Formal Analysis of PANA Authentication and Authorisation Protocol

Steven Gordon
Sirindhorn International Institute of Technology
Thammasat University
131 Moo 5, Bangkadi Muang, Pathumthani 12000, Thailand
steve@siit.tu.ac.th

Abstract

The Extensible Authentication Protocol (EAP), which is typically used over wireless LANs and point-to-point links, allows a server to request authentication information from a client. The Protocol for Carrying Authentication for Network Access (PANA) is designed to transport EAP messages over IP networks. This paper presents a formal Coloured Petri net model and analysis of PANA, focusing on the initial Authentication and Authorisation phase. State space analysis of selected configurations reveals a deadlock may occur at the client when the server aborts a PANA authentication session. The analysis also derives a formal definition of the service between PANA and EAP, which is important for verifying that PANA correctly interfaces with EAP, and can later be used for automated testing.

Keywords: *formal methods, Petri nets, authentication, communication protocols, verification*

1. Introduction

The Extensible Authentication Protocol (EAP) [1] is a framework for performing authentication in computer networks (Figure 1. A typical usage scenario, as illustrated in Figure 2, involves a server (known as *authenticator* in EAP) initiating an authentication request to a *peer*. The peer responds to this, and any subsequent requests, until the authenticator determines the procedures to be a success (the peer is authenticated for network access) or failure (the peer is denied access to the network). In practice, a third entity, the *authentication server* may be utilised for storage of credential information. EAP is designed to support different authentication methods (e.g. MD5, TLS, IKE) and to operate over different (non-IP-based) network technologies. For example, a laptop can authenticate with a wireless LAN access point using IEEE 802.11i, or a home PC can authenticate with a dial-in server using EAP over PPP.

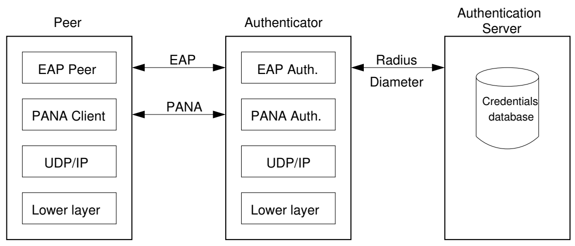

Figure 1. EAP framework

In order to allow EAP to be carried over IP networks, PANA has been developed and released as an IETF RFC in May 2008. The Protocol for Carrying Authentication for Network Access [5] is a lower layer for EAP, and PANA itself uses UDP as a lower layer. In addition to the protocol definition in [5], the PANA Working Group has maintained a state-table model of PANA [4]. Although the state-table model is for informative purposes, combined with the protocol definition, it provides a detailed explanation of the behaviour of PANA. However, as with many distributed protocols, it is important that the PANA specification is accurate and unambiguous. This is particularly important for an authentication protocol, where small errors or an ambiguous specification may lead to implementations with potentially damaging security flaws.

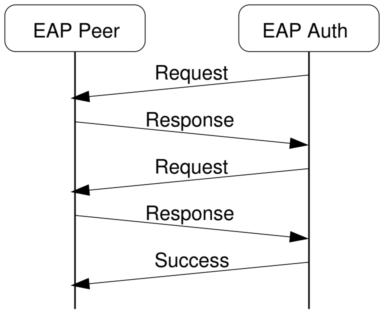

Figure 2. Typical EAP message sequence

Most research on PANA has been applying it to wireless networks [12, 11, 3], especially performance analysis of PANA re-authentication during handovers [2, 6]. Little effort has been directed to the formal analysis of PANA, including security analysis. The overall aim of our research is to verify the design of PANA to ensure a complete and correct specification is available. This paper does not attempt a formal security analysis (from a cryptographic viewpoint) of PANA. In fact, such analysis depends largely on EAP and other authentication methods, as PANA is only a protocol that carries EAP messages.

The contributions of this paper are:

1. A formal model of PANA using Coloured Petri nets (CPNs) [9]. Although a state-table model of PANA exists in [4], the CPN model is executable, allowing for detailed study of PANA operations through simulation, as well as verification of properties through state space analysis.

2. Definition and analysis of the EAP/PANA interface. This interface is partially and informally described in [4, 13]. Through an iterative modelling and analysis process, a formal definition of the EAP/PANA interface has been derived. This is important for the verification of certain properties of PANA, and can also be used for implementation and conformance testing.

3. Verification of functional properties of the PANA Authentication and Authorisation Phase. Selected parts of PANA have been analysed using state space and language analysis techniques, investigating deadlocks, livelocks and correct interfacing with EAP. A potential deadlock in PANA is identified from the analysis.

