Unreliable Network Re-Authentication Protocol Based on Hybrid Key Using CSP Approach

Sureswaran Ramadass[†], Rahmat Budiarto[†], and Ho Cheah Luan[††],

[†]Universiti Sains Malaysia, Penang, Malaysia
[††]*Dell, Malaysia*

Summary
Network security is becoming increasingly vital in today's fast growing mobile computing environment. Due to constraints in device size and portability, limited processing power, small disk capacity, intermittent network disconnections and frequent switching between network access points have been observed in mobile devices. Thus, protocols for mobile devices must minimize processing overhead to save battery power. In addition, connectivity performance and reliability need to be maintained as in wired environment. In order to achieve this, usage of resource intensive methods for instance public key cryptography and central verification server is foreseen to be reduced. Nonetheless, this poses a security threat to the system. It should be noted that if server security is being easily penetrated especially in the wireless network, the attacker can control the compromised party's side of the communication channel. In this study, improved re-authentication scheme have been focused on, by performing comprehensive analysis to propose an enhanced authentication protocol which supports fast re-authentication for connection or disconnection of client-server system and other possibility of enhancement in terms of message reduction besides providing more secured solutions through CSP (Communicating Sequential Process) protocol modeling.

Key words:
Security, Re-Authentication, Communicating Sequential Process.

1. Introduction

Advancements in wireless networking technology and portable information appliances have provoked mobile computing, in which portable devices users have access to information services through a shared environment, regardless of their physical locations.

Mobile computing is distinguished from conventional wired computing from the mobility feature of users and their computers and the mobile resource constraints such as limitations in wireless bandwidth, battery power and disk capacity. To overcome the constraints, sometimes it is necessary to distort the client-server distinction resource limitations with migration of some client activities to resource-rich servers or by moving some server activities to client-side in the event of disconnections [2].

Mobility implies users require connection from different access points through wireless links and stay connected all time while on move, despite some intermittent disconnections. Re-authenticating users at every connection attempt increases number of connections when users switched between access points. New pre-authentication scheme for fast hand-off has been proposed [6] where stations can authenticate with several access points (APs) during the scanning process. When connection is required, authenticated stations are able to re-associate with APs immediately upon moving into their coverage area, rather than waiting for the authentication exchange. Pre-authentication makes roaming a smoother operation because authentication can take place before it is needed to support an association. However, since this scheme could not predict pattern of client moves in the future, the pre-authentication may be useless in some cases and increases traffic in the wireless link.

Full authentication is time-consuming and iterations of comprehensive cryptographic algorithms with frequent retransmission of messages involved, adversely effects network performance. Since re-authentication latency affects the service quality of multimedia applications, minimizing this factor is essential in order to support real-time multimedia and streaming applications on the wireless IP network in keeping pace with the ever-growing multimedia technology.

Though providing lower level of security, secret key encryption algorithm is faster than public-key methods. However, faster algorithms, such as elliptic curve cryptography, have extended the use of public key cryptography, by increasing security with lower computational load. Hence, the motivation of this study is to achieve high level of security via the use public key cryptography scheme.

Result of this study addresses the possibility of implementation through CSP (Communicating Sequential Process) protocol modeling to perform data collection for protocol benchmarking. Formal verification via third party simulator; CASPER and FDR (Failure-Divergences Refinement checker) is used to validate protocol

Manuscript received November 5, 2007
Manuscript revised November 20, 2007

performance besides debugging vulnerabilities and identifying merits of the protocol

2. Theoretical Consideration

Mobile computing client-server systems can be viewed as a specialized class of distributed systems where clients can disengage from joint distributed operations, move freely in the physical space and reconnect to a possibly different segment of network; a collection of servers at a later stage in order to resume suspended activities.

Mutual Authentication enables communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys before establishing a communication session. Main problem of authenticated key exchange are two issues: confidentiality and timeliness. To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form [9]. A system that is authorized to transmit temporary session keys to principals is called *Key Distribution Center (KDC)*. Each session key is transmitted in encrypted form, using a master key (typically, are distributed by non-cryptographic means) that the key distributor shares with the target principal. *Certificate Authority (CA)* is a trusted third party which certifies identities of other entities; users, clients and servers. When it certifies a user, the certificate authority first seeks verification that the user is not in the *Certificate Revocation List (CRL)*. Upon verification, it grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to validate signatures the certificate authority has made.

2.1 Authentication for mobile client-server

Figure 1 illustrates an example of protocol message exchange level in mobile authentication which involves mobile client, KDC and target application server [5] known as M-PKINIT. In M-PKINIT, Client A only possesses a signing key, but the similar key is usable with public key encryption, allowing both signing and encryption. In this situation, the client generates the session key and encrypts it with the KDC's public key [11].

Normally, KDC will generate a session key and encrypts it with the client's public key. This feature swaps a private key operation for a public key operation on the mobile platform, assuming that the client knows the KDC's public key prior to receiving it as a part of the certification chain.

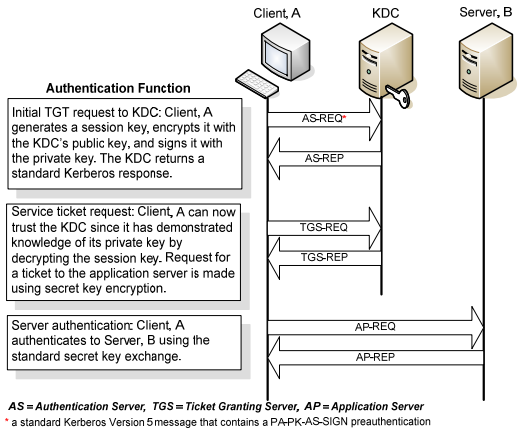

Figure 1. Mobile Public Key Infrastructure authentication with KDC.

Upon establishing a trusted connection, the client will request a reliable ticket to further communicate with the application server through the secret/ session key exchange. The full authentication protocol involves multiple steps which in the event of intermittent disconnections or hand-offs, might incur intensive computations and involves more message transactions.

2.2. Re-Authentication Protocol

Initial authentication schemes focused on connecting mobile users to networks by validating only authorized users shall access its services. Thus, secure establishment of a session key for mutual authentication and non-repudiation is the main aim. Due to performance reasons, the solutions require secret key cryptography usage or the RSA algorithm with short keys, resulted in lower level of security. Despite of this disadvantage, it was necessary to compromise. Multiple re-authentication approaches have been proposed. Upon establishment of a communication session for certain period of time, the client is required to be re-authenticated via faster re-authentication. Low computational intensive with limited security algorithm secret key algorithm was used. In addition, the latency from the re-authentication sessions incurred unnecessary re-authentication overhead.

Other approach includes performing initial authentication followed by less complex re-authentication when client was disconnected. Another improved protocol, known as Neuman-Stubblebine, was introduced based on secret key cryptography where the session key between the communicating nodes was re-used in the re-authentication process until it expired after a given time. This easily comprises to security threat when frequent re-authentication provide large amount of data for analysis by an adversary. Protocol which combines nonce and public key authentication to achieve high level of security with

improvement in subsequent re-authentications, while keeping the mobile client's computational load low by moving the load to server has been introduced.

3. Design of Protocol

There are multiple major approaches to the specification and analysis of authentication protocols. Each method has their strengths and weaknesses. They are addressed as below:

1. Use of existing formal methods to specify and analyze authentication protocols
2. Use of logics of knowledge and belief
3. Use of expert systems and algebraic term-rewriting systems

The above methods are all independent of the cryptographic mechanism used. This is the strength since in producing a protocol specification it is yet to specify a particular implementation mechanism.

3.1 Communicating Sequential Processes (CSP)

Communicating Sequential Processes (CSP) is a notation for describing systems of parallel agents that communicate by passing messages between them. CSP is fundamentally a notation for describing interaction, and can be used to describe a huge range of systems whose only feature in common is that there are different components of distributed computing systems, for example the protocols that can describe interactions between humans and machines. CSP is traditionally written in a fairly mathematical notation, with special symbols representing its operators, and is typeset core like mathematics than a typical programming language [7].

3.2 Trustworthy processes

The typical security protocol involves several agents (often two: an initiator and a responder) and perhaps a server that performs some service such as key generation, translation or certification.

3.2.1 Data type for protocol models

Real implementations of cryptographic protocols uses data which is composed of blocks of bits, and encryption algorithms are computed over the same type of structures and might have properties that are open to exploitation by an intruder. In order to build CSP models, abstract data types are used. Rather than looking at the representation in bytes, and model all the constants like agent names, nonces and keys as symbolic constants and constructions like encryption, hashing as formal symbolic operations over the type.

3.2.2 Modeling an Intruder

The solution is amazingly simple; build a process that can, at any stage, perform any actions that is cryptographically justifiable:

- Eavesdrop and/ or block message that one agent sends to another (including servers)
- Generate any message that can be built on the basis of what the intruder heard, knew initially, or might legitimately made up (nonces and keys), all under the assumptions of what if cryptographically feasible
- Act as agents other than those we explicitly build into network as trustworthy; namely the intruder will have all the keys, that such agents would have

3.2.3 Putting the network together

After having idea of how reliable agents, the server and even the intruder operates. These will be put together into a network that can be used to test the resilience of the protocol.

3.3 Casper

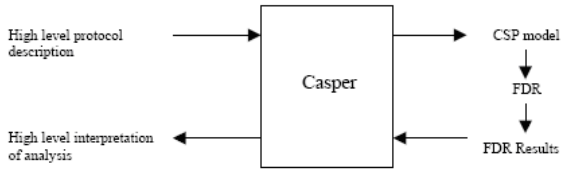

Figure 3. Casper high level overview [1].

Casper is a compiler that converts a high level description of a protocol to CSP code, [1] which is being executed in a model checker for instance *Failure Divergence Refinement (FDR)* to ascertain whether the protocol meets the specified requirements. Casper provides a simple but powerful high level protocol definition language that is very convenient to describe protocols, and diversify high level simulation of intruders' properties. It is role-based model checker with formalized mathematical environment beneath. It interfaces easily and intuitively to its protocol flaw checking tool, FDR.

A few common steps could be followed to reduce the state space required to represent the behaviors of system. There has been approach to create a variety of heuristics to minimize the search space to make the model checking feasible.

