802.1X & EAP & Keying

State Machines and Interfaces

Jim Burns

Paul Congdon

Nick Petroni

John Vollbrecht

The Working Groups

- Several specifications MUST align to enable a working implementations:
  - IEEE 802.1aa (update to 802.1X)
    - http://www.ieee802.org/1/files/private/aa-drafts/d5/
  - IEEE 802.11 TGi (security)
    - http://www.ieee802.org/11/private/Draft_Standards/11i/802.11i-D3.0.doc
  - RFC 2284bis (EAP)
    - http://www.levkowetz.com/pub/ietf/drafts/eap/
    - http://www.ietf.org/internet-drafts/draft-ietf-eap-rfc2284bis-01.txt
    - http://www.drizzle.com/~aboba/EAP/eapissues.html
  - EAP state machine work
    - http://www.ietf.org/internet-drafts/draft-ietf-eap-esteem-01.txt
  - RFC 2869bis (RADIUS support for EAP)
    - http://www.drizzle.com/~aboba/EAP/draft-aboba-radius-rfc2869bis-10.txt
  - Draft-congdon (RADIUS and 802.1X)
    - http://www.ietf.org/internet-drafts/draft-congdon-radius-8021x-23.txt

What has been done so far?

- A number of issues resolved with RFC 2284bis (EAP)
  - http://www.drizzle.com/~aboba/EAP/eapissues.html
- Interface between 802.1X and EAP well defined
  - http://www-personal.umich.edu/~jrv/eap.htm
- Preliminary EAP state machines defined
  - http://www.cs.umd.edu/~npetroni/EAP/
- Last call on RFC 2869bis (RADIUS/EAP)
- Last call on draft-congdon (RADIUS/802.1X)
- Proposed changes to 802.1X machines and 802.1aa/D5
  - This presentation
- Proposed changes to key interface for 802.11i
  - This presentation

Resulting Issues to Discuss 802.11 & 802.1X

- How to best incorporate 802.11 into the 802.1X/EAP interface diagrams?
- What is the proper sequence for key exchange and sending final EAP-Success?
- What is the interface to generic 4-way handshake machine?
- Where to define the specification of EAPOL-Key message processing?

Consensus from 802.11i Ad-Hoc Interim on Keying

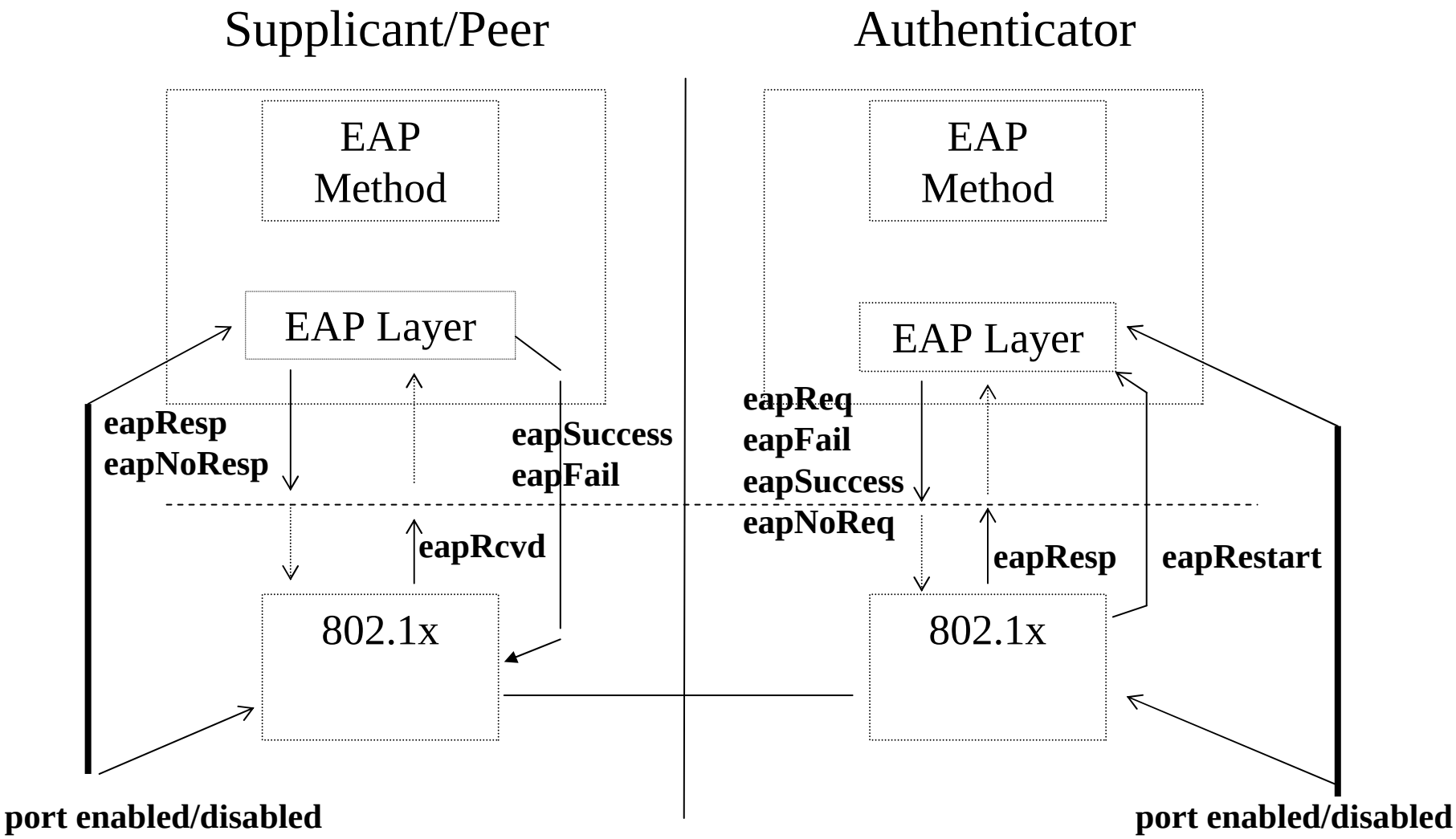
- Recommend that current key machines in 802.1aa are optional
  - Indicate that other key machines defined in 802.11i may be used
  - Indicate in 802.11i that 4-way handshake ‘replaces’ key machines of 802.1X and does not ‘use’ them as defined.
- Recommend and document appropriate key machine interface in 802.1aa
  - Diagram interface to key machines
  - Define variables and interface procedures
- Force opposite sequence of EAP-Success and key machine initiation in 802.1aa

Proposed 802.1aa/D5 Changes

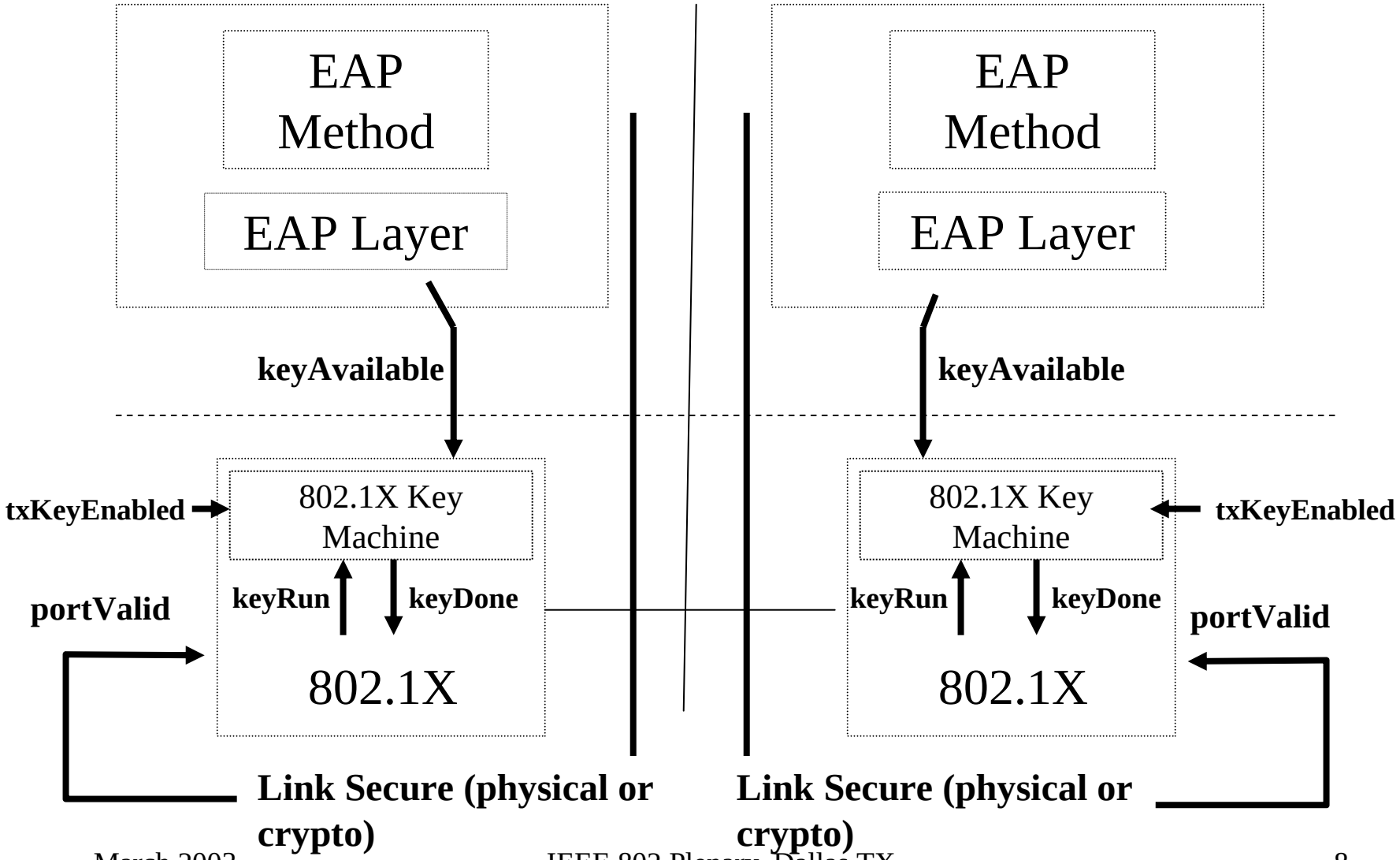
- Specification of interface between EAP/802.1X
- No more EAP packet processing in 802.1X
- Addition of controlled port in Supplicant
- Initial Authenticator request comes from EAP
- Ability for EAP to silently discard frames
- Proposed inclusion of EAP machines in 802.1X Annex
- EAPOL-Key exchange sequenced before EAP-Success
- Propose to include generic key machine interface within 802.1X

EAP / 802.1X Interface

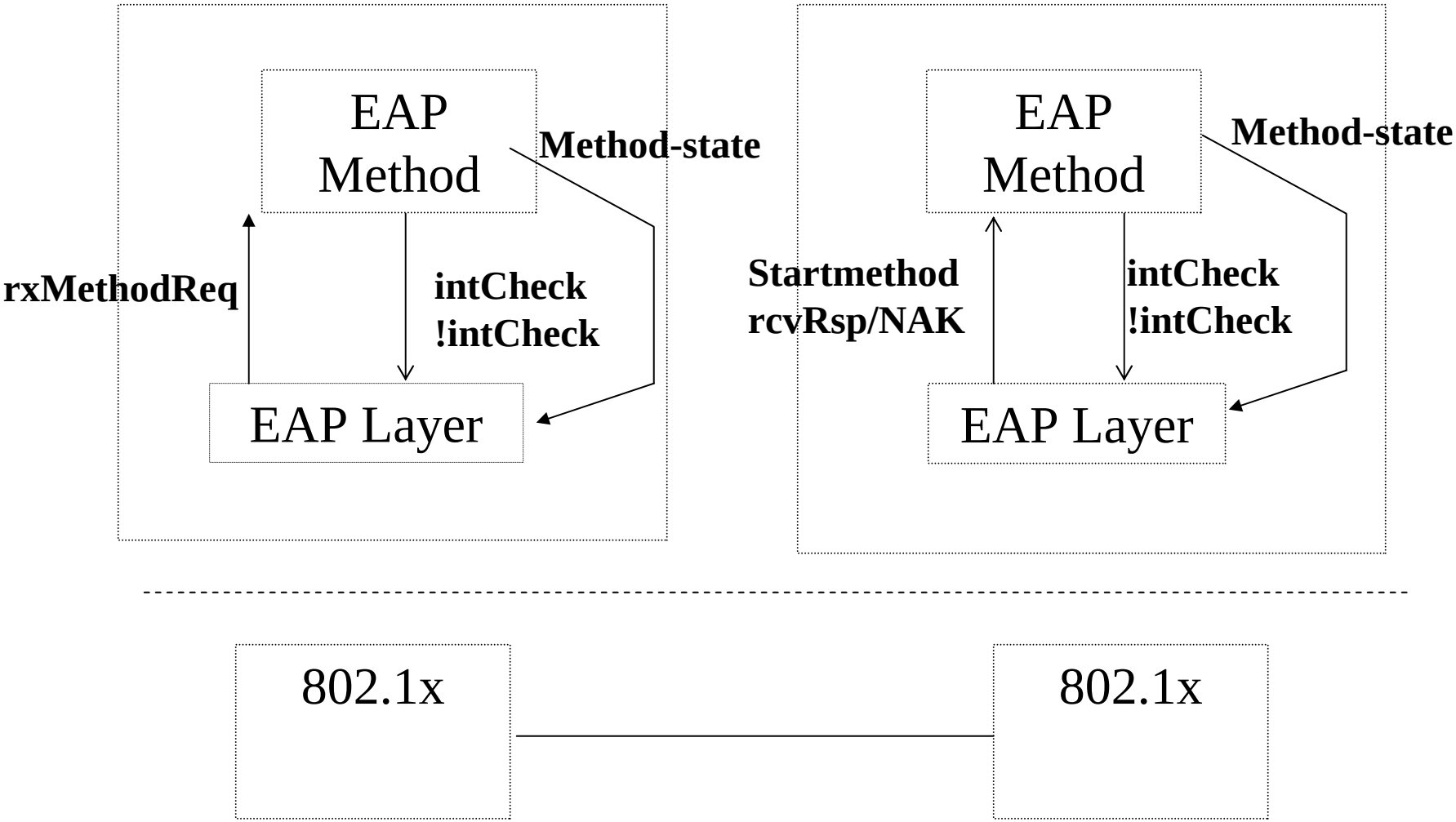
(excluding key exchange)



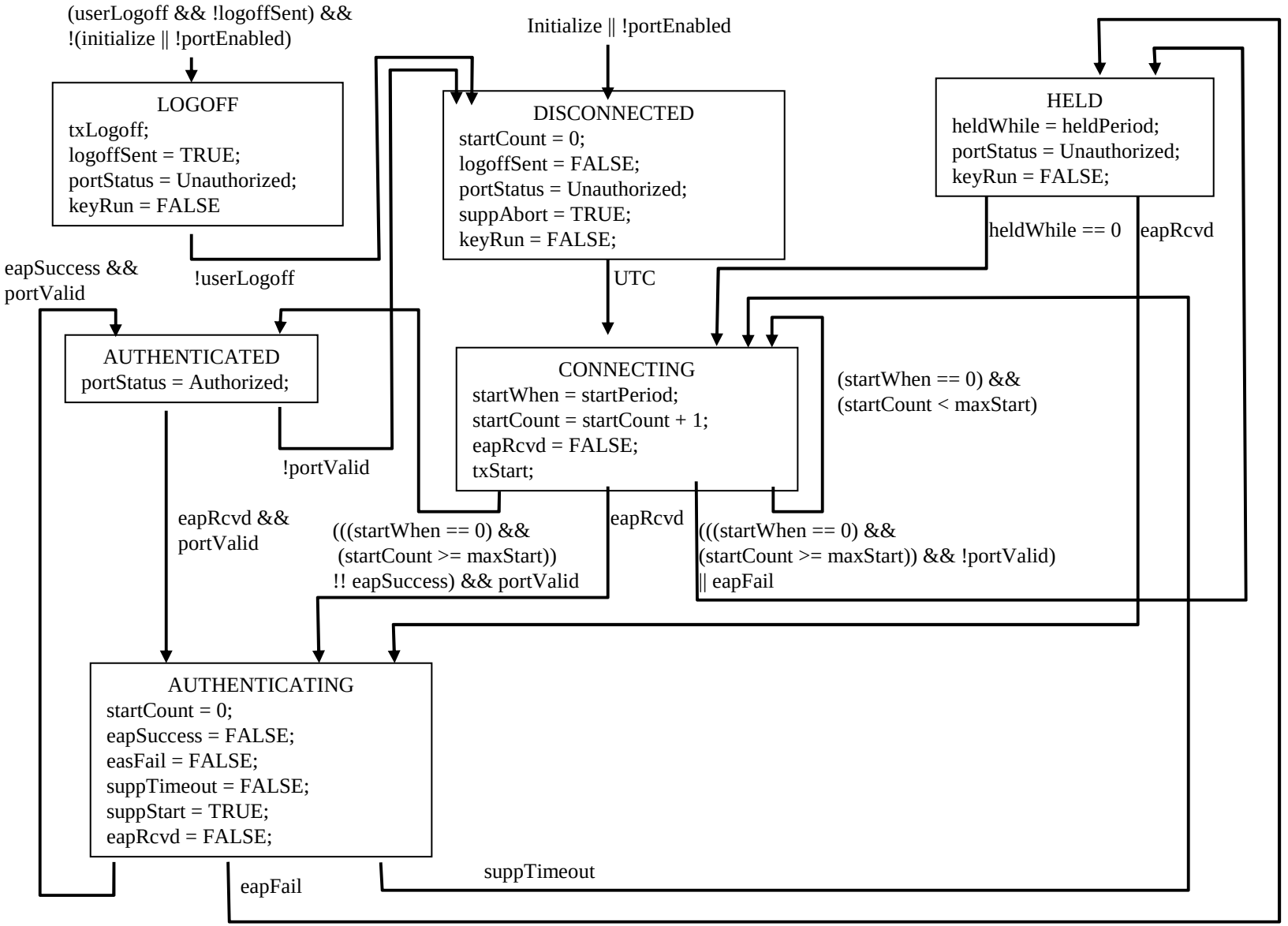
Key Interface with EAP 802.1X & 802.11



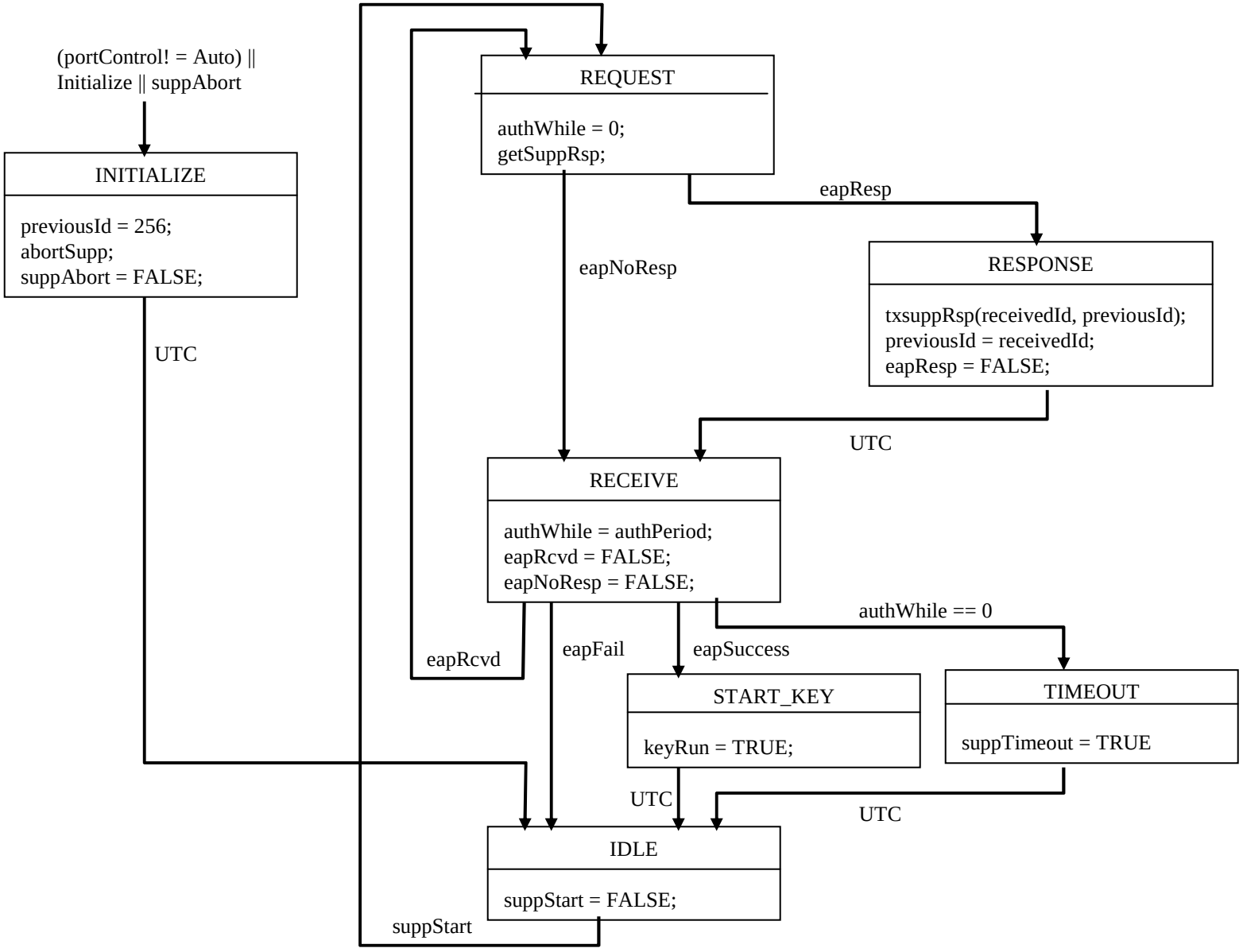
EAP / EAP Method Interface



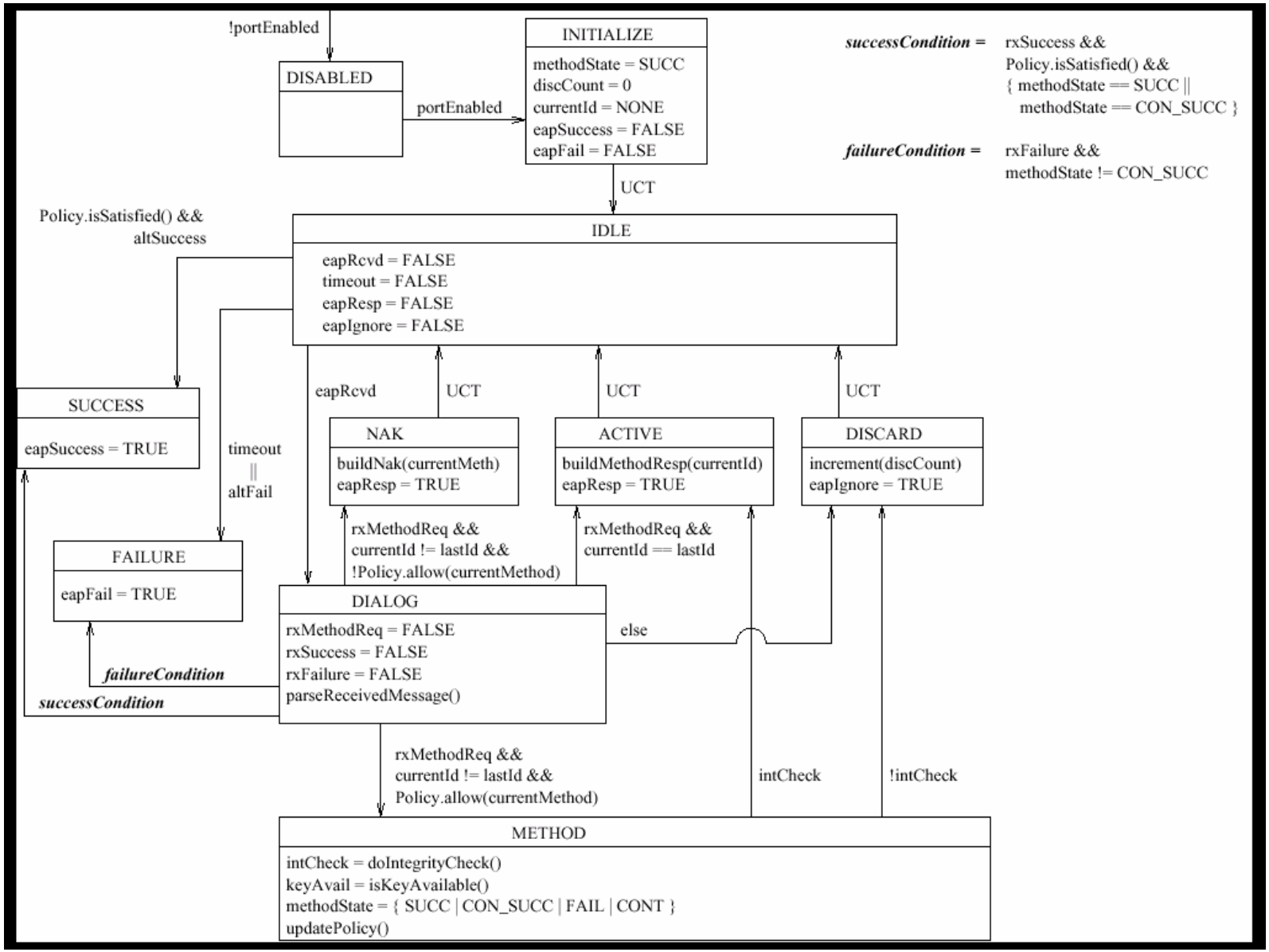
Supplicant Front-End



