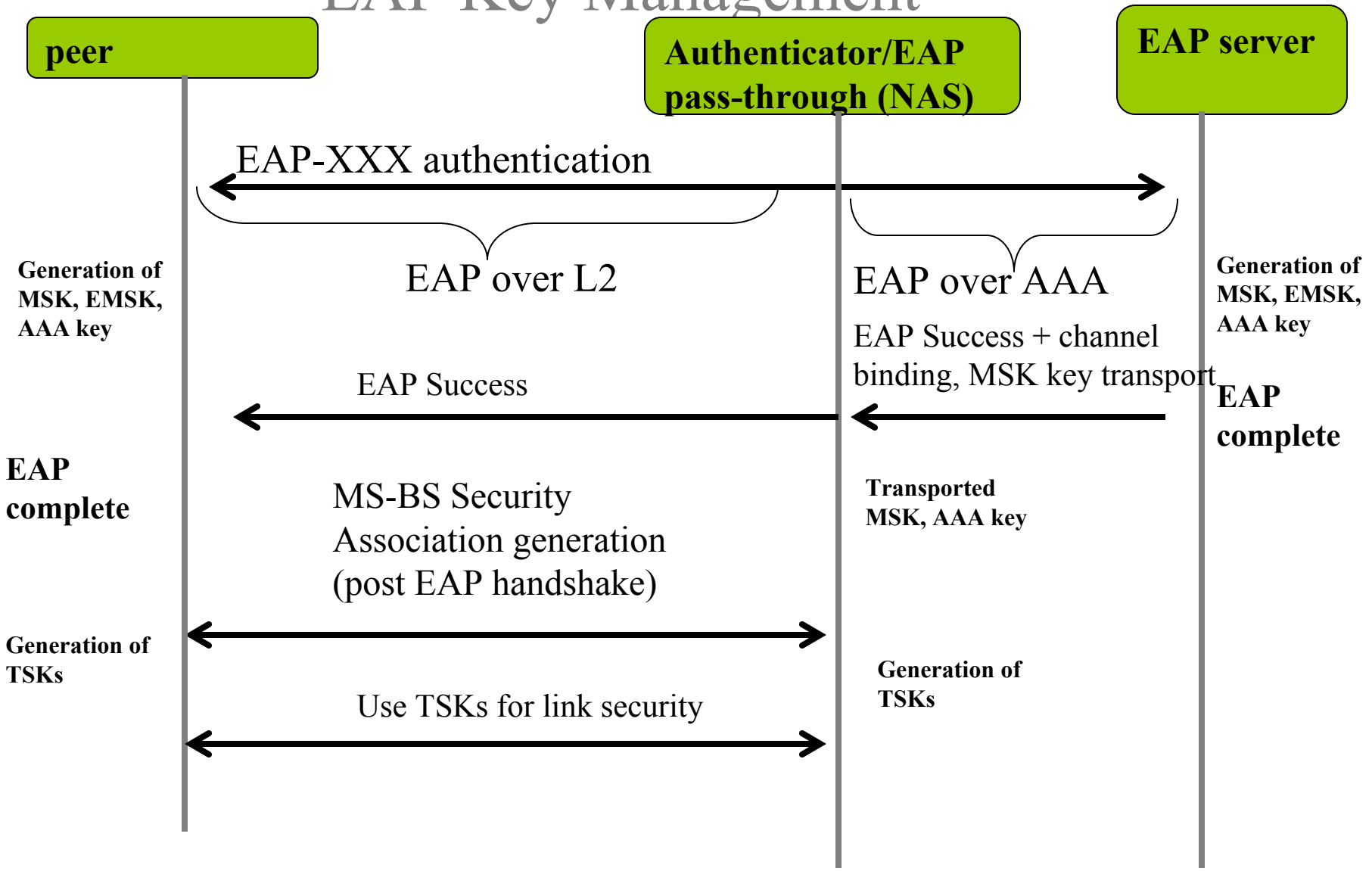
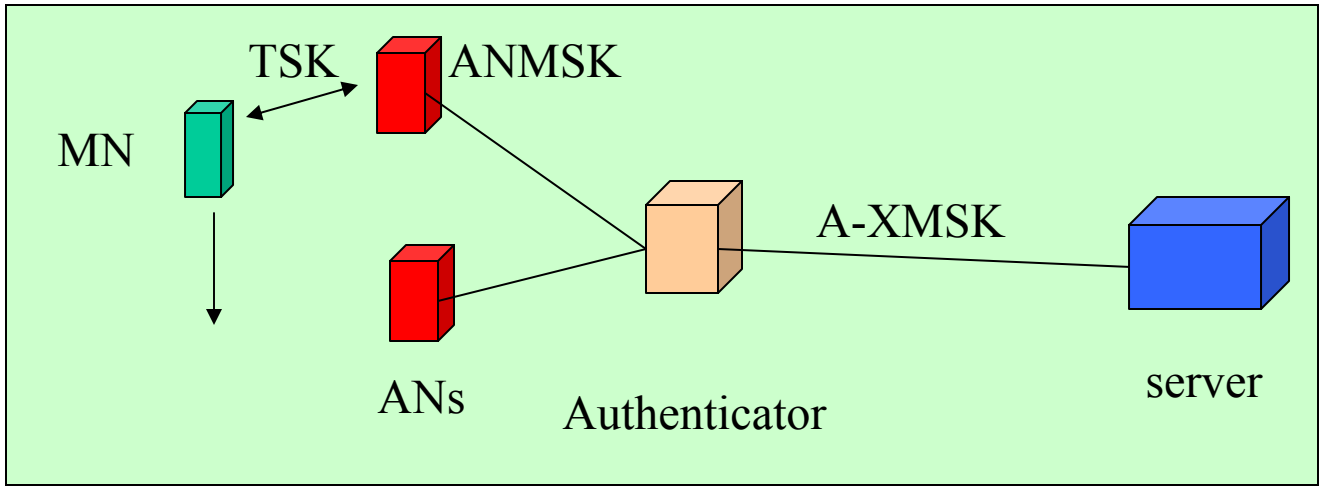
EAP and Handover
draft-nakhjiri-eap-ho-ps-00.txt
IETF 64, November 2005