The remainder of this paper is organised as follows: Section 2 describes PANA and EAP in further detail. Section 3 presents the analysis methodology used. Section 4 overviews the CPN model of PANA. Results from the formal analysis of the PANA Authentication and Authorisation Phase are presented in Section 5. The conclusions and areas of future work are summarised in Section 6.

2. EAP and PANA

2.1. EAP

EAP is a request/response protocol where only a single packet is in-flight at once, i.e. the authenticator cannot send a new request until the response from the previous request is received. The requests contain authentication challenges to the client. EAP assumes the lower layer (in our case, PANA) will provide in-order delivery of packets, however it does not require the lower layer to be reliable, provide security or remove duplicates. A typical scenario, as illustrated in Figure 2, involves one or more EAP Request/Response exchanges (always initiated by the authenticator) followed by a final EAP Success or EAP Failure message, depending on the authentication information supplied by the client.

2.2. PANA

The role of PANA is to transport EAP messages between peer (referred to as PANA Client or PaC) and authenticator (PAA). PANA uses UDP as a transport layer, and hence the service provided to PANA may have packet losses, duplication and re-ordering. An exchange of messages in PANA is a session, which is divided into four phases:

Authentication and Authorisation At the start of a PANA session this phase involves the exchange of EAP messages to perform authentication.

Access Once authentication of the peer is successful, network access is provided. During this phase either PaC or PAA may test for the liveness of the session (which has a limited lifetime).

Re-authentication May be performed to maintain the session liveness.

Termination Either PaC or PAA may terminate a session. If a session isn't terminated gracefully, then a timeout on the PANA session will result in the termination.

PANA communications are implemented as a series of request and answer messages. To explain the Authentication and Authorisation phase consider the example scenario in Figure 3 (which is generated from our CPN model in Section 4). It shows the communication between EAP entity and corresponding PANA entity, as well as between PANA entities across the network.

The PANA session can be initialised by either the PAA or PaC. The methods for each entity learning about the presence of the other is out side of the scope of PANA (e.g. it may be through DHCP). For a PAA-initiated-session, after it discovers the presence of a PaC, it sends an AuthRequest message to start the session (the 'S' flag indicates this message is to start the session). This initial AuthRequest is used to force a restart of the EAP session at the Peer. The PaC responds with a AuthAnswer, which results in the EAP session at the Authenticator restarting.

The EAP Authenticator initiates the authentication with an EAP Request. This triggers the PAA to send an AuthRequest carrying the EAP Request method. Upon receipt of the AuthRequest, the PaC passes the EAP Request method to the EAP Peer and replies with an AuthAnswer. The AuthAnswer messages are acknowledgements to the AuthRequest messages.

The PaC sends the response to the challenge in an AuthRequest (which is also acknowledged by the PAA with a AuthAnswer). This sequence of EAP requests and responses (and AuthRequest and AuthAnswer messages) may repeat until the authentication is complete. Finally the EAP Authenticator will send a Success or Failure method indicating the result of authentication. The EAP Success is shown in Figure 3, which is carried in an AuthRequest with the Complete flag set. Once the AuthAnswer is received by the PAA, both PAA and PaC have the PANA session established and the Access phase is entered. Other relevant details of PANA include (refer [5] for more information):

- 32-bit sequence numbers are used to maintain ordering and perform error detection. The sequence numbers at PAA and PaC are independent. An outgoing request message contains a sequence number, and the corresponding answer message must have the same sequence number.
- Request messages are retransmitted if an answer is not received within a specified time. The session is terminated if too many retransmissions occur.
- PANA messages contain 16 bytes of fixed size header (e.g. flags, message type, sequence number, session identifier) as well as a variable number of Attribute-Value Pairs (AVPs). AVPs include: the actual EAP message; authentication data; session lifetime; and other security related information.
- Optional piggybacking of messages allows either PaC or PAA to send a single PANA message that represents both an answer and a request. For example in Figure 3 *without* piggybacking, PaC sends an AuthAnswer(8) followed by AuthRequest(3). With piggybacking turned on, the PaC could send a single message, AuthRequest(3) which acts as the acknowledgement for AuthRequest(8) received from PAA.

2.3. EAP/PANA Interface

In order to verify if the PANA protocol correctly interacts with EAP, it is necessary to understand the interface between the two layers. The EAP state-machines [13] specify the variables used for communication between EAP and a lower layer. We have summarised this information in Figure 4. As an example, when the EAP Authenticator sends an EAP Request, the eapReq flag will be set to true and the Request method will be included in eapReqData.

