Extensible Authentication Protocol State Machine

Bryan D. Payne and Nick L. Petroni, Jr. *

EAP BOF 3/20/02

1 Introduction

This paper offers a proposed state machine for RFC 2284bis (EAP). There is a state machine for the peer and one for the authenticator. Accompanying each state machine diagram is a description of the notation used. Whenever possible, the same notation has been used in both the peer and authenticator state machines.

Each state machine provides explicit details regarding state transitions that are associated with each message. Therefore, if a message arrives that is not handled by the current state, then the message must be dropped and should be logged. All malformed messages (i.e., messages not complying to the format specified in RFC 2284bis) must be dropped and should be logged.

One type of EAP message has been purposely omitted from the state machines: the notification messages. This was done because notification request messages can be sent from the authenticator to the peer at any point in the state machine. Upon receipt of this request, the peer must send a notification reply. This transaction does not induce any changes to the peer or authenticator state machines.

Additional details about the meaning of each state, and how to handle messages in each state are included in the text accompanying each state machine diagram.

2 Policy

As noted in the title, EAP is extensible in nature. Although originally designed to allow for a series of one-way authentications, current uses and new EAP methods allow for mutual authentication via a single run of the protocol. As a direct result of the extensibility of the protocol, both authenticators and peers are given the opportunity to enforce policies via the protocol. For example, a peer may choose to only allow EAP methods that provide mutual authentication or an authenticator may choose to mandate three successful method authentications out of the five it supports before allowing access. While an advantage of this is more flexibility, one disadvantage is the difficulty involved in formalizing the process. In order to provide for such flexibility, we have introduced the concept of a policy for both the authenticator and the peer. Given the procedures described below, the the authenticator can insure that its policy is met before sending a success message and the peer can insure that any success message it receives comes at an allowable time. The result is a well-defined state machine that maintains the extensibility of the protocol. Of course, the security of the protocol directly depends on the effectiveness of the policy being enforced.

3 EAP Peer State Machine

The following is a diagram of the EAP Peer state machine. Also included is an explanation of the primitives and procedures referenced in the diagram, as well as a clarification of notation.

* {bdpayne,npetroni}@cs.umd.edu

EAP Peer State Machine

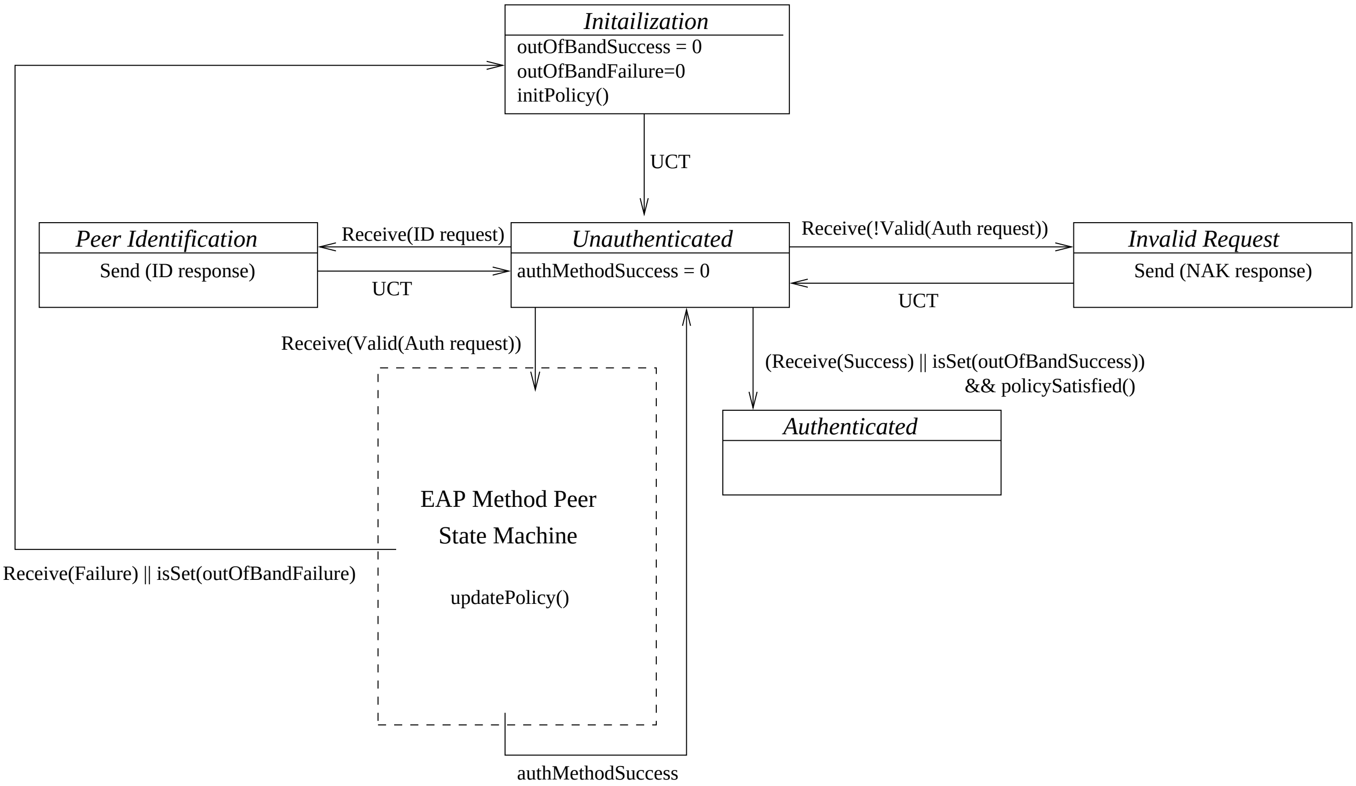

Figure 1: EAP Peer State Machine Diagram

3.1 Variables

- **outOfBandSuccess**. True if an out of band success indication has been received. EAP provides for out of band mechanisms for success and receive.
- **outOfBandFailure**. True if an out of band failure indication has been received.
- **authMethodSuccess**. Set when the current authentication method has been determined to succeed. Should be set by the EAP Method Peer.

3.2 Procedures

- **Send**. Send the EAP Response Packet with the ID field corresponding to the appropriate EAP Request Packet to the lower layer for delivery.
- **Receive**. Determine if a previously unprocessed packet of the specified type has been delevered from the lower level.
- **isSet**. Evaluate the specified variable to determine its truth value.
- **Valid**. Based on the peer's own policy, determine if the specified request type is supported and allowed. Return true if so, false if not.
- **initPolicy**. Reset the variables associated with the Peer's policy to their initial values.
- **updatePolicy**. Generic procedure call relating to the update of any/all necessary variables related to the Peer's policy.
- **policySatisfied**. Determine if the Peer's policy has been satisfied to the point that Success is an allowable transition.

3.3 States

- **INITIALIZATION**
  This state is entered when the peer protocol begins and anytime the protocol resets due to a failure.

- **UNAUTHENTICATED**
  The state of the protocol after being initialized, but before entering a specific method's state machine.

- **PEER IDENTIFICATION**
  State that handles the response to an ID request.

- **INVALID REQUEST**
  State that responds to a request of an invalid type. Invalid is as defined by the Valid procedure and refers to the peer's policy for certain EAP methods (either unsupported or disallowed).

- **EAP METHOD PEER STATE MACHINE**
  Each EAP method has its own state machine specific to that method. Because of the extensible nature of EAP, a particular peer's policy can significantly alter the operation of the protocol and therefore alter the peer state machine. For this reason, the state machine provides for policy and method-specific operation within the general context of the protocol. As shown in Figure 1, the state "EAP Method Authenticator State Machine" is left as a black-box representation of EAP methods. This method-specific state machine is responsible for updating the global variables associated with possible policies held by the larger authenticator state machine. This notion is represented in the Figure with a call to the function updatePolicy.

- **AUTHENTICATED**
  The state reached after a receiving some indication of success, but only after determining such a success occurs at an allowable time.

4 EAP Authenticator State Machine

4.1 Variables

- **idCounter**. Counts the number of times that a client is allowed to return an invalid ID reply. RFC 2284bis recommends allowing the client at least three invalid ID replies to account for user error while typing in an indentity.

- **authMethodFailure**. Set when the current authentication method has been determined to fail. Should be set by the EAP Method Authenticator.

- **authMethodSuccess**. Set when the current authentication method has been determined to succeed. Should be set by the EAP Method Authenticator.

4.2 Constants

- **idCounterMax**. Specifies the max number of times that a peer can send an invalid ID reply before moving to a failure state.

4.3 Procedures

- **Send**. Send the EAP Request Packet to the lower layer for delivery.

- **Receive**. Read a previously unprocessed packet of the specified type from the lower level once it is available.

- **Valid**. Based on the authenticator's policy, determine if the specified reply is what is expected. Return true if so, false if not.