EAP Key Management

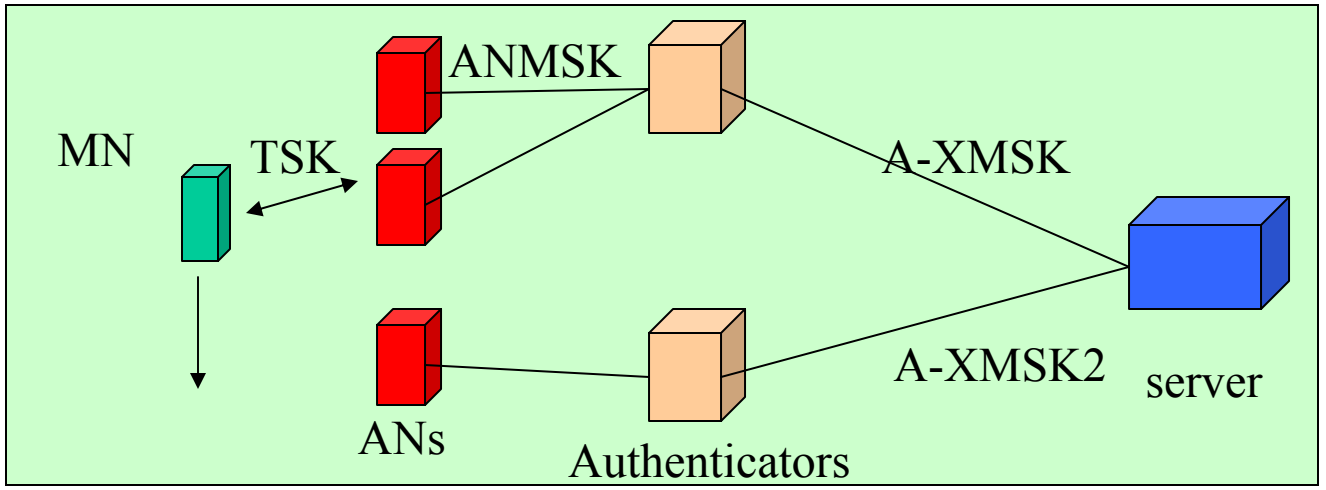


Wireless or Mobility scenarios (1)



- The Access Node (AP, AR, BS) is a cheap device with an Authenticator in the back.
  - A-XMSK generated from EAP-XMSK
  - An ANMSK to be generated from A-XMSK received from server
  - Peer-AN TSK bootstrapped from ANMSK received from Authenticator
  - New Peer-AN TSKs needed after each AN-HO.
    - New ANMSK from the authenticator (from same A-XMSK)
    - ANMSK context transfer?
  - How to do channel binding? Bind scope to what identities?
  - Who can caches, use, drop?