In addition to the EAP-defined interface, the PANA state-tables [4] describes its own set of variables and procedures used for communication between PANA and EAP. Through our analysis we have defined a set of service primitives that attempt to unify the interface between EAP and PANA. Table 1 lists the EAP-defined variables, the PANA-defined variables, as well as our service primitives.

Table 1. EAP/PANA interface messages and service primitives

No.	Entity	EAP	PANA	Primitive
1	Peer/PaC	-	AUTH_USER	CAuthUser
2	Peer/PaC	eapRestart	EAP_RESTART	CRestart
3	Peer/PaC	eapReq	EAP_REQUEST	CRequest
4	Peer/PaC	eapResp	EAP_RESPONSE	CResponse
5	Peer/PaC	eapSuccess	EAP_SUCCESS	CSuccess
6	Peer/PaC	eapFail	EAP_FAILURE	CFailure
7	Peer/PaC	-	-	CTimeout
8	Peer/PaC	-	ABORT	CAbort
9	Auth/PAA	-	PAC_FOUND	APacFound
10	Auth/PAA	eapRestart	EAP_RESTART	ARestart
11	Auth/PAA	eapReq	EAP_REQUEST	ARequest
12	Auth/PAA	-	-	AResponse
13	Auth/PAA	eapSuccess	EAP_SUCCESS	ASuccess
14	Auth/PAA	eapFail	EAP_FAILURE	AFailure
15	Auth/PAA	-	EAP_TIMEOUT	ATimeout
16	Auth/PAA	-	ABORT	AAbort

3. Analysis Methodology

Formal modelling and analysis of distributed protocols is important to reduce the chance of erroneous behaviour in implementations. The steps applied in our research are:

1. Formal modelling of the protocol specification. The PANA standard and state-tables specify the protocol in an informal manner. Creating a formal model, in our case using Coloured Petri nets [9], is a first step in identifying possible problems in the protocol design, as well as gaining an in-depth understanding of the protocol behaviour.

2. Simulation of the protocol model. An executable formal model allows for investigation of specific scenarios in the protocol operation. Developing the CPN model with CPNTools [10] and BRITNeY [14] allows sequences of events to be stepped through, and automatic generation of message sequence charts (e.g. Figure 3). This graphical output is useful for analysing expected and unexpected scenarios in depth.

3. Functional property verification from state space analysis. By generating all possible states of the PANA CPN model, properties such as absence of deadlocks, livelocks and redundant events/actions can verified. CPNTools provides software support for generating the state space from the CPN model, and querying the properties.

4. Verification of the protocol against service definition using language analysis. The PANA protocol should interface with EAP correctly—that is, the service provided

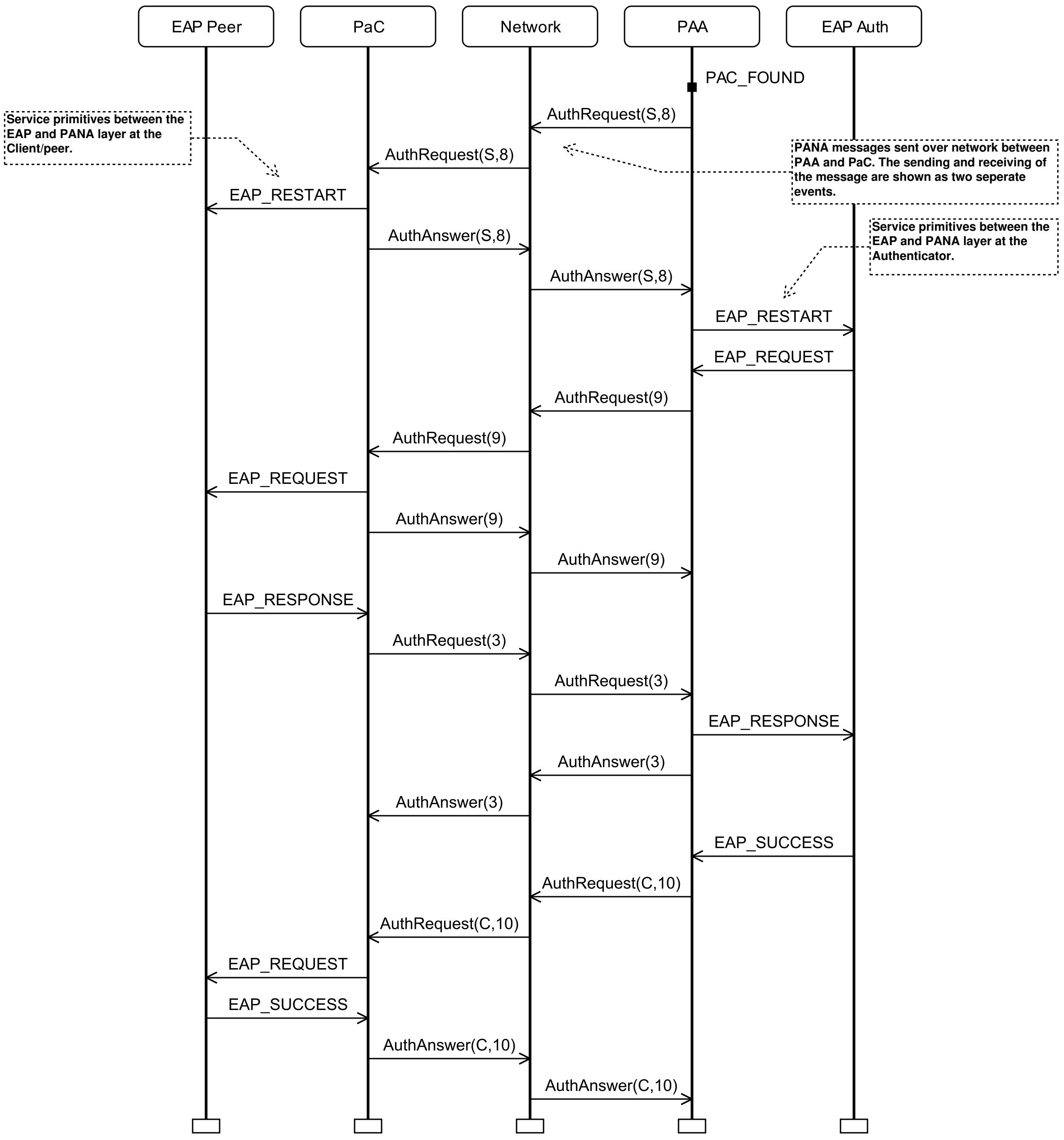

Figure 3. Message sequence chart of PANA Authentication phase

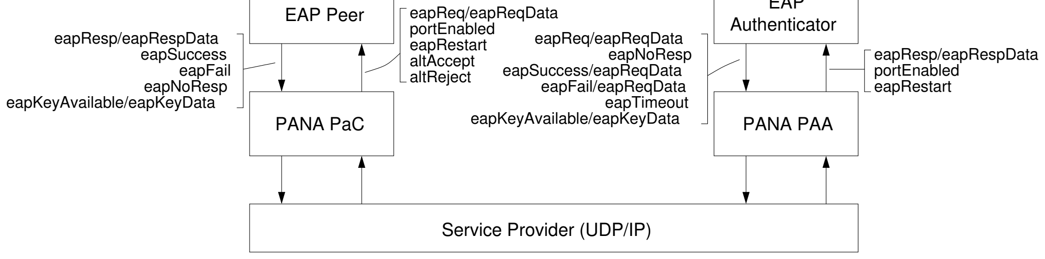

Figure 4. EAP/PANA interface based on [13]

by PANA to the higher layer must be a faithful refinement of the service EAP assumes of the lower layer. The service definition (or language) is the set of methods/primitives for interfacing between layers, and the possible ordering of primitives. As illustrated in Section 2.3, both EAP and PANA separately define a set of interface messages, although not the expected ordering. As a service definition is not immediately available for PANA, we have used our CPN model of PANA to generate a definition. We have defined a unifying set of service primitives (Table 1) and obtained possible orderings from the state space. This is done by treating the state space of PANA as a finite-state automata. The labelled transitions of the FSA are those arcs in the state space that correspond to a service primitive being issued (to or by PANA); all other arcs correspond to epsilon transitions in the FSA. Language analysis is then used to generate the PaC service language, the PAA service language, as well as the entire PANA service language. Currently visual inspection is used to validate the language. The sequence's of primitives as seen by the PaC are generated and manually checked for correctness. Similarly for the primitives seen by the PAA. In the future, comparison of the entire PANA service language with the EAP standard and subsequently verification of PANA against the service definition will be carried out.

4. Coloured Petri Net Model of PANA

A CPN model of PANA has been created based on the state tables in [4]. The model consists of 23 pages, 63 transitions and 7 places. The model focuses on the components of the protocol important for functional verification, i.e. the ordering of exchange of messages. Where possible, abstraction is used so that details of message contents and formats can be omitted. This makes analysis easier, but at the expense of a complete protocol specification. The model is too large to present in its entirety in this paper, and hence only key assumptions, limitations and design decisions are presented. Further details about creating CPN models based on state-tables can be found in [8].

The PANA CPN is hierarchical, with the PaC and PAA modelled on separate pages, and then each state of the PaC/PAA modelled on separate pages. This is achieved using substitution transitions and fusion places. At the highest level (Figure 5(a)) there are two transitions (PaC and PAA) and two places modelling the communication channel between PaC and PAA (and vice versa). Both the PaC and PAA transitions contain detailed models on respective sub-pages. These sub-pages (see Figure 5(b) for the PAA sub-page) contain transitions that model the events at each state, a place to model the current state (and related state information) and other auxiliary places. Each transition on these sub-pages is further decomposed to individual pages which model the events, conditions, actions and next states as presented in the state-tables. In summary, the top level page is a system model, the two medium level pages model the PaC and PAA, and the lowest level pages model the individual entries of the state-tables. This net structure has the advantage of clarity by presenting the model at different levels of abstraction, as well as the ability to validate the CPN against the state-tables in [4].

For a given state, the state-tables in [4] specify: an exit condition, i.e. the conditions that must occur; exit actions, i.e. the actions that will be executed upon the conditions being met; and the exit state, i.e. the next state of the entity. Each entry in a state table is modelled by a single transition in the CPN. Each transition has or may have (the bottom transition in Figure 5(c) is used as an example):

- An input arc from a place containing the current state, and related state information, e.g. sequence numbers, flags. (The example transition is only enabled when PAA is in the A_INITIAL state).
- An input arc from the communication places (Client2Auth, Auth2Client) if the event involves receiving a message. (The example transition is only enabled when a AuthAnswer has been sent by the PaC).
- A guard for the conditions related to the event. (The AuthAnswer must have the Start flag set, not contain EAPPayload, and PAA must be using OptimizedInit).
- An output arc to the communication places if the action involves sending a message. (No message is sent for the example transition).
- An output arc to the place containing state information, where the next state is stored. (The new state is A_WAIT_PAN_OR_PAA for the example transition).

5. Analysis Results

This paper reports results from formal analysis of selected scenarios of PANA Authentication and Authorisation phase. Specifically, the analysis assumes only a single EAP Request is sent by the Authenticator and the retransmission of request messages is not allowed, i.e. the maximum number of retransmissions allowed before a termination (or abort) is 0. For simplicity, four-bit sequence numbers are used (instead of 32-bit), and the initial sequence numbers are randomly set at 3 and 8 for PaC and PAA, respectively. The option of optimising the initiation procedure at the PAA is not analysed.