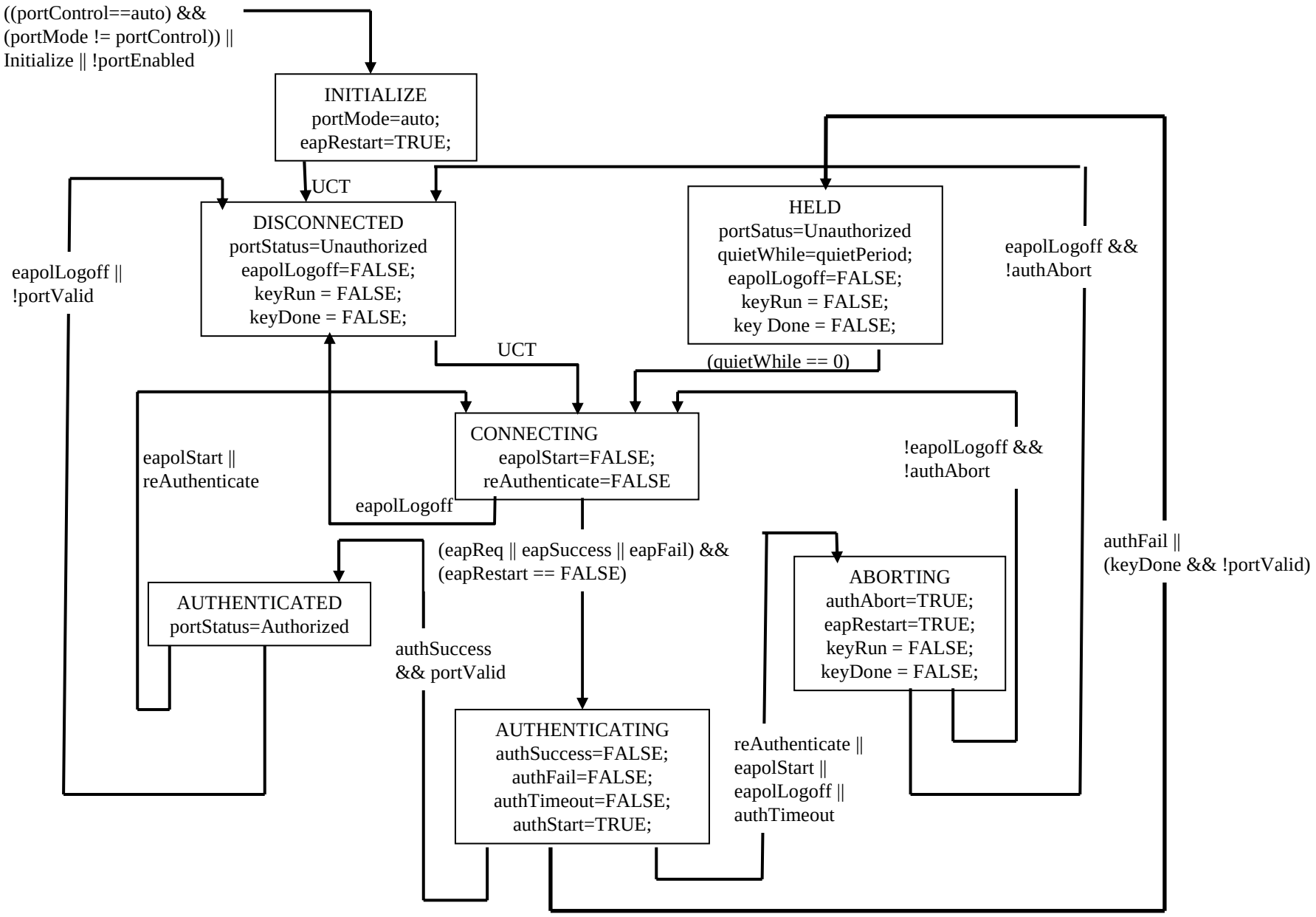
Supplicant Back-End



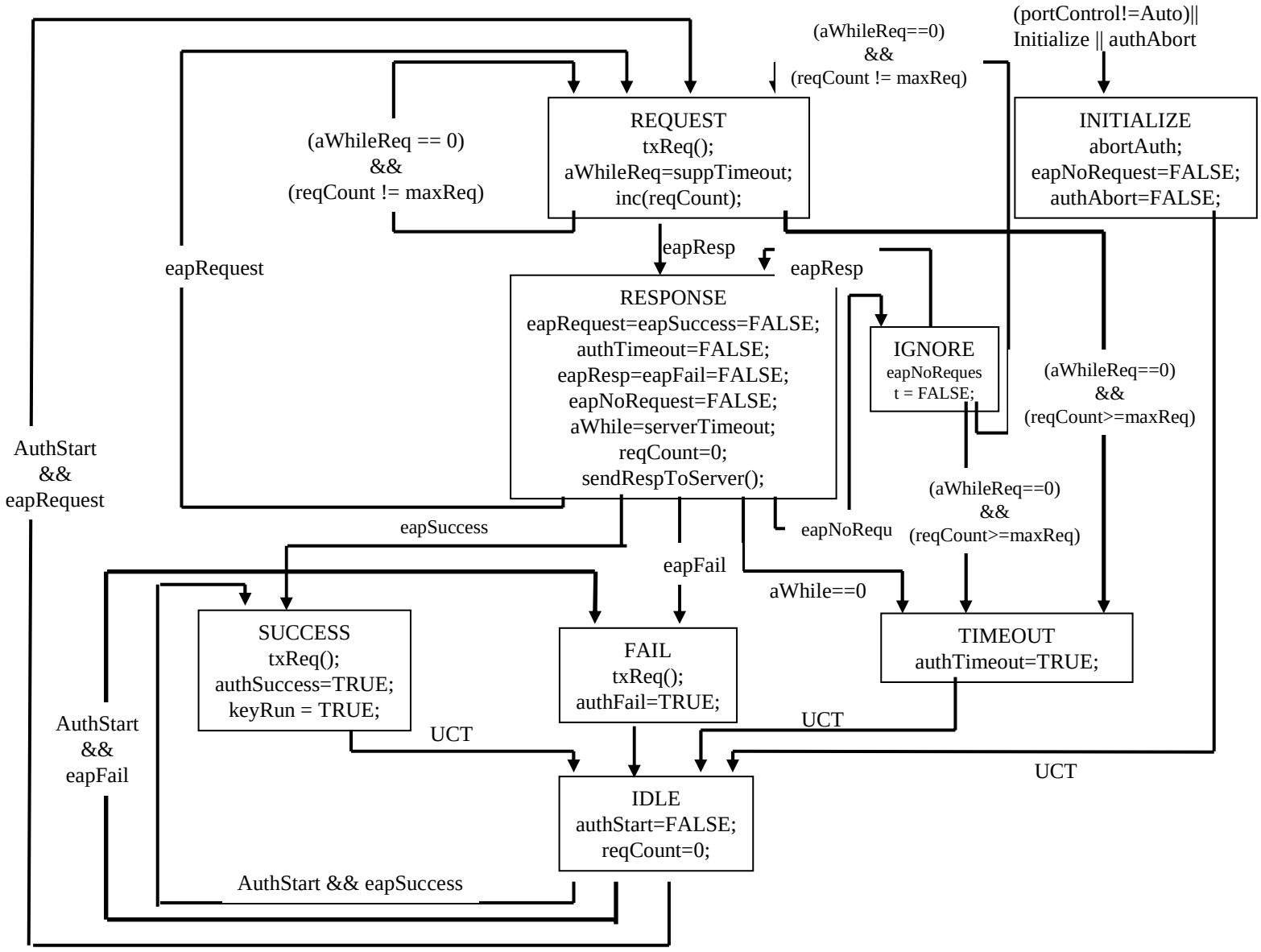
EAP Peer (v6)