Wireless or Mobility scenarios (2)



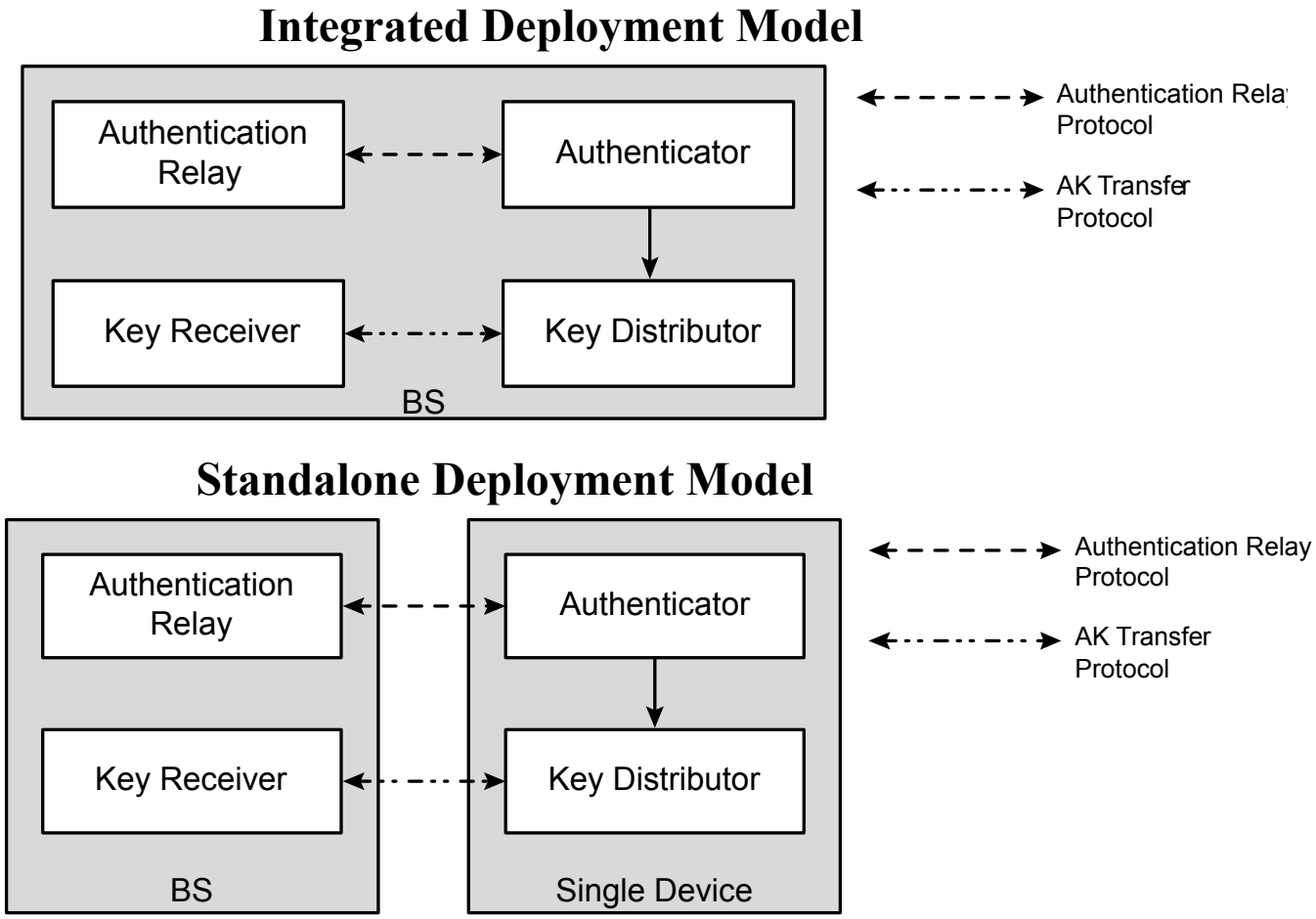
- Mobility between authenticators (with or without ANs in the way).
  - New Authenticator gets a new A-XMSK from AAA?
  - New authenticator gets A-XMSK locally

Connectivity through new Authenticator
Discussion in draft-nakhjiri-eap-ho-ps-00

- Old→ New Authenticator A-XMSK CT?
- New A-XMSK at Authenticator?
  - Go back to AAA? (too slow for mobility)
    - A new EAP authentication?
    - Or a new A-XMSK from previous EAP (EAP-XMSK)
      - Who has EAP-XMSK? AAA server? EAP server? Central KDC?
      - What is EAP-XMSK in EAP hierarchy?
  - From a local KDC? (faster for mobility)
    - An entity that has cached EAP-XMSK and can calculate A-XMSK?
  - First Authenticator acts as an anchor
  - Level in the hierarchy?