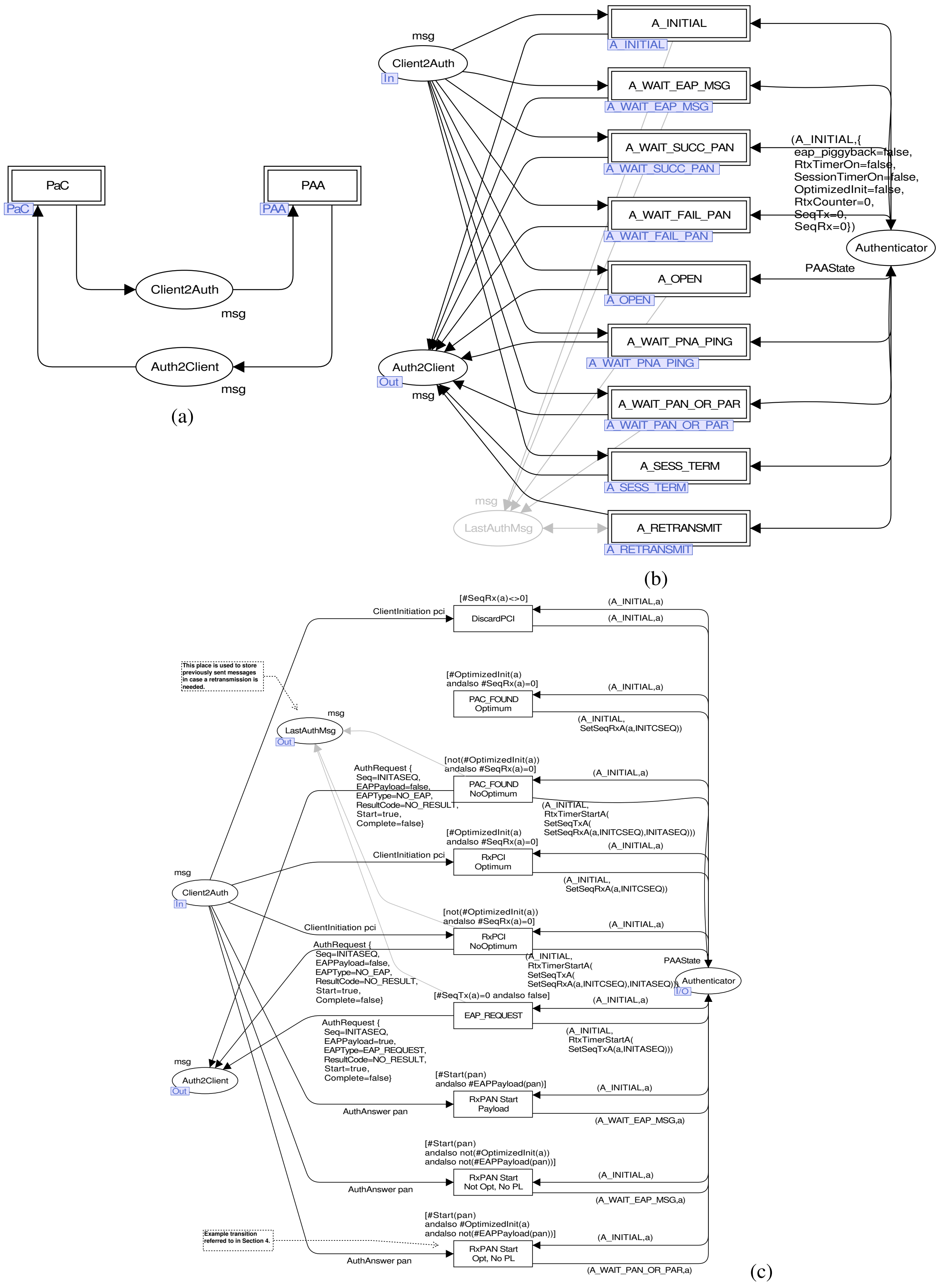
PaC
PAA
Client2Auth
Auth2Client
msg
(a)
A_INITIAL
A_WAIT_EAP_MSG
A_WAIT_SUCC_PAN
A_WAIT_FAIL_PAN
A_OPEN
A_WAIT_PNA_PING
A_WAIT_PAN_OR_PAR
A_SESS_TERM
A_RETRANSMIT
In
Out
LastAuthMsg
Authenticator
PAAState
(A_INITIAL,{ eap_piggyback=false, RtxTimerOn=false, SessionTimerOn=false, OptimizedInit=false, RtxCounter=0, SeqTx=0, SeqRx=0})
(b)
DiscardPCI
[#SeqRx(a)<>0]
ClientInitiation pci
(A_INITIAL,a)
This place is used to store previously sent messages in case a retransmission is needed.
PAC_FOUND Optimum
[#OptimizedInit(a) andalso #SeqRx(a)=0]
(A_INITIAL, SetSeqRxA(a,INITCSEQ))
PAC_FOUND NoOptimum
[not(#OptimizedInit(a)) andalso #SeqRx(a)=0]
AuthRequest { Seq=INITASEQ, EAPPayload=false, EAPType=NO_EAP, ResultCode=NO_RESULT, Start=true, Complete=false}
(A_INITIAL, RtxTimerStartA(SetSeqTxA(SetSeqRxA(a,INITCSEQ),INITASEQ)))
RxPCI Optimum
RxPCI NoOptimum
EAP_REQUEST
[#SeqTx(a)=0 andalso false]
AuthRequest { Seq=INITASEQ, EAPPayload=true, EAPType=EAP_REQUEST, ResultCode=NO_RESULT, Start=true, Complete=false}
(A_INITIAL, RtxTimerStartA(SetSeqTxA(a,INITASEQ)))
I/O
RxPAN Start Payload
[#Start(pan) andalso #EAPPayload(pan)]
AuthAnswer pan
(A_WAIT_EAP_MSG,a)
RxPAN Start Not Opt, No PL
[#Start(pan) andalso not(#OptimizedInit(a)) andalso not(#EAPPayload(pan))]
RxPAN Start Opt, No PL
[#Start(pan) andalso #OptimizedInit(a) andalso not(#EAPPayload(pan))]
Example transition referred to in Section 4.
(A_WAIT_PAN_OR_PAR,a)
(c)

Figure 5. CPN model of: (a) top-level page; (b) PAA; and (c) A_INITIAL state

5.1. State Space Analysis

The state space size for two configurations of the PANA CPN model (piggybacking off and on) is shown in Table 2. Closer inspection of the terminal states is needed to determine whether any are unexpected, i.e. deadlocks. Using CPNTools we classified the terminal states by the state the PaC and PAA finished in. The five classes of terminal states are:

1. C_OPEN and A_OPEN (7 terminal states with piggybacking off; 4 with piggybacking on): both entities have opened a PANA session, i.e. authentication was successful.

2. C_CLOSED and A_CLOSED (396/101): both entities have closed a PANA session, e.g. after failed authentication attempt or abort due to too many retransmissions.

3. C_OPEN and A_CLOSED (20/11): PaC successfully opens a session, however the PAA aborts before opening the session thereby leaving it in the A_CLOSED state. This terminal state is valid: the PaC will enter the Access phase and eventually the PANA session will timeout (and close) after receiving no responses from the PAA.

4. C_CLOSED and A_OPEN (24/7): PAA successfully opens a session, however the PaC aborts (same reasoning as above).

5. C_WAIT_PAA and A_CLOSED (9/30): After the PaC responds to the initial AuthRequest (with Start bit set), it enters the C_WAIT_PAA state. If the PAA aborts before receiving the AuthAnswer from PaC, then PAA enters A_CLOSED. This is an *invalid* terminal state, as the PaC should terminate with either a OPEN or CLOSED session. As the PANA session has not yet been created, there will be no session timeout.

The state space analysis reveals a problem with how aborted sessions are handled in PANA. If one entity aborts the Authentication and Authorisation phase (e.g. due to too many retransmissions), as there are no explicit abort messages sent to the other entity, only a timeout will allow the other entity to proceed. In most cases the timeout occurs, however as illustrated in case 5 above, the PaC may wait indefinitely in the C_WAIT_PAA state. A straightforward method of solving this problem is to introduce an additional timer in the C_WAIT_PAA state. The details and implications of this modification will be considered in future work.

Table 2. State Space Analysis of PANA CPN

Piggyback	States	Arcs	Terminal States
Off	1108	2184	456
On	397	742	153

5.2. Language Analysis

The PANA state space generated in CPNTools was output to a file and the tools FSM, Lextools and Graphviz used to perform the language analysis (further details of these steps are given in [7]). As stated in Section 3, the languages for the PAA and PaC are generated separately (as well as a combined language). Statistics of these languages are given in Table 3, and the PaC language with piggybacking off is shown in Figure 6.

Visual inspection of the PaC language (e.g. Figure 6) and the PAA language (not shown) revealed no obvious errors. For example, the ordering of Requests then Responses is as expected; there are no Requests or Responses after a Success, Failure or Abort; and the PAA only sends a single Request. The sequence of primitives as seen by the PaC shown in Figure 3 are captured in the sequence from states 1, 2, 4, 6, 8 and 7 in Figure 6.

The PaC language and PAA language have approximately 20 sequences of primitives each. Manual inspection is a feasible method to check for errors. However, the combined PANA language is too large for inspection (200 sequences). Methods for analysing the PANA language are considered in future work.

6. Conclusions

PANA is designed to carry authentication information (in the form of EAP messages) over IP networks. Formal analysis of PANA is important to ensure the protocol contains no errors or ambiguities, as they may lead to operational failures, performance bottlenecks or even security flaws during real-world usage. This paper presented the first known formal, executable model of PANA, developed using Coloured Petri nets. The model is useful for testing and validating common scenarios in PANA. State space analysis is performed using the model which allowed properties of PANA such as absence of deadlocks and livelocks to be investigated. Our analysis of the possible terminal states revealed that a deadlock may occur at the PaC if the PAA aborts before receiving the initial AuthAnswer message from the PaC. In addition, the service language (sequence of primitives) between PANA and EAP was derived from the state space. Visual inspection of the interface reveals no obvious errors, however automatic analysis is needed for more rigorous analysis. Additional future work includes extending the CPN analysis to cover the four PANA phases, and detailed analysis of EAP to ensure the PANA service language correctly interfaces with EAP.

Table 3. Language Analysis of PANA CPN

Language	Piggyback	States	Arcs	Final States	Sequences
PANA (PaC and PAA)	Off	39	109	4	200
PaC Only	Off	10	20	4	21
PAA Only	Off	6	15	2	15
PANA (PaC and PAA)	On	37	103	4	170
PaC Only	On	10	20	4	17
PAA Only	On	6	16	2	16

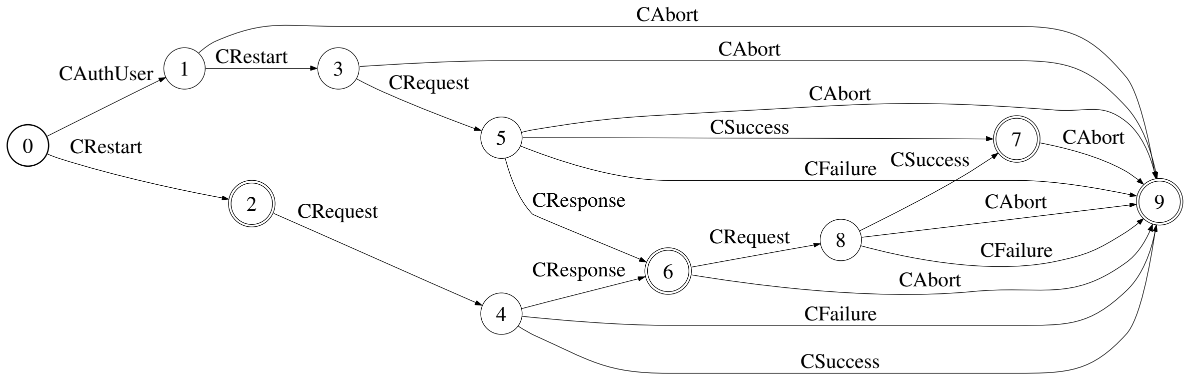

Figure 6. PaC language (piggybacking off)

References

[1] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol. IETF RFC 3748, June 2004.

[2] P. Chamuczynski, O. Alfandi, H. Brosenne, C. Werner, and D. Hogrefe. Enabling pervasiveness by seamless inter-domain handover: Performance study of PANA pre-authentication. In *Proc. 6th IEEE Intl. Conf. Pervasive Computing and Communications*, pages 372–376, Hong Kong, China, 17–21 Mar. 2008.

[3] V. Fajardo, Y. Ohba, and S. Das. Network service provider selection and security bootstrapping using PANA. In *Proc. 2nd Intl. Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities*, Barcelona, Spain, 1–3 Mar. 2006.

[4] V. Fajardo, Y. Ohba, and R. Lopez. State machines for protocol for carrying authentication for network access (PANA). IETF Internet Draft draft-ietf-pana-statemachine-06 (work in progress), Oct. 2007.

[5] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for carrying authentication for network access (PANA). IETF RFC 5191, May 2008.

[6] B. Gaabab, D. Binet, and J.-M. Bonnin. Authentication optimization for seamless handovers. In *Proc. 10th IFIP/IEEE Intl. Symp. Integrated Network Management*, pages 829–832, Munich, Germany, 21–25 May 2007.

[7] S. Gordon. *Verification of the WAP Transaction layer using Coloured Petri nets*. PhD thesis, Institute for Telecommunications Research, University of South Australia, Adelaide, Australia, Nov. 2001.

[8] S. Gordon and J. Billington. Analysing the WAP class 2 Wireless Transaction Protocol using Coloured Petri nets. In *Proc. 21st Intl. Conf. Application and Theory of Petri Nets*, pages 207–226, Aarhus, Denmark, 26–30 June 2000.

[9] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, Berlin, 1997.

[10] K. Jensen, L. M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4):213–254, June 2007.

[11] P. S. Pagliusi and C. J. Mitchell. PANA/GSM authentication for Internet access. In *Proc. Joint 1st Workshop Mobile Future and Symp. Trends in Communications*, pages 146–152, Bratislava, Slovakia, 26–28 Oct. 2003.

[12] T. Tanizawa, M. Goto, V. I. Fajardo, and Y. Ohba. A wireless LAN architecture using PANA for secure network selection. In *Proc. IEEE Intl. Conf. Wireless And Mobile Computing, Networking And Communications*, pages 111–118, Montreal, Canada, 22–24 Aug. 2005.

[13] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba. State machines for Extensible Authentication Protocol (EAP) peer and authenticator. IETF RFC 4137, Aug. 2005.

[14] M. Westergaard. BRITNeY: Basic Real-time Interactive Tool for Net-based animation. URL: http://wiki.daimi.au.dk/britney/, Sept. 2006.