3.4 Analyzing the Proposed Protocol Using Casper

The list notation is a compact but simple form widely used. A protocol is formulated as a sequence of messages, with the names of sender and receivers:

Message n a -> b: data

The message content data can be composed of:

- atoms: This may be names, variables and literal constants.
- nonces: A nonce, usually notated like 'n_a', is an unpredictable, freshly generated unique number

The subscript indicates which participant created it for notational convenience.

- Encryption: The term *{data}k* denotes the encryption of data with the key k
- Authentication: *Signk(data)* denotes the signature of data using the key k
- Concatenation: ***a.b*** denotes the concatenation of *a* and *b*, for example the two terms are sent consecutively.

In order to generate a CSP description of the proposed protocol, we first construct a Casper input file. The file defines not only the operation of the protocol, but also the system to be checked. The Casper input description hence, contains two distinct parts:

- A definition on which the protocol operates, describing the messages passed between the agents, the tests performed by the agents, the types of the data items used, the initial knowledge of the agents, a specification of what the protocol is supposed to achieve, and a definition of any algebraic equivalences over the types used.
- A definition of the actual system to be checked, defining the agents taking part in the actual system and the roles they play, the actual data types to be used, and the intruder's abilities.

The type of variables and functions that are used in the protocol definition are defined under the heading '*#Free variables*'. The definition of free variables of the proposed protocol takes the following form. We define the agents, symmetric keys, public keys, and nonce. The functions *PK* and *SK* return an agent's public key and secret key respectively.

#Free variables
a,b : Agent
ka, kb : SessionKey
PK : Agent -> PublicKey
SK : Agent -> SecretKey
na : Nonce

Each agent running in the system will be represented by a CSP process. The names of the CSP processes representing all the agents are defined below. The parameters and variables following keyword *'knows'* define the knowledge that the agent in question is expected to have at the initial of the protocol run. Thus, client *a* of our protocol is expected to know his own identity *a*, the server's identity *b*, the session key *ka*, his nonce *na*, the public key function *PK*, and his secret key *SK(a)*.

#Processes
CLIENT(a,b,ka,na) knows PK, SK(a)
SERVER(b,a,kb) knows PK, SK(b)

The main part of the definition of the protocol is the definition of the sequence of messages in our protocol. Example from Yahalom protocol:

#Protocol description
0. -> a : b
1. a -> b: na
2. b -> s: {a, na, nb}{ServerKey (b)}
3.a s -> a : {b, kab, na, nb}{ServerKey (a)}
3.b s -> b: {a, kab}{ServerKey (b)}
4. a -> b: {nb}{kab}

Two kinds of specifications are currently supported: secrets and authentication. Secret specifies that certain data items should be secret. The first secret specification above may be paraphrased as: *a* thinks that *ka* is a secret that can be known to only itself and *b*. The lines starting Agreement are authentication specifications. The first *Secret* function specifies that *a* is correctly authenticated to *b*, and the two agents agree on the data values *na*.

#Specification
Secret(a, ka, [b])
Secret(b, kb, [a])
Agreement(a,b,[na])

The types of variables to be used in the actual system to be checked are defined in a similar way to the types of free variables.

#Actual variables
A, B, M : Agent
rA, rB, rM : SessionKey
Na, Nm: Nonce
InverseKeys = (rA,rA), (rB,rB), (rM,rM)

Any functions used by the agents in our protocol description have to be defined under the *#Function* heading. Public key functions and secret key functions of agents are shown as follows:

#Functions
symbolic PK, SK

The system definition specifies which agents in the system need to be checked. We consider a system with a single client ***A*** and a single server ***B***. They use *rA, rB, Na*.

#System
CLIENT(A, B, rA, Na)
SERVER(B, A, rB)

Finally, the operation of the intruder is specified by giving his identity, and the set of data values that he initially knows. The following description defines the intruder's identity to be ***M***, and the intruder initially knows all the

agents' identities, his session key *rM*, his nonce *Nm*, public key functions, and his secret key *SK(M)*.
#Intruder Information
Intruder = M
IntruderKnowledge = {A, B, M, PK, rM, Nm, SK(M)}

3.5 Proposed Design

The proposed protocol design is illustrated in Figure 4.

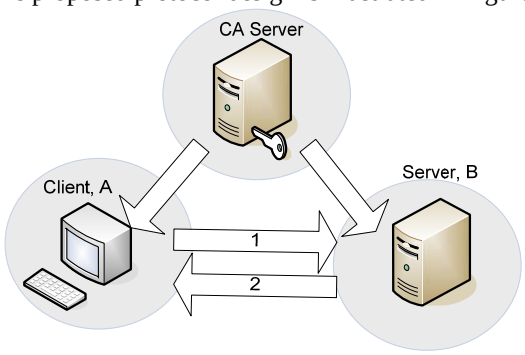

Figure 4. Nonce-based with hybrid keys fast re-authentication protocol.

3.5.1 Initial Authentication

Extract of CSP notation:
#Free variables
a, b : Agent
ka, kb : SessionKey
PKey : Agent -> PublicKey
SKey : Agent -> SecretKey
InverseKeys = (PKey,SKey), (ka,ka), (kb,kb)
na : Nonce

#Processes
INITIATOR (a,b,ka,na) knows PKey, SKey(a)
RESPONDER (b,a,kb) knows PKey, SKey(b)
#Protocol description
0. -> a : b
[a!=b]
1. a -> b : {b,ka}{PKey(b)}, na, {{b,ka}{PKey(b)}, na}{SKey(a)}
[a!=b]
2. b -> a : a, ka(+)kb, {a,ka(+)kb}{SKey(b)}, {na}{kb}

#Specification
Secret(a, ka, [b])
Secret(b, kb, [a])
Agreement(a, b, [na])

1. Client ***A*** initiates the session with a random number generation, nonce ***kA*** as the key-encryption key. ***A*** sends message (1) to server ***B*** containing PB(***B,kA***), ***A***'s nonce ***na***, ***A***'s signature over PB(***B,kA***) and ***na***, together with own certificate certA which contains ***A***'s identity ***A***, ***A***'s ECC encrypted public key ***PA***, plus trusted certificate server CA's signature, ***SCA*** over these SCA(***A,PA***).

2. ***B*** receives message from ***A*** and verifies ***A***'s identity, integrity of message through CA's signature, SCA and ***A***'s signature. ***B*** who kept CA's signature public key then decrypts PB(***B,kA***) using its RSA private key and obtains ***kA***. ***B*** need not keep the client's public key to verify ***A***'s signature, since ***A*** has sent the message containing its public key (certA).

3. ***B*** generates random number ***kB*** as a session key between ***A*** and ***B***.Using Vernam encryption of two keys ***kA*** and ***kB*** and then sends message (2) which consists of certB as ***B***'s identity validity, ***A***'s identity, ***A***'s nonce which are encrypted under the session key ***kB***, signature over ***A*** and encryption of (***kA,kB***) to client ***A***. ***B***'s identity is authenticated.

4. ***A*** then decrypts encrypted (***kA,kB***) by its key-encryption key ***kA*** and gets ***kB*** as a session key with ***B***. ***A*** authenticates the session key ***kB*** explicitly and also verifies that the nonce ***nA*** is the same as it had included in message (1). Matching ***A*** ensures freshness to the session key ***kB***. Therefore, ***A*** and ***B*** share mutually authenticated session key, ***kB***.

ECC encryptions Protocol pseudo (Véronique Cortier)**:**
A, B: principal
f : number -> number
G,Q : public point
k : fresh number
sa,d : private key of A
m,n : number
A chooses a number k.
A computes k.G = (x1,y1), r = x1 mod n and s = k^{-1}(f(m)+dr).
1. A -> B: m,(r,s)
Computation done by ***B*** to check the identity of sender ***A***. The agent ***B*** verifies:
X= (f(m)s_1).G ˙+ (rs_1).Q
=(f(m)(k_1(f(m)+dr))_1).G˙+ (r(k_1(f(m)+dr))_1).(d.G)
= (f(m)(f(m)+dr)_1k).G ˙+(r(f(m)+dr)_1k).(d.G)
= (f(m)(f(m)+dr)_1k + r(f(m)+dr)_1kd).G
= (f(m)(f(m)+dr)_1 + dr(f(m)+dr)_1).(k.G)
= (f(m)+ dr)(f(m)+dr)_1).(k.G)
= 1.(k.G), since s = 0
= k.G

3.5.2 Subsequent Re-authentications

Figure 5 illustrates the subsequent re-authentications process that used in our design.

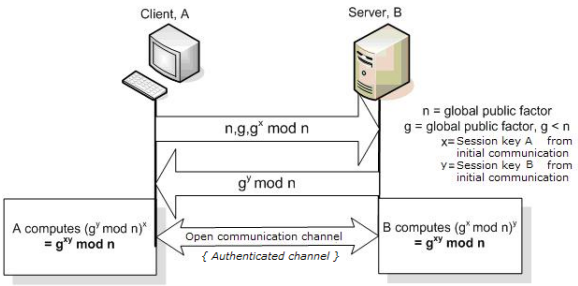

Figure 5. DH reduces message exchange in subsequent re-authentications.

Extract of CSP notation:

#Processes

INITIATOR(A,kab)

RESPONDER(B,kab)

#Protocol description

0. ->A:B

--[A!=B]

1. A->B: Exp(G,kab) % hkeyA

[A!=B and good(hkeyA)]

<key:=Exp(hkeyA,kab)>

2. B->A: Exp(G,kab) % hkeyB

[B!=A and good(hkeyB)]

<key:=Exp(hkeyB,kab)>

3. A->: key

#Channels

authenticated

1. ***A*** computes g^x(mod n), where x is previously secret session key between client ***A*** and ***B*** communications, and send to ***B***. ***A*** have in hand the result of predicted computed return result from ***B*** which is supposed to be g^y(mod n)^x to open communication with ***B***.
2. With the provided value from ***A***, ***B*** computes the common shared value g^x(mod n)^y where y is also the previously secret session key between client ***A*** and ***B*** communications. The data according to protocol step 2 is sent back to ***A***.
3. Upon two parties re-authentication and guarantees of the messages were authentic, both ***A*** and ***B*** will resume communications through the authenticated channel.

This reduces message exchange between client and the server during re-authentications contributing in faster re-authentications. Previous re-authentications require mutual authentications between ***A*** and ***B*** involving public key encryptions.

4. Implementation and Results

This section provides details on the experiment constituted to model NBA-ECC protocol for CSP formal verification under SUSE 9.3 Linux environment. The proposed protocol is based on a symmetric key, nonce-based mutual authentication which enables two participants, the client and server to establish a common secret key using symmetric cryptography and a trusted third party, CA server. This protocol uses the trusted entity in the exchange of public keys.

4.1 Modeling the Protocol

The typical input file for Casper is being described into parts of defining the protocol as a template and defining the actual system to be analyzed by FDR as the instantiation of the template. Protocol extracts in CSP specification is as specified in Section 3.1. The input file .spl is being compiled into CSP script .csp from a more concise description as in Figure 6. The type and consistency checking is carried out to validate the CSP script produced.

As known that the Casper takes an input file, and produces an output file, that contains a CSP description of the system in question. This output is suitable for model checking using the FDR as the extract captured shown in Figure 7.

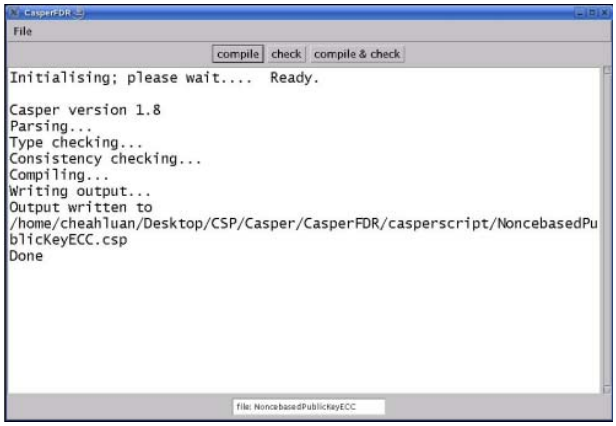

Figure 6. CSP Compilation for Proposed Protocol.

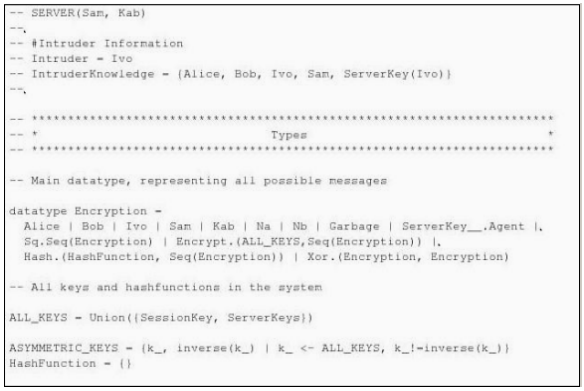

Figure 7. Captured FDR.

Formally, at the end of a successful protocol run each agent (***A,B***) should possess the other's public key and ensure that this key belongs to the other agent. In addition, the participants of the protocol should be satisfied that the session key is a suitable key for communication and that it is fresh. Each agent should also be able to confirm that the other protocol participant possesses the session key (***SK***) and ensure it to be a shared secret. With this, both agents are mutually authenticated to begin actual message communications.

4.2 Analyzing Modeled Protocol

Upon successful CSP compilation, Casper FDR is used to validate whether the system satisfies the specifications and analyze any corresponding attacks as denoted by the system setup. Figure 8 indicates that the proposed protocol is free from system level attack. Casper also generates a refinement assertion for each authentication specification in the input file, for each agents of the appropriate type. The SECRET and AUTH claim generalizes and asserts whether the agent, based on the *#Specification* definition has completed a protocol run with legitimate agents.

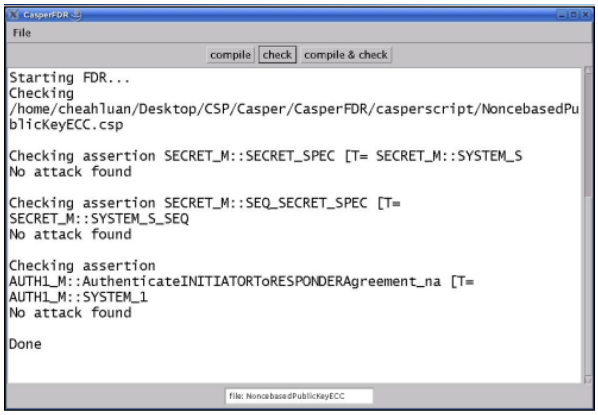

Figure 8: Failure Divergence Refinement Checking.

4.3 Result of Verification

The protocol satisfies the constraint for secrecy of the symmetric session key. A nonce is incorporated to assure the freshness of the received message. The protocol basically needs two message exchanges for complete key establishment; message sent, message received, exponential computations between the client and server. Key management problem does not exist as it eliminates KDC (Key Distribution Centre) and uses both symmetric-key and public-key schemes. It increases efficiency since it uses ECC-based signature whose key size is considerably smaller than other signature schemes.

When this description of protocol in Casper is run with FDR tool, we found some flaws in the protocol. Some security properties below are at risk:

1. The Intruder can capture messages between client-server and delete them. It may result in fail in key agreement of *kb* between two agents causing disconnections in mutual authentication.
2. It is possible to fake certificate's identity easily. Powerful intruder can sniff the certificates. However, as the identities in the certificate have been encrypted with PK and ECC signed by both client-server, it has further prevent intruder from masquerading the identities of initiator (client) and responder (server).

Data collection upon formal verification using Casper has been compiled and summarized from the status as seen in Figure 9, with each rounds of protocol run.

Casper / FDR2 has successfully analyzed and discovered flaws in a wide range of protocols. Among the protocols of the Clark/Jacob library, it has found attacks on 20 protocols previously known to be insecure, as well as attacks on 10 other protocols originally reported as secure [3].

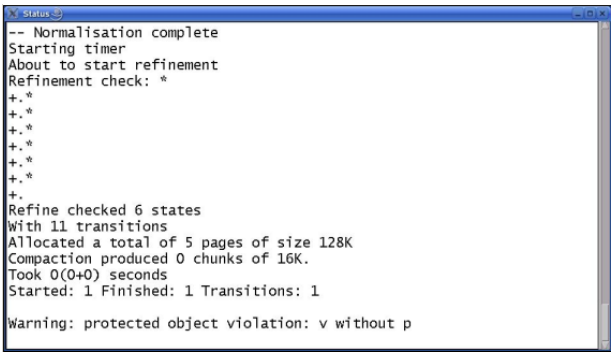

Figure 9: Refinement Check Result Status.

In order to simulate potential system attacks to the proposed protocol, four different system environments has been established by changing the *#System* notations and *#Intruder* knowledge.

#Specification
Secret(a, ka, [b])
Secret(b, kb, [a])
Agreement(a, b, [na])

System 1: Intruder, Ivo knows both ***A*** and ***B*** existence, with respective PKeys.
Result: No attack in secret and agreement scheme.

```
#System
INITIATOR(Alice, Bob, Ka, Na)
RESPONDER(Bob, Alice, Kb)

#Intruder Information
Intruder = Ivo
IntruderKnowledge = {Alice, Bob, Ivo,
PKey, KI, NI, SKey(Ivo)}
```

System 2: Intruder, Ivo listens to message 1 sent over by Initiator, ***A*** and nonce ***A*** is known by Ivo.
Result: No threats of active attack in secret and agreement scheme.

```
#System
INITIATOR(Alice, Ivo, Ka, Na)
RESPONDER(Bob, Alice, Kb)
#Intruder Information
Intruder = Ivo
IntruderKnowledge = {Alice, Bob, Ivo,
PKey, KI, NI, SKey(Ivo)}
```

System 3: Intruder, Ivo masquerades and listens to both ***A*** and ***B*** by eavesdropping authentication messages transacted between both parties.
Result: No threats of active attack in secret and agreement scheme.

```
#System
INITIATOR(Alice, Ivo, Ka, Na)
RESPONDER(Bob, Ivo, Kb)

#Intruder Information
Intruder = Ivo
IntruderKnowledge = {Alice, Bob, Ivo,
PKey, KI, NI, SKey(Ivo)}
```

System 4: Intruder, Ivo successfully attained the session key established by both parties.
Result: Attack found upon key compromised in secret scheme authentications between ***A*** and ***B***.

```
#System
INITIATOR(Alice, Bob, Ka, Na)
RESPONDER(Bob, Alice, Kb)

#Intruder Information
Intruder = Ivo
IntruderKnowledge = {Alice, Bob, Ivo,
PKey, KI, NI, SKey(Ivo), Kb}
```

This experiment firmly concludes that the freshness and confidentiality of the session key established upon mutual authentication against exposure to intruder is crucially essential to avoid key compromise and subsequently communication intrusion by adversaries.

4.4 Result Comparison

The key concept demonstrated by the performance measurements is the benefit of fast re-authentication in a wireless environment. Characteristics of the proposed protocol are analyzed and compared with existing re-authentication protocols in terms of certain factors. It is vital to satisfy the following characteristics [8] as in their proposed key distribution protocol:

- Nature of the authentication; authentication entity and key authentication should be provided.
- Reciprocity of authentication; each of the authentication entity and key authentication may be unilateral or mutual.
- Key freshness where a session key is fresh from the viewpoint of one party, if it can be guaranteed to be new. It is opposed to possibly an old session key being reused through the actions of an adversary.
- Efficiency; considerations include the number of message exchange required between parties, bandwidth required by the message (total number of bits transmitted) and processing complexities for each party.

Table 1 reflects the micro-benchmarks of ECC performance advantage over RSA for different security levels. ECC performance advantage increases even faster than its key-size advantage, as security needs increase.

Table 1. Comparison of ECC and RSA Micro-benchmark. Illustrated from [4].

	ECC-160	RSA-1024	ECC-192	RSA-1536	ECC-224	RSA-2048
Ops/sec	**271.3**	**114.3**	**268.5**	**36.4**	**195.5**	**17.8**
Performance ration	**2.4 : 1**		**7.4 : 1**		**21.4 : 1**	
Key-size ration	**1 : 6.4**		**1 : 8**		**1 : 9.1**	

Our result is summarized in Table 2 which derives the number of total messages and transitions with respective average execution time (second) for each protocol modeled and run in CSP. The table comprises of the comparison of the protocol with other five well-known client-server authentication protocols in the mobile computing environment and also two subsequent authentication pseudos.
The number of messages is one important factor which affects efficiency of the protocol. The number of messages sent or received also affects each entity's processing complexity. The load balance of messages for each entity also has to be considered. A look at the execution times for different protocols provides fair idea of the correlation between the protocol complexity and execution times. The average re-authentication response time of protocol after network disconnections is measured to ensure it offers good scalability and performance, and suitable for practical applications.
The initial authentication key size is small by replacing current initial authentication using Public Key Encryption with ECC. High security level is still maintained. Upon

re-connecting to another server, the ECC-based protocol is foreseen to provide faster re-connections with smaller bandwidth, which is essential to mobile computing environment.

The proposed subsequent authentication protocol is designed to achieve faster re-authentications after temporary disconnections using optimized Diffie-Hellman-like session keys mutual authentications, which reduces message exchange to two steps during both parties verification. Low computational load is placed on the mobile client by performing full authentication protocol at initial connection.

Table 2. Protocols Characteristics Comparison.

Protocol	Encrypt Scheme	Use of Nonce	Number of total messages	Number of transitions (Avg)*	Execution time sec (Avg)	Attacks
Initial Authentication						
Needham Schroeder PK	Symmetric	Yes	7	12,132	1	Yes
Otway Rees	Symmetric	Yes	4	4,920	6	No
KLS	Symmetric	Yes	3	2,366	3	Yes
Yahalom	Symmetric	Yes	4	779	-1	No
Yahalom3	Symmetric	Yes	4	581	4	No
NBA-ECC	Hybrid	Yes	2	11	3	No
Subsequent Authentication						
Neumann Stubblemine	Symmetric	Yes	3	278	-1	No
NBA-non DH	Symmetric	Yes	3	21	-1	No
NBA-DH like	Hybrid	No	3	50	-1	No

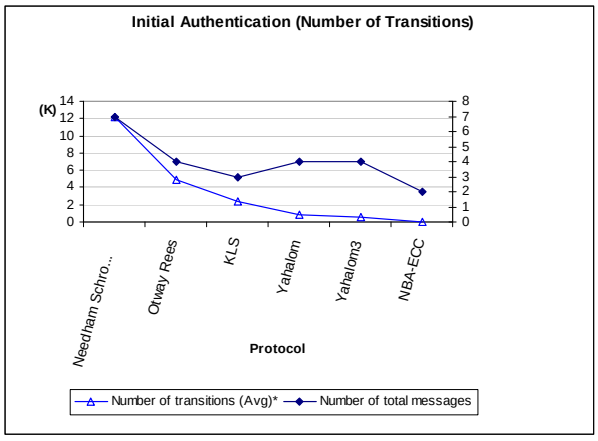

Figure 10. Number of Transitions and Messages Transaction for Initial Authentication.

Figure 10 indicates the low number of transitions for the proposed protocol corresponding to only 3 messages exchange for initial authentication compared to the rest of the protocols. With lesser number of transitions, it will lower risks of exposure to active and passive attacks during mutual authentication.

Although the number of transitions is considerably the lowest among the rest, the proposed protocol average execution time ~3 seconds is ranked mediocre in this experiment as in Figure 11.

For subsequent authentications using Diffie-Hellman concept as proposed, the number of transitions is considerably low compared to the existing re-authentications protocol. The lower the number of messages and transitions, the less risks of attacks of replay, eavesdrop and tracing by intruders on subsequent authentications based on the mutually agreed valid session keys (Figure 12).

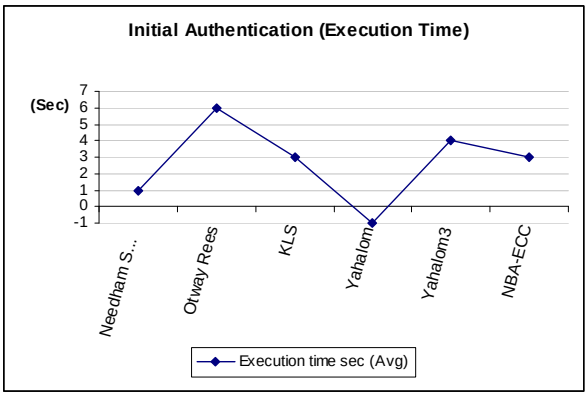

Figure 11: Average Execution Time (Secs).

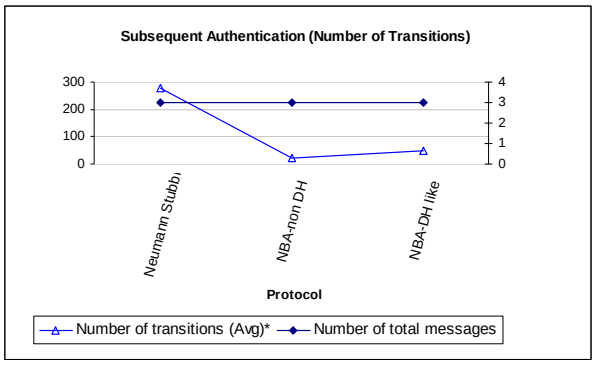

Figure 12: Number of Transitions and Messages Transaction for Subsequent Authentications.

5. Conclusion and Future Work

Wireless devices need to utilize Public Key Cryptography which is widely used to solve many security issues such as authentication and key exchange**.** While RSA algorithms (as one of widely known public-key encryption algorithm) are common in public key applications, there are other ciphers that are also of interest for authentication. The PKINIT draft cited the Diffie-Hellman key establishment algorithm as an obligatory implementation requirement [10]. There has also been significant interest and use of elliptic curve-based encryption algorithms in mobile computing environments.

ECC algorithm is believed to provide comparable security levels to RSA with a smaller key which resulted in a less computationally intensive calculation. The extension of the analysis to substitute Diffie-Hellman or ECC for RSA which potentially reduces the message transmission during authentication in mobile client-server communication has been recommended in [5] for future advancements. There is also an urge to enable broad industry adoption of ECC by promoting ECC standardization within SSL, the dominant security protocol used on the Internet, and contributing ECC technology to OpenSSL and NSS/Mozilla the two most popular open source cryptographic libraries [4].

A number of protocols are designed to provide repeated authentication. These are typically in two stages: (i) an initial authentication, during which a key and ticket (or key certificate) is established; (ii) repeat authentication, which be repeated several times. A ticket is used to re-establish subsequent authentications. With this research experiments, Casper currently has no way of specifying that part (ii) may be repeated arbitrarily multiple times using the ticket established in part (i). However, it is attainable to model the two parts separately or to model a system that runs part (i) followed by a single instance of part (ii).

Future research is intended to extend the Casper syntax to allow the users to specify definitions of the protocol which constitutes repeating authentication phases (possibly with fresh nonces). This is to enable a more complete and accurate analysis of such protocols [3].

Acknowledgments
The authors would like to thanks to USM for the grant support.

References

[1] Lowe, G., Casper: A Compiler for the Analysis of Security Protocols, *Journal of Computer Security*, Volume 6, 1998 53-84

[2] Bresson, E., Chevassut, O., Essiari, A., and Pointcheval, D. Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. *Computer Communications 27*(17 - 2004), 1730-1737.

[3] Donovan, B., Norris, P., and Lowe, G. 2003. Analyzing a Library of Security Protocols using Casper and FDR, *Proc. of the Workshop on Formal Methods and Security Protocols*, Trento,1999.

[4] Gupta, V., Stebila, D., Fung, S. Chang, S., Gura, N. Eberle, H. *Speeding up Secure Web Transactions using Elliptic Curve Cryptography (ECC)*. Sun Microsystems Inc., 1999

[5] Harbitter,A., Menasce', D. A. 2001. The Performance of Public Key-enabled Kerberos Authentication in Mobile Computing Applications. *Proceedings of the ACM Conference on Computer and Communications Security*, 2001, 78-85.

[6] Pack, S., Choi, Y. Pre-Authenticated Fast Hand-off in a Public Wireless LAN Based on IEEE 802.1x Model. *Proceedings of the IFIP TC6 Personal Wireless Communications(PWC2002* (Singapore, October, 2002), 175-182

[7] Ryan, P., Schneider, S. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2001

[8] Sung-Min Lee, Tai-Yun Kim. Two-Pass Hybrid Key Distribution Protocol Based on ECC. *Journal of Information Science and Engineering.* (No. 18), 2002, 125-139.

[9] Tanenbaum, A.S. *Computer Networks*, Prentice Hall, fourth Ed., 2003.

[10] Tung, B. Public Key Cryptography for Initial Authentication in Kerberos. *http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt*. (2001)

[11] William Stallings. *Cryptography and Network Security*. ISBN 0-13-111502-2. Prentice Hall, 2003

Sureswaran Ramadass received B.Sc. degree and Master of Science in Electrical/Computer Engineering from University of Miami, Coral Gables, Florida in 1987 and 1990, respectively, and PhD in Computer Science from Universiti Sains Malaysia in year 2000. Currently, he is an associate professor at School of Computer Sciences, USM. He is the director of National Advanced IPv6 (NAv6) Center, USM. His areas of concentration is Computer Networks and Data Communication, with special focus in: Multimedia Conferencing Systems and Multimedia Networks, Real-Time Network Monitoring, Network Security and IPv6. He is currently the Director of Network Technology Area of APAN.

Rahmat Budiarto received B.Sc. degree from Bandung Institute of Technology in 1986, M.Eng, and Dr.Eng in Computer Science from Nagoya Institute of Technology in 1995 and 1998 respectively. Currently, he is an associate professor at School of Computer Sciences, USM. He is the deputy director of National Advanced IPv6 (NAv6) Center, USM. His research interest includes IPv6, network security, Ethnomathematics and Intelligent Systems. He is chairman of Security Working Group of APAN

Ho Cheah Luan received B.Sc. degree in Computer Science from Malaysia University of Technology(UTM) in 2001, and M.Sc in Computer Science from Universiti Sains Malaysia(USM) in 2006. She joined DELL Inc in 2005 and currently is Commodity Specialist in Dell Global Supply Chain team. Her research interests are in knowledge management, data security and authentication in supply chain management