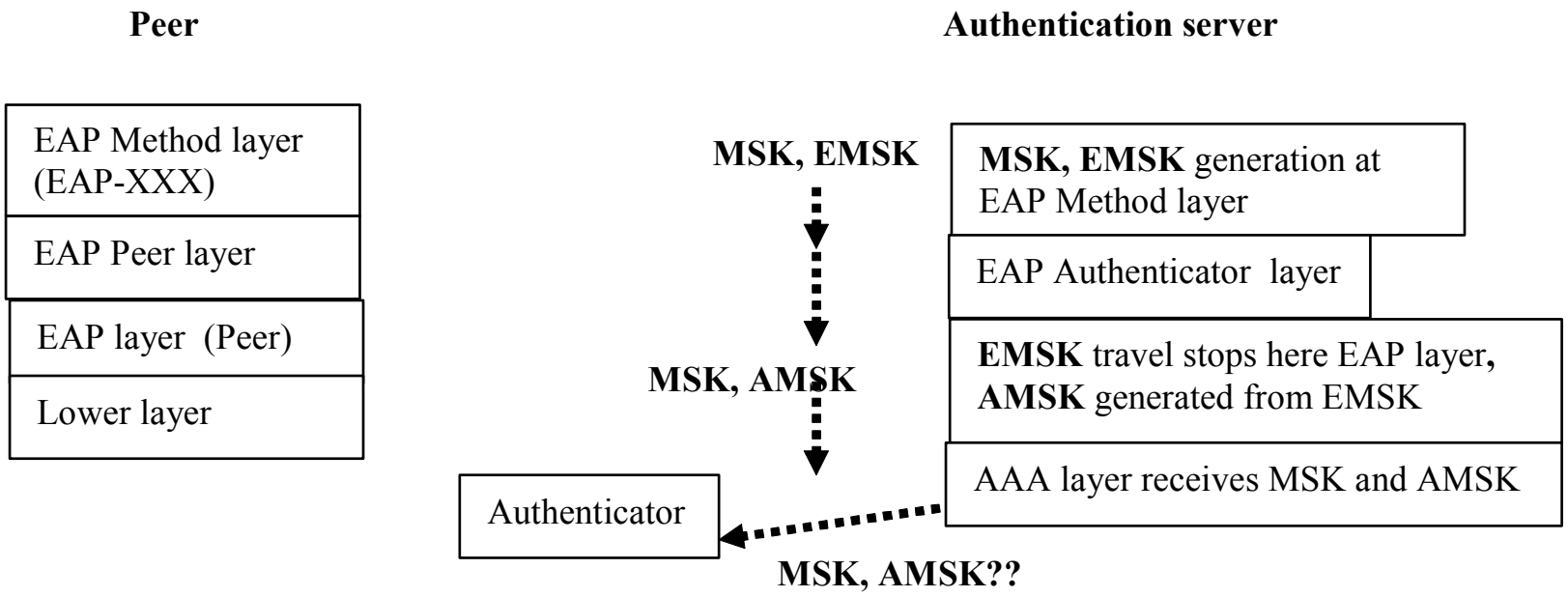
Which one meets the AAA key management requirements???

EAP architecture WiMAX stage 2, Rel 1



Use of EAP keying draft

Current EAP Keying details(?)



And the fine print…

- MSK and EMSK are created by the EAP method at the EAP server.
- EAP layer exports MSK (but not EMSK) to AAA server
- Application keys (AMSK) generated only from EMSK (not MSK), so only EAP server (not AAA server) can do it
- EAP server deletes EMSK right away, meaning it cannot create any AMSKs later on.
- The AAA server can receive MSK and AMSK, but cannot create its own AMSK.
- If the AAA server sends any of MSK and AMSK to another entity, but it must delete it right away.
- Channel binding (key to peer, authenticator binding) prohibits CT?
- Not a well defined method to carry keys from the AAA server to the authenticator.

Key derivation Suggestions?

- Define XMSK:=AMSK
- Use existing AMSK definition
  - AMSK=KDF(EMSK, "key derivation for multiple attachments", length)
    - If EMSK only at EAP server, Cache AMSK at AAA server for future use

EAP WG?

- A-AMSK1=Bind AMSK to (peer ID, authenticator ID)
  - Send A-AMSK1 to authenticator and delete at AAA? What if we want to do a re-authentication? What to do?
  - HO: AAA server create A-XMSK2 for new authenticator from cached AMSK at AAA server and send away
- AN-XMSK=created according to link KDF and specs.
  - AN handover: create AN-XMSK2 at authenticator
  - Who does channel binding?
  - We may have to do context transfers to support fast handovers.

XYZ WG?

Extensions to current spec

- AAA server should be able to create AMSKs itself, since AAA authorizes services
- AAA layer-EAP layer protocol undefined
- Channel binding at all layers.
  - Authenticator-BS (WiMAX authentication relay) protocol, messaging
- AAA server is the source of trust, it should not have to delete keys it transports
- It should be possible to do re-authentications and handover without having to do a full authentication.
- AAA support (not in EAP WG)
  - Define a RADIUS-way of carrying the key to authenticator
  - Authorizations/ lifetimes/ key scopes
- What architecture model to choose for handover?
  - Local KDC at the edge?
  - Split authenticator architectures (WiMAX)? (still no support for auth-auth HO)