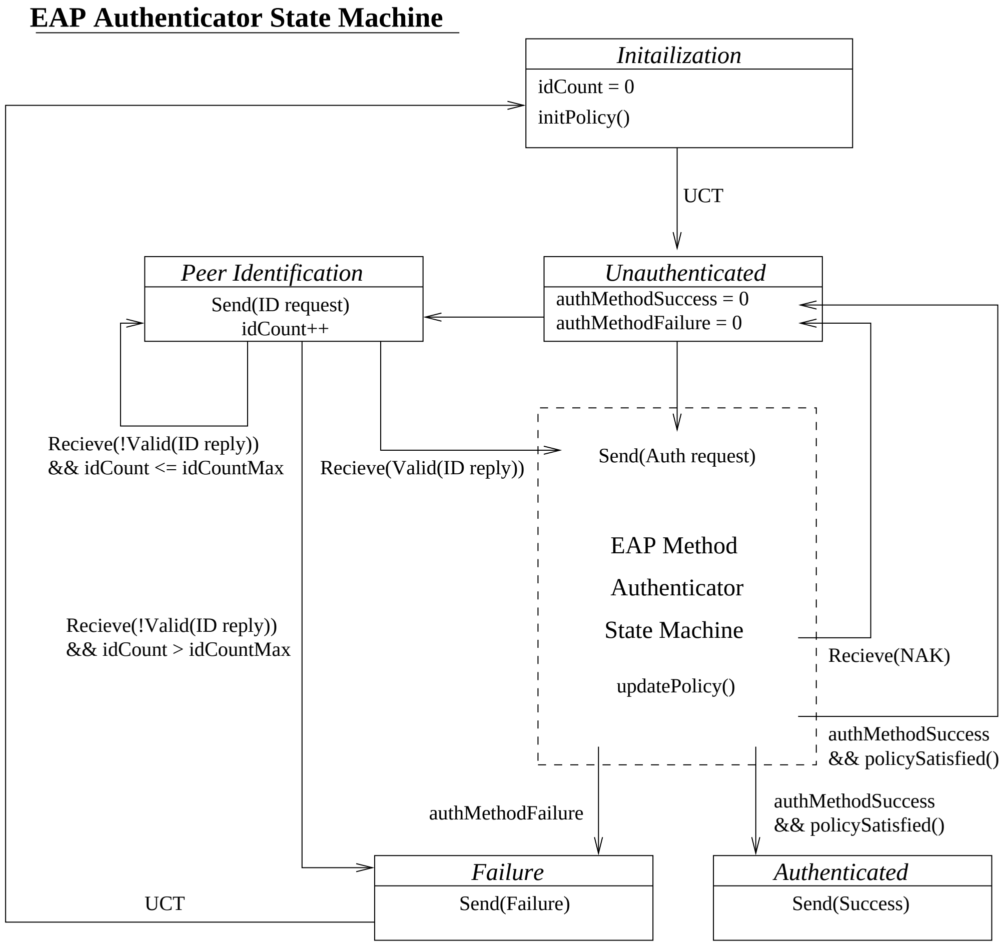

Figure 2: EAP Authenticator State Machine Diagram

- **initPolicy**. Reset the variables associated with the Authenticator's policy to their initial values.
- **updatePolicy**. Generic procedure call relating to the update of any/all necessary variables related to the Authenticator's policy.
- **policySatisfied**. Determine if the Authenticator's policy has been satisfied to the point that Success is an allowable transition.

4.4 States

- **INITIALIZATION**
  This state is entered when the authenticator protocol begins and anytime the protocol resets due to a failure.
- **UNAUTHENTICATED**
  Under this state, the authenticator must decide if it will send an ID Request or proceed directly to the Auth Request. If multiple Auth Requests are required by the authenticator's policy, then this state may be reached multiple times.
- **PEER IDENTIFICATION**
  State that sends an ID request.
- **EAP METHOD AUTHENTICATOR STATE MACHINE**
  Each EAP method has its own state machine specific to that method. Because of the extensible nature of

EAP, a particular authenticator's policy can significantly alter the operation of the protocol and therefore alter the authenticator state machine. For this reason, the state machine provides for policy and method-specific operation within the general context of the protocol. As shown in Figure 2, the state "EAP Method Peer State Machine" is left as a black-box representation of EAP methods. This method-specific state machine is responsible for updating the global variables associated with possible policies held by the larger peer state machine. This notion is represented in the Figure with a call to the function updatePolicy.

- **FAILURE**
  The state reached after sending the failure message. The authenticator should reinitialize the state machine after sending a failure.

- **AUTHENTICATED**
  The state reached after sending a success message.