!portEnabled
DISABLED
portEnabled
INITIALIZE
methodState = SUCC
discCount = 0
currentId = NONE
eapSuccess = FALSE
eapFail = FALSE
successCondition = rxSuccess && Policy.isSatisfied() && { methodState == SUCC || methodState == CON_SUCC }
failureCondition = rxFailure && methodState != CON_SUCC
UCT
Policy.isSatisfied() && altSuccess
IDLE
eapRcvd = FALSE
timeout = FALSE
eapResp = FALSE
eapIgnore = FALSE
SUCCESS
eapSuccess = TRUE
eapRcvd
UCT
UCT
UCT
timeout || altFail
NAK
buildNak(currentMeth)
eapResp = TRUE
ACTIVE
buildMethodResp(currentId)
eapResp = TRUE
DISCARD
increment(discCount)
eapIgnore = TRUE
FAILURE
eapFail = TRUE
rxMethodReq && currentId != lastId && !Policy.allow(currentMethod)
rxMethodReq && currentId == lastId
DIALOG
rxMethodReq = FALSE
rxSuccess = FALSE
rxFailure = FALSE
parseReceivedMessage()
else
failureCondition
successCondition
rxMethodReq && currentId != lastId && Policy.allow(currentMethod)
intCheck
!intCheck
METHOD
intCheck = doIntegrityCheck()
keyAvail = isKeyAvailable()
methodState = { SUCC | CON_SUCC | FAIL | CONT }
updatePolicy()

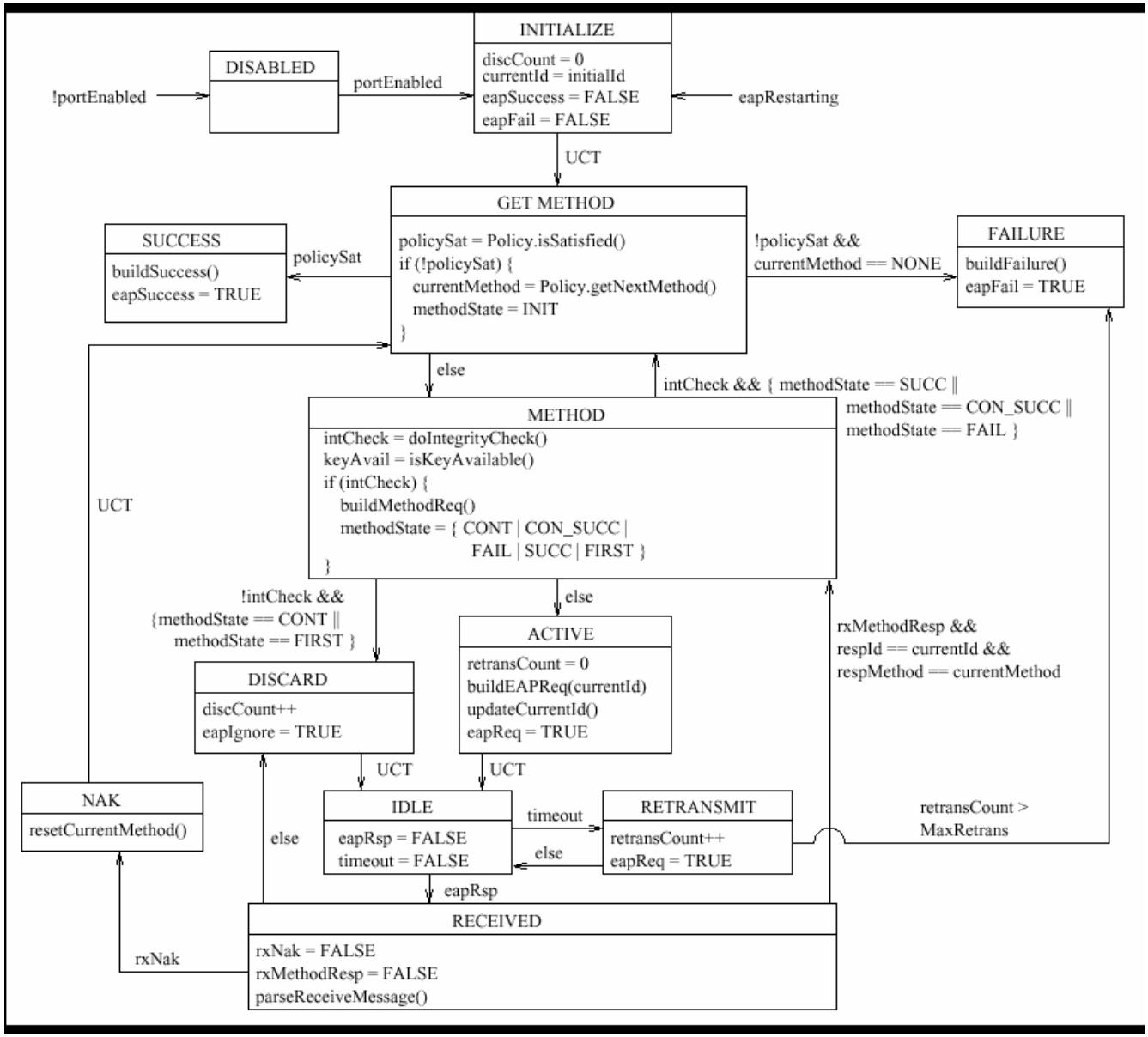
Authenticator Front-End



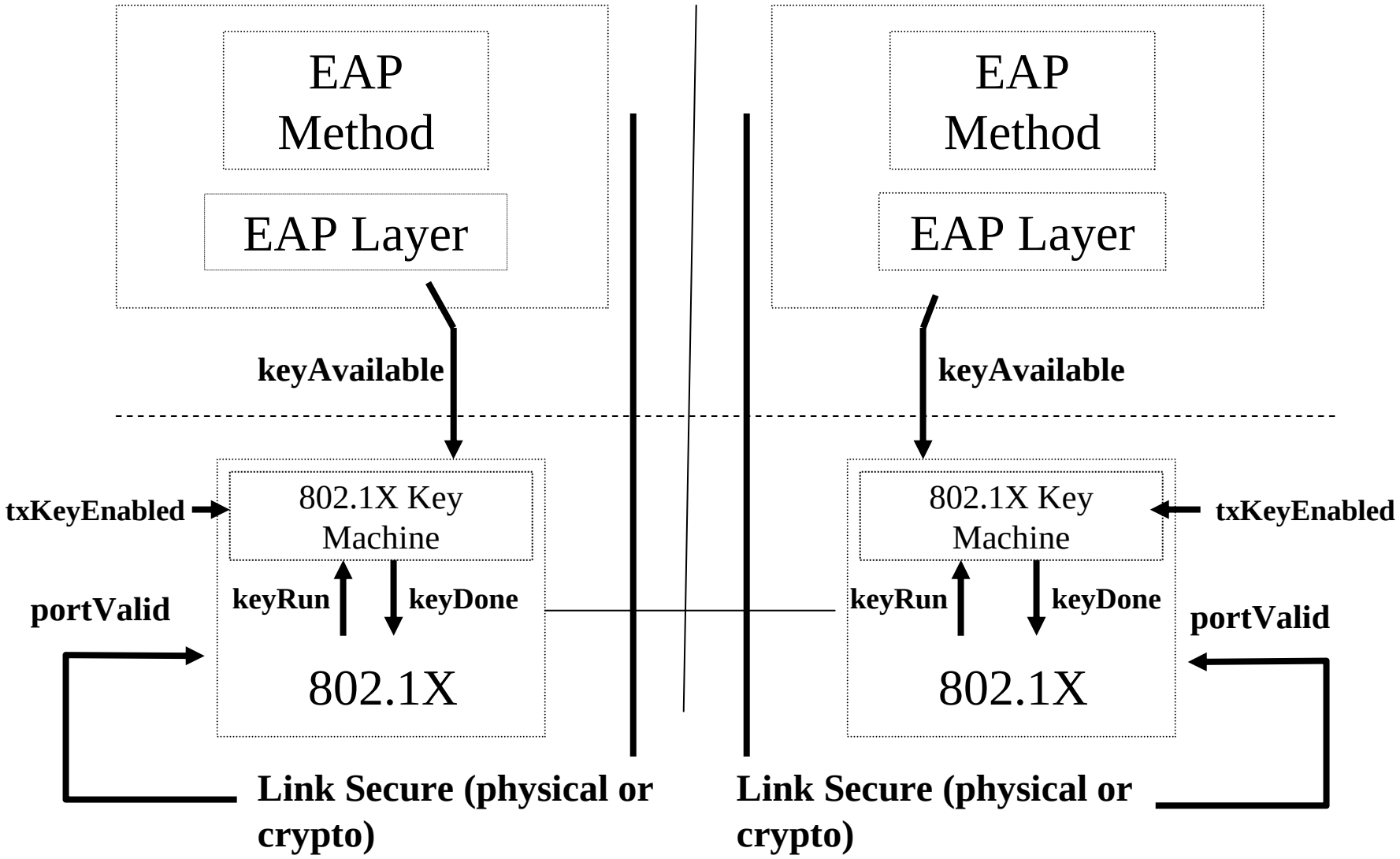
Authenticator Backend



EAP Authenticator (v6)



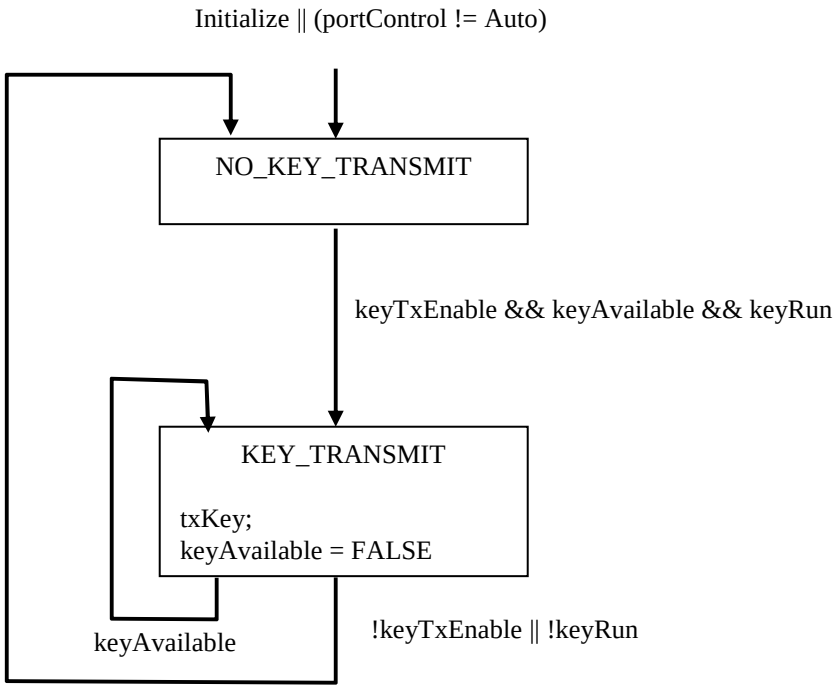
Key Interface with EAP 802.1X & 802.11



Key Interface

- keyAvailable:
  - indicates to the key machine that key material is available to send. No change from previous versions. Set by someone external (e.g. EAP) and cleared by the key machine after the info has been sent. The 4-way machines may or maynot use this variable. It isn't tested by the authenticator machines.
- txKeyEnable:
  - indicates we are using keys. An external management configuration value. No change from previous versions.
- keyRun:
  - A new variable that signals to the key machine to fire-up. It is set true by the authenticator machines after the EAP-Success has been sent and it is cleared by the authenticator machines if it gets reset or abort.
- keyDone:
  - A new variable that signals back from the key machines that keys have been installed or the 4-way handshake has completed successfully and it is ok to test portValid.
- portValid:
  - indicates that keys have been installed and a secured port is now in operation. Set by someone external. No change from previous versions.

Authenticator Key Tx Machine



Supplicant Key Tx Machine