A New Biometric ID-Based Cryptography Protocol and Security Analysis Using Petri Nets

Dania Aljeaid
School of Science and Technology
Nottingham Trent University
Nottingham, United Kingdom
N0360890@ntu.ac.uk

Xiaoqi Ma
School of Science and Technology
Nottingham Trent University
Nottingham, United Kingdom
xiaoqi.ma@ntu.ac.uk

Caroline Langensiepen
School of Science and Technology
Nottingham Trent University
Nottingham, United Kingdom
caroline.langensiepen@ntu.ac.uk

Abstract—**This paper presents a Petri net (PN) approach to modelling, simulating, and analysing the new protocol we have proposed. This new protocol is an enhanced authentication scheme based on a biometric verification mechanism and identity based cryptography. A formal approach like Petri nets allows one to represent cryptographic protocols. For the sake of simplicity, a complex PN model will not be discussed in this paper until all attacks are demonstrated and the model proved to be secure. This paper shows how Petri nets are used to model, analyse and detect flaws in our new protocol. First, our proposed protocol is modelled without an adversary, and then a generic adversary model is added to examine all possible adversary behaviours. Finally we demonstrate how Petri nets can be used to analyse security threats such as man-in-the-middle attack, reflection attack, and parallel session attack on this protocol.**

***Keywords-** identity-based cryptosystem; biometrics; security analysis; cryptographic protocol; Petri nets.*

I. INTRODUCTION

Due to the unique characteristics possessed by cryptographic protocols, analysis and evaluation tend to be more difficult than normal protocols. Typically cryptographic protocols, also known as security protocols, tend to inhabit a complex environment by utilising various cryptographic mechanisms, such as symmetric and asymmetric encryption, hash functions, timestamps, and digital signature [1]. For this reason, Petri nets offer the opportunity to conduct an in-depth analysis and overcome security vulnerabilities and weaknesses. Moreover, they simplify the modelling of exchange messages between nodes and describe behaviour of authentication and key agreement procedure. A number of researchers have used Petri nets to model and analyse cryptographic protocols [2 -6].

The structure of this paper is organised as follows. In Section 2, we briefly review previous works on Petri nets and our new protocol. In Section 3, we model the client-server trust model using PN. In Section 4, we add the adversary entity to the trust model and simulate various attacks using PN. We then provide a brief discussion on security analysis in Section 5. Finally, the conclusions are given in Section 6.

II. REVIEW OF RELATED WORK

A. Petri Nets

The concept of the Petri net [7] was introduced in 1962 by Carl Adam Petri [8]. Petri nets are graphical diagrammatic tools based on strong mathematical foundations. It is used as a visual communication aid to model concurrency, synchronisation, limited resources, sequentially, mutual exclusion and behaviour in distributed systems [9-11]. A Petri net is defined as a bipartite directed, weighted graph with two types of nodes called places and transitions, linked by directed arcs. In other words, a Petri net must consist of the following components [9-11]:

- A set of places (drawn as circles in the graphical representation), represent conditions and possible states of the system.
- A set of transitions (drawn as rectangles or thick bars), represent a change of state which caused by events or actions
- A set of arcs (drawn as arrows), connecting a place to transition and vice versa.
- Tokens (drawn as black dots), occupy places to represent the truth of the associated condition.

The formal definition of a Petri net is shown in Table 1 [10]. Generally Petri nets focus on specific properties such as liveness, deadlock, livelock, boundedness and safeness [9-11].

Table 1. Formal Definition of a Petri Net

A Petri net is 5-tuple, $PN=(P,T,F,W,M_0)$ where:
$P=\{p_1, p_2,\ldots,p_m\}$ is a finite set of places,
$T=\{t_1,t_2,\ldots,t_n\}$ is a finite set of transitions,
$F \subseteq (P \text{ X } T) \text{ U } (T \text{ X } P)$ is a set of arcs (flow relations),
W: $F \rightarrow \{1, 2, 3,\ldots\}$ is a weight function,
M_0: $P \rightarrow \{0, 1, 2, 3,\ldots.\}$ is the initial marking,
$P \cap T= ø$ and $P \text{ U } T \neq ø$.

A Petri net structure $N=(P, T, F, W)$ without any specific initial marking is denoted by N.

A Petri net with the given initial marking is denoted by (N, M_0).

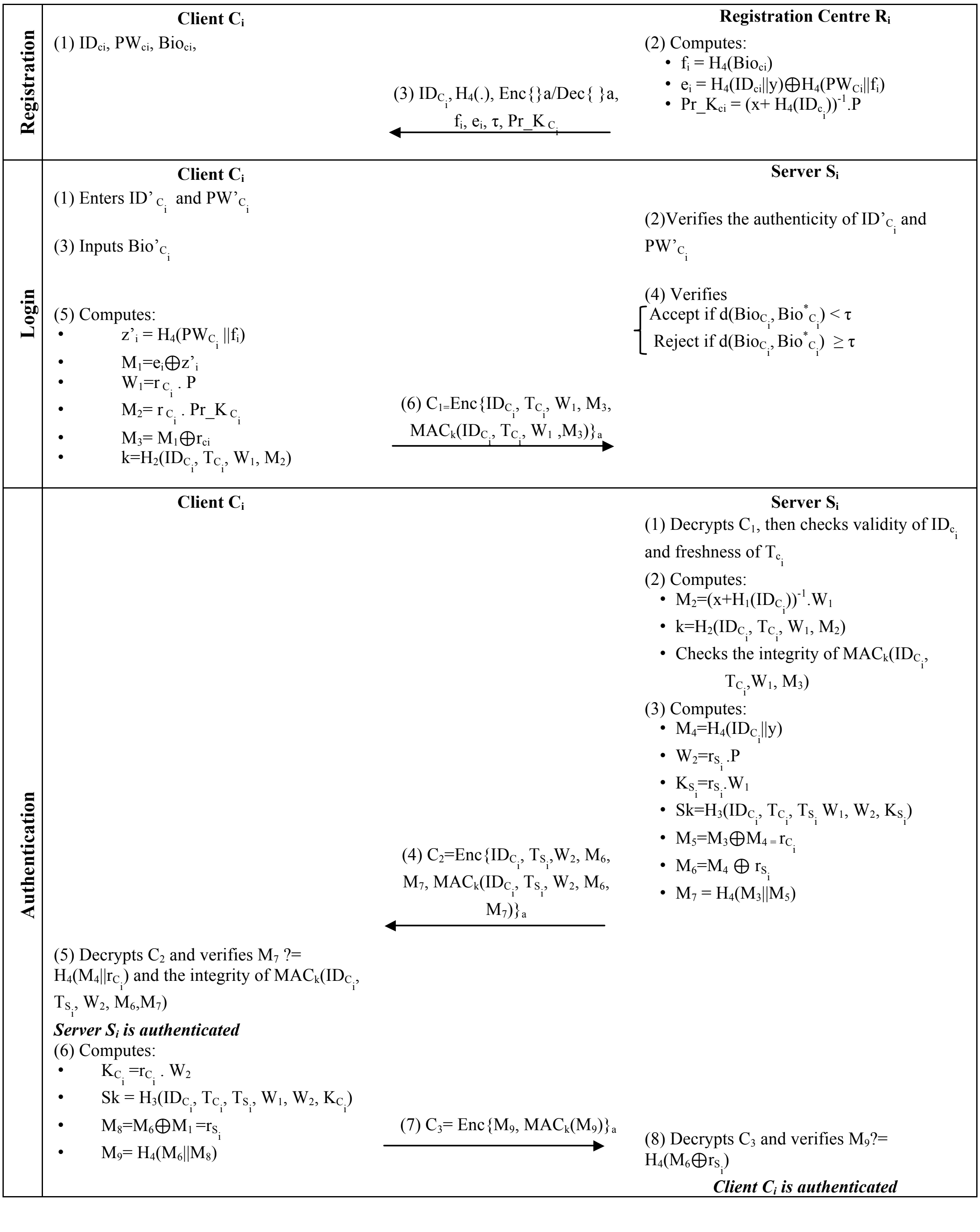
Petri nets are used in this paper to ensure the soundness of the protocol analysis. This approach is a very useful tool for modelling and simulating a range of possible attacks on the proposed protocol. The key features of using Petri nets can be summarised as follows:

1. The ability to model the concurrency of the protocol progress with tokens
2. The ability to model intermediate and final objectives as places

3. The ability to model transitions as commands and inputs

B. Review of Proposed Protocl

In our previous work [12], we have developed a new authentication protocol that allows remote mutual authentication with key agreement. Our new protocol is based on biometric verification and ID-based Cryptograph [13]. Moreover, the new protocol is aimed to initiate secure authentication and communication between the client and server by building a robust mechanism between communicating parties The proposed protocol may be described as a two-factor user authentication mechanism and three-way handshake procedure to establish a reliable connection and ensure secure data sharing. Our new protocol consists of four phases: system initialising phase, registration phase, login phase, and authentication phase. The new protocol is summarised in Fig. 1 and the notations used for the new protocol are summarised in Table 2.



Registration

Client C_i

(1) ID_{ci}, PW_{ci}, Bio_{ci},

Registration Centre R_i

(2) Computes:
- $f_i = H_4(Bio_{ci})$
- $e_i = H_4(ID_{ci}||y) \oplus H_4(PW_{Ci}||f_i)$
- $Pr\_K_{ci} = (x + H_4(ID_{c_i}))^{-1}.P$

(3) ID_{C_i}, $H_4(.)$, Enc{}a/Dec{ }a, f_i, e_i, τ, $Pr\_K_{C_i}$ ←

Login

Client C_i

(1) Enters ID'_{C_i} and PW'_{C_i}

Server S_i

(2)Verifies the authenticity of ID'_{C_i} and PW'_{C_i}

(3) Inputs Bio'_{C_i}

(4) Verifies

$\begin{cases} \text{Accept if } d(Bio_{C_i}, Bio^*_{C_i}) < \tau \\ \text{Reject if } d(Bio_{C_i}, Bio^*_{C_i}) \geq \tau \end{cases}$

(5) Computes:
- $z'_i = H_4(PW_{C_i}||f_i)$
- $M_1 = e_i \oplus z'_i$
- $W_1 = r_{C_i} . P$
- $M_2 = r_{C_i} . Pr\_K_{C_i}$
- $M_3 = M_1 \oplus r_{ci}$
- $k = H_2(ID_{C_i}, T_{C_i}, W_1, M_2)$

(6) $C_1 = Enc\{ID_{C_i}, T_{C_i}, W_1, M_3, MAC_k(ID_{C_i}, T_{C_i}, W_1, M_3)\}_a$ →

Authentication

Client C_i

Server S_i

(1) Decrypts C_1, then checks validity of ID_{c_i} and freshness of T_{c_i}

(2) Computes:
- $M_2 = (x + H_1(ID_{C_i}))^{-1}.W_1$
- $k = H_2(ID_{C_i}, T_{C_i}, W_1, M_2)$
- Checks the integrity of $MAC_k(ID_{C_i}, T_{C_i}, W_1, M_3)$

(3) Computes:
- $M_4 = H_4(ID_{C_i}||y)$
- $W_2 = r_{S_i} .P$
- $K_{S_i} = r_{S_i}.W_1$
- $Sk = H_3(ID_{C_i}, T_{C_i}, T_{S_i} W_1, W_2, K_{S_i})$
- $M_5 = M_3 \oplus M_4 = r_{C_i}$
- $M_6 = M_4 \oplus r_{S_i}$
- $M_7 = H_4(M_3||M_5)$

(4) $C_2 = Enc\{ID_{C_i}, T_{S_i}, W_2, M_6, M_7, MAC_k(ID_{C_i}, T_{S_i}, W_2, M_6, M_7)\}_a$ ←

(5) Decrypts C_2 and verifies M_7 ?= $H_4(M_4||r_{C_i})$ and the integrity of $MAC_k(ID_{C_i}, T_{S_i}, W_2, M_6, M_7)$

Server S_i is authenticated

(6) Computes:
- $K_{C_i} = r_{C_i} . W_2$
- $Sk = H_3(ID_{C_i}, T_{C_i}, T_{S_i}, W_1, W_2, K_{C_i})$
- $M_8 = M_6 \oplus M_1 = r_{S_i}$
- $M_9 = H_4(M_6||M_8)$

(7) $C_3 = Enc\{M_9, MAC_k(M_9)\}_a$ →

(8) Decrypts C_3 and verifies M_9?= $H_4(M_6 \oplus r_{S_i})$

Client C_i is authenticated

Figure 1. The new proposed protocol

TABLE 2. NOTATIONS USED IN THE NEW PROTOCOL

Symbol	Definition
C_i	User/Client /Computer
S_i	Server
R_i	Registration Centre
ID_{S_i}	Identity of Server
ID_{C_i}	Identity of user C
PW_{C_i}	User's password
Bio_{C_i}	Biometric template of C
Pub_K	Public Key
Pr_K	Private Key
$\|\|$	Message concatenation operation
P	A point on elliptic curve E with order n
xP	Denotes point multiplication on elliptic curve
y	A piece of secret information maintained by the server
(x, Pub_K_s)	The server S's Private/Public key pair, where $Pub\_K_s = xP$
r_{C_i} , r_{S_i}	A random number chosen by the C_i and S_i respectively
H(.)	A secure one-way hash function
$MAC_k(m)$	The secure message authentication code of m under the key *k*
$\oplus$	XOR operation

We have examined and validated the behaviour of the proposed protocol by using finite-state machines and Petri nets [14]. The following steps explain the methodology to model the proposed protocol with Petri nets:

1) Build a PN trust model of the trust relationship using TAPAAL [15] simulation and verification software. The following steps are necessary for the process of modelling:
    (a) Define the places and transitions and declare their functionalities
    (b) Implement a token passing scheme once the initial marking is set
    (c) Assess the model's behaviour by examining reachability, boundedness, and liveness
    (d) Validate the model using simulation

2) Add the adversary model. This step involves the following:
    (a) Extend the original model and define places and transitions for the adversary entities
    (b) Implement the token-passing scheme with the adversary
    (c) Model different attack and identify any insecure behaviour

III. Client-Server Trust Modelled via PN

The trust model is a notation for determining whom the organisations should trust with its assets. For example, organisations usually verify the applicants' resumes and references, and conduct background and history checks before trusting their employees. Once they are employed, they will be issued photo ID badges and parking permits. In contrast to the real world, it is challenging in the virtual world to identify individuals who are trusted and those who are not. A trust relationship between a client and a server can be obtained in different practices. Some systems use the traditional way that relies on passwords and digital certificates. Sometimes it may involve a trusted third party to operate the authentication and validation, such as the Kerberos login protocol [1], while other systems deploy biometric automated verification systems to recognise trusted users.

In the proposed trust model, the client-server trust relationship is initiated during the registration phase. First, the client submits his/her ID, password (PW_{C_i}), and biometric data (Bio_{C_i}). Then the server will issue in return a corresponding private key ($Pr\_K_{C_i}$), secret key (*a*) for the symmetric encryption, and τ predetermined threshold for biometric verification. The assumption for this model is that the client and server are trustable entities, and they never cheat. Timed-arc Petri Nets are used to model the new protocol. The trust model consists of two Petri net entities: one for the client ***C*** and the other for the server ***S***. The protocol entities are derived from the protocol description in [12]. The assumption made for this model is that each legitimate participant is honest, i.e. behaves according to the protocol rules. The Petri net model in Fig. 2 represents the trust model for the proposed protocol. The definitions of the places and transitions used in this model are illustrated in Table 3 and Table 4, respectively.

Table 3. DEFINITIONS OF PLACES FOR THE TRUST MODEL

Place	Definition	Place	Definition
P_1	Client random number	P_{14}	Encrypted SYN/ACK
P_2	Client timestamp	P_{15}	Decrypted SYN/ACK
P_3	SYN request	P_{16}	Verification message
P_4	Login request	P_{17}	Rejected request
P_5	Encrypted login request	P_{18}	Accept request – Server is authenticated
P_6	Decrypted login req.	P_{19}	Session key
P_7	Verification message	P_{20}	ACK
P_8	Rejected request	P_{21}	Encrypted ACK
P_9	Accepted request	P_{22}	Decrypted ACK
P_{10}	Server random number	P_{23}	Verification message
P_{11}	Server timestamp	P_{24}	Rejected request
P_{12}	Session Key	P_{25}	Accept request – Client is authenticated
P_{13}	SYN/ACK		

Table 4. DEFINITIONS OF TRANSITIONS FOR TRUST MODEL

Trans.	Definition	Trans.	Definition
T_1	Compute login request + SYN	T_{10}	Split the packet and verify
T_2	Encrypt	T_{11}	Drop the packet
T_3	Decrypt	T_{12}	Accept
T_4	Split the packet and verify	T_{13}	Compute ACK and session key
T_5	Drop the request	T_{14}	Encrypt ACK
T_6	Accept	T_{15}	Decrypt ACK
T_7	Compute SYN/ACK and session key	T_{16}	Split the packet and verify
T_8	Encrypt SYN/ACK	T_{17}	Drop the packet
T_9	Decrypt SYN/ACK	T_{18}	Accept

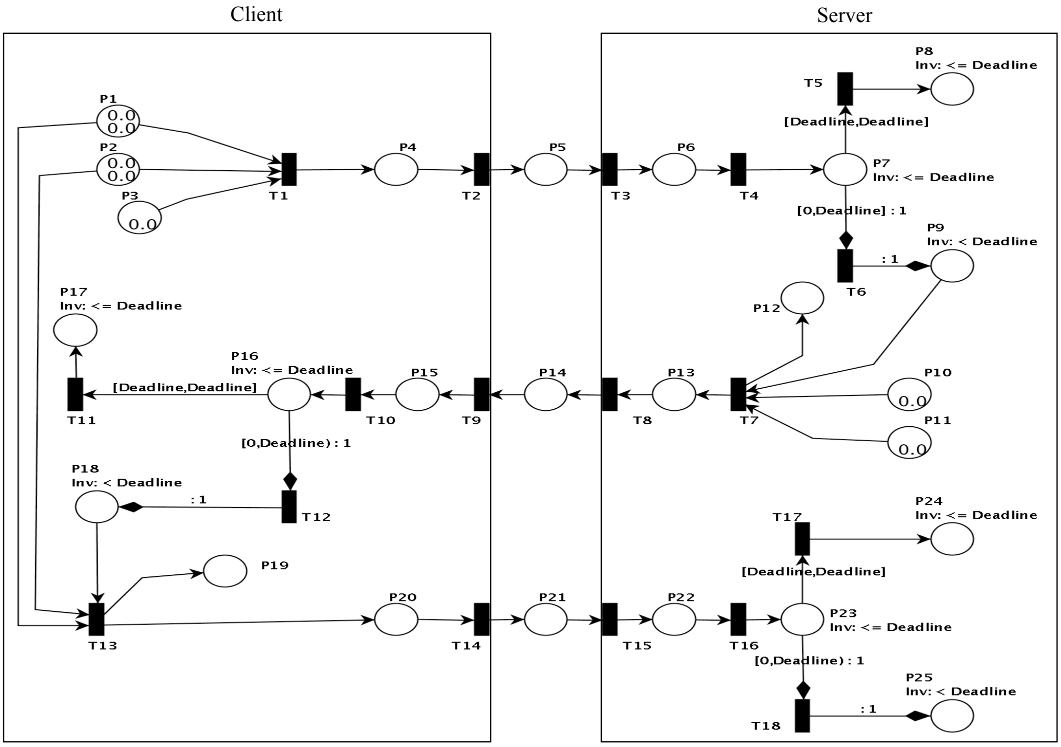


Figure 2. The client-server trust model

In the trust model, the *channels* between ***C*** and ***S*** are depicted by interconnected arcs, which are attached to places. The exchange messages procedure is represented by tokens. *Places* represent storage for requests, messages, ciphers, or session keys. *Transitions* in the model describe particular functions or procedures, which may be performed while in an execution state. For example, the following events produce a new state: encryption, decryption, verification, and computations. *Tokens* are modelled in PN as shown in Fig. 2 to represent the key agreement and message exchange between the client and server. During simulation, the token firing rule imitates the three-way handshake procedure. The structure of *a place linked to a transition* represents a segment of serial processes performed by the entity to fulfil its role in the protocol run. For instance, the transition T_1 in Fig. 2 consumes three tokens from P_1, P_2, and P_3 to calculate the login request. The PN trust model represents a three-way handshake producer between ***C*** and ***S***. It allows both ***C*** and ***S*** to agree on a shared session key over an insecure channel. The steps of protocol analysis for PN trust model are described as follows:

- At first, the protocol is initiated by a client. The client entity of the PN trust model generates a random value (P_1), Timestamp (P_2), SYN request (P_3) to compute the login request (P_4) within a certain period of time. ***C*** sends the encrypted request (P_5) to ***S***.
- Upon receiving the request, ***S*** will check the age of the token. Note that, computing and sending the request to ***S*** takes some units of time. ***S*** will drop the request if the time processing exceeds the deadline. This is guaranteed by the use of transport arcs that preserve the age of the tokens and the corresponding invariants.
- In the second message of the handshake, the server entity generates a random value (P_{10}), timestamp (P_{11}) to compute the session key (P_{12}), and SYN/ACK request (P_{13}). Then S sends the encrypted SYN/ACK (P_{14}) to ***C***.
- Upon receiving SYN/ACK, ***C*** checks the token age and computes the session key (P_{19}). At this stage, ***C*** authenticates ***S*** and sends an enciphered ACK (P_{21}) to ***S***.
- Finally, the server entity checks the token age and authenticates ***C***.

IV. Trust Model with Adversary Modelled via PN

The purpose of this analysis is to find weaknesses and flaws in the proposed protocol. It is essential to examine the behaviour of the protocol with the presence of a malicious adversary. An adversary entity can be a hacker, a malicious insider, a disgruntled employee, a terrorist, organised crime, or competitors.

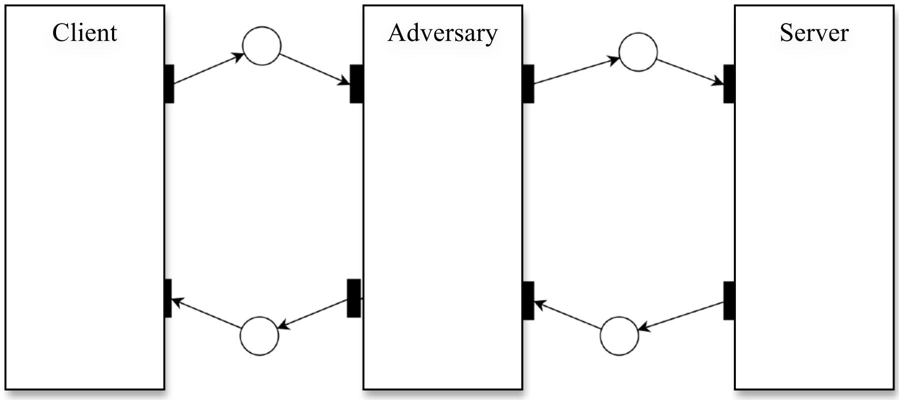


Figure 3. High-level view of adversary entity attacking the protocol

The worst-case scenario would be if attackers obtained illegitimate access to the target system. They could install malicious software, like a rootkit, to remove or modify data. This act of unauthorised access could lead to privilege escalation and allow the attacker to gain elevated entry to resources that are meant to be protected from other application users. Moreover, faulty protocols may allow an attacker to compromise other machines in the network to act as zombie computers to launch denial-of-service attacks.

PN modelling is capable of mapping out how messages flow throughout the protocol with an adversary. A high-level view of the adversary model with information flow is shown in Fig. 3.

The adversary entity is composed of processes, each designed for a specific function in the protocol. Each process models the adversary's possible actions to capture tokens. It can intercept messages from the channel, alter them, and pass them to the target source.

Conceptually, the adversary entity is nondeterministic, in that it may perform different possible actions under different client identities at a given time to ultimately compromise the target system. The following assumptions are considered for the adversary model:

1) The adversary can eavesdrop, intercept, and store messages. It may block or pass any of these messages. Additionally, it may construct forged messages from captured data and inject them into the channel.
2) The adversary has zero knowledge such that it does not possess any elements of messages transmitted between the legitimate nodes but it can learn by observing the traffic.
3) The traffic between client and server is not encrypted.

The main goal of the adversary model is to examine the protocol behaviour with the presence of an adversary while modelling attacks. In the adversary model (attack model), the description of client and server entities is similar to the trust model descried in section 3. For adversary entity, *places* represent an adversary database, which store, control, knowledge and accumulate all the intercepted messages. *Transitions* represent a set of input events and commands the adversary may perform to launch an attack. The *input token* in the adversary entity indicates that the message has been captured. The token movement from place to place through the directed arcs indicates the progress of an attack. To distinguish a genuine traffic from forged traffic, the grave symbol ` is used to indicate that the variable could be modified. For example, if the adversary intercepts the message [A, B, C], the output message would be [A`, B`, C`], which means the message has been manipulated by the adversary.

A. Analysis of Man-in-the-Middle Attack

After adding an adversary entity to the model, it can be noticed that there is the possibility of a man-in-the-middle between the two entities ***C*** and ***S***. An active adversary ***A*** can intercept the communication line between a legitimate client and a trusted server as well as manipulate the protocol by using some means to successfully masquerade either as server or client. The attack model in Fig. 4 represents the man-in-the-middle attack for the proposed protocol. The definitions of the places and transitions used in this model are illustrated in Table 5 and Table 6, respectively.

Table 5. DEFINITIONS OF TRANSITIONS - MAN-IN-THE-MIDDLE ATTACK MODEL

Trans.	Definition	Trans.	Definition
T_1	Compute login request + SYN	T_{13}	Send forge SYN/ACK
		T_{14}	Receive forge SYN/ACK
T_2	Send MSG	T_{15}	Split the packet and verify
T_3	Intercept MSG	T_{16}	Drop the request
T_4	Duplicate MSG	T_{17}	Accept
T_5	Send forge MSG	T_{18}	Compute ACK and session key
T_6	Received Forge MSG		
T_7	Split the packet and verify	T_{19}	Send ACK
		T_{20}	Intercept MSG
T_8	Drop the request	T_{21}	Send forge ACK
T_9	Accept	T_{22}	Receive forge ACK
T_{10}	Compute SYN/ACK and session key	T_{23}	Split the packet and verify
		T_{24}	Drop the request
T_{11}	Send SYN/ACK	T_{25}	Accept
T_{12}	Intercept MSG		

Table 6. DEFINITIONS OF PLACES - THE MAN-IN-THE-MIDDLLE MODEL

Place	Definition	Place	Definition
P_1	Client random number	P_{22}	Sent SYN/ACK for A
P_2	Client timestamp	P_{23}	Sent SYN/ACK for C
P_3	SYN request	P_{24}	Received SYN/ACK for C
P_4	Login request	P_{25}	Received SYN/ACK for A
P_5	Sent request	P_{26}	Sent forge SYN/ACK to C
P_6	Intercepted MSG	P_{27}	Received forge SYN/ACK
P_7	Forge MSG A	P_{28}	Verification message
P_8	Forge MSG C	P_{29}	Rejected request
P_9	Sent forge MSG A	P_{30}	Accept request – A is authenticated
P_{10}	Sent forge MSG C		
P_{11}	Received forge MSG A	P_{31}	Session key
P_{12}	Received forge MSG A	P_{32}	ACK
P_{13}	Verification message	P_{33}	Sent ACK
P_{14}	Rejected request	P_{34}	Intercepted ACK
P_{15}	Accepted request	P_{35}	Forge ACK
P_{16}	Server random number	P_{36}	Received forge ACK
P_{17}	Server timestamp	P_{37}	Verification message
P_{18}	A Session Key	P_{38}	Rejected request
P_{19}	C Session key	P_{39}	Accept request – A is authenticated
P_{20}	SYN/ACK for A		
P_{21}	SYN/ACK for C		

According to Fig. 4 the man-in-the-middle attack proceeds as follows:

- In the login phase, when the client ***C*** initiates and sends the login request (*P_4*) to the server ***S***, an adversary ***A*** may intercept the login message. Transition T_3 represents the initial phase of the attack. ***A*** can duplicate the login message and then start two sessions with ***S*** by sending two copies of request: $P_7 = P_8 = [ID`_C, T`_C, W`_1, M`_3, MAC`_k(ID_C, T_C, W_1 ,M_3)]$ to ***S***.
- Upon receiving (*P_{11}*) and (*P_{12}*), ***S*** generates two random numbers and two timestamps and computes the following:
  - Two session keys (*P_{18}*, *P_{19}*) for ***A*** and ***C***, respectively
  - Two SYN/ACK messages (*P_{20}*, *P_{21}*) for ***A*** and ***C***, respectively

  Then, ***S*** sends the messages (*P_{22}*, *P_{23}*) for the two sessions respectively.
- In the meantime, ***A*** captures (*P_{22}*, *P_{23}*) and sends a forged message (*P_{25}*) to ***C***.
- After receiving the (*P_{27}*), ***C*** verifies it, which in this case is a genuine request $[ID_C, T_S, W_2, M_6, M_7, MAC_k(ID_C, T_S, W_2, M_6, M_7)]$. Consequently, ***C*** authenticates ***A*** masquerading as ***S***. Then ***C*** computes the shared session key (*P_{31}*) and sends ACK (*P_{32}*) to ***S***.

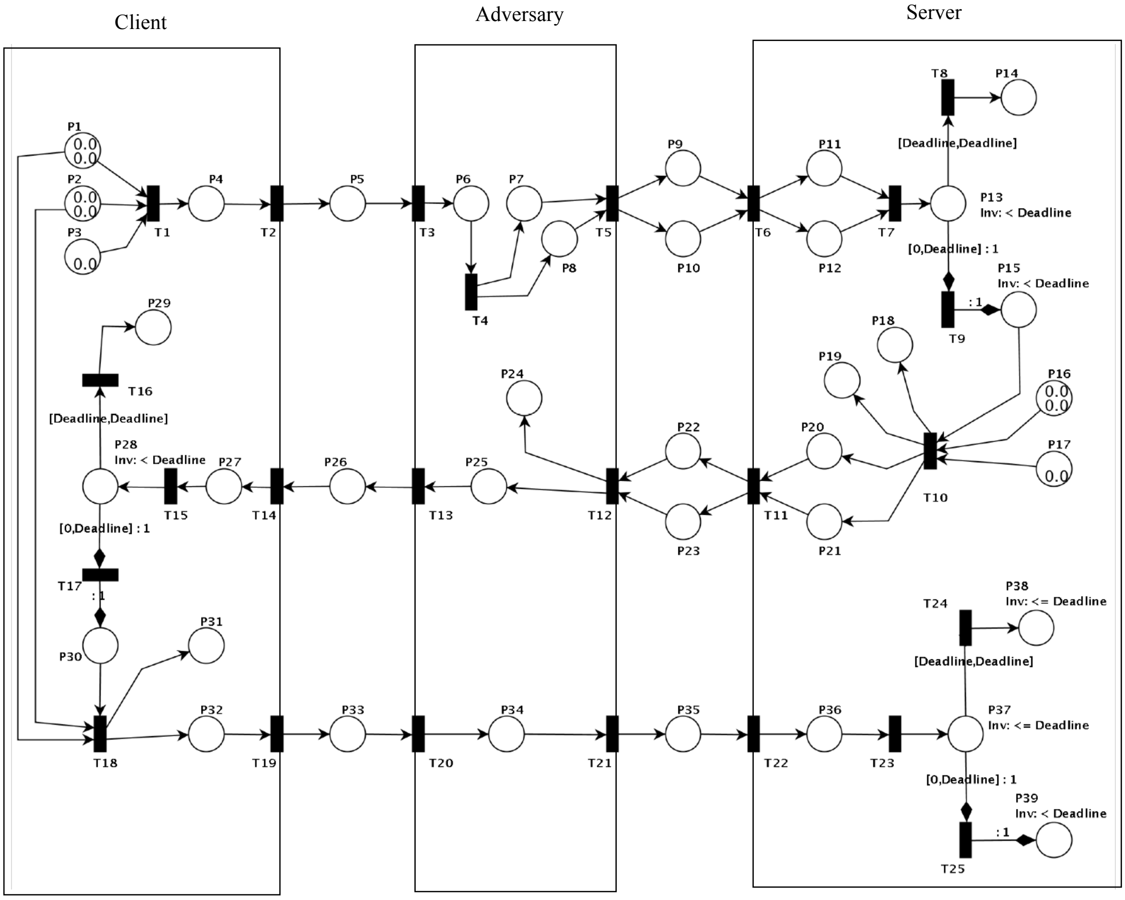


Figure 4. Modelling man-in-the-middle attack

- ***A*** intercepts (P_{32}) and forwards it to ***S***.
- After receiving (P_{36}), ***S*** verifies ACK= $H_4(M_6 \oplus r_S)$. Thus, ***A*** is successfully authenticated by ***S*** masquerading ***C***.

By analysing the protocol, without encrypting the traffic, the proposed protocol is prone to man-in-the-middle attack. The adversary has the ability to control the negotiation between the client and the server. In fact, the adversary can clearly modify, substitute or delete all subsequent messages. It is obvious that both the client and the server have established a bogus session with the adversary.

B. *Analysis of Reflection Attack*

The reflection attack consists of two parties. The adversary in this model is masquerading as the server. In this PN model, *places* represent either input or output of protocol run. *Transitions* are used to explicit the client and adversary actions. *Tokens* indicate the progress of the attack. Fig. 4 describes the execution of a reflection attack for the proposed protocol with presence of the client and adversary. The definitions of the places and the transitions used in this model are illustrated in Table 7 and Table 8, respectively.

Table 7. DEFINITIONS OF PLACES - THE REFLECTION ATTACK MODEL

Place	Definition	Place	Definition
P_1	Client random number	P_9	Received forge SYN/ACK
P_2	Client timestamp	P_{10}	Verification message
P_3	SYN request	P_{11}	Rejected request
P_4	Login request	P_{12}	Accepted request
P_5	Sent request	P_{13}	Session key
P_6	Intercepted MSG	P_{14}	ACK
P_7	Sent forge SYN/ACK	P_{15}	Sent ACK
P_8	Received forge SYN/ACK	P_{16}	Received ACK

Table 8. DEFINITIONS OF TRANSITIONS - REFLECTION ATTACK MODEL

Trans.	Definition	Trans.	Definition
T_1	Compute login request and SYN	T_7	Split the packet and verify
		T_8	Drop the request
T_2	Send MSG	T_9	Accept the request
T_3	Intercept MSG	T_{10}	Compute ACK and session key
T_4	Fabricate SYN/ACK		
T_5	Send fake SYN/ACK	T_{11}	Send ACK
T_6	Received fake SYN/ACK	T_{12}	Receive ACK

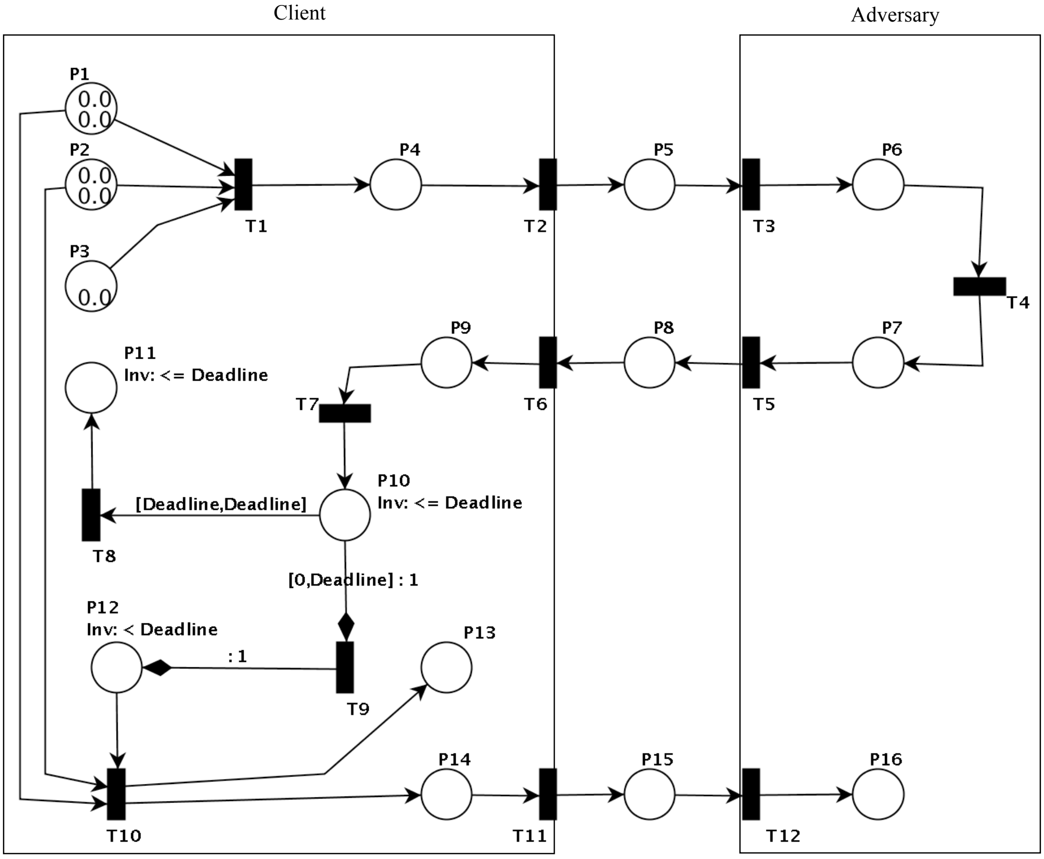


Figure 5. Modelling reflection attack

Since client ***C***'s login request [ID_C, T_C, W_1, M_3, $MAC_k(ID_C, T_C, W_1 ,M_3)$] is symmetrical to server ***S*** response [ID_C, T_S, W_2, M_6, M_7, $MAC_k(ID_C, T_S, W_2, M_6, M_7)$] but the differences between them can be only found in the timestamps and hash values. This symmetry flaw leads to reflection attack. To exploit the reflection attack, the adversary ***A*** intercepts the login request while listening to the electronic conversation between client ***C*** and server ***S***. Then, the adversary sends the same login request [$ID`_C$, $T`_C$, $W`_1$, $M`_3$, $MAC`_k(ID_C, T_C, W_1 ,M_3)$] to ***C*** in a timely manner.

It is obvious that, upon receiving the forged server's response (which is in fact the adversary's reply request), ***C*** will automatically acknowledge the response since the computation is accomplished with the correct key, so the MAC integrity check will succeed. Consequently, ***A*** successfully masquerades as ***S*** and the protocol fails to provide mutual authentication.

Although, ***A*** can cheat ***C*** into believing it is communicating with ***S***, ***A*** cannot obtain the corresponding session key *sk*. Still this type of attack is deemed to represent a breach of the basic obligation of mutual authentication with limited damage. ***A*** performed the exploit without the knowledge of key *k*, merely by intercepting the challenge and sending it back to ***C***.

C. *Analysis of Parallel Session Attack*

Another attack, which is effective against the proposed model without encrypted traffic, is parallel session attack. This attack uses deception to compromise authentication protocols. It involves selecting a valid combination of information from ongoing protocol executions. Fig. 6 explains the exploitation of parallel session attack on the proposed protocol with presence of adversary. The message exchange in this attack is mainly between the server and the adversary leaving the client completely out of the picture. The definitions of the places and the transitions for this model are defined in Table 9 and Table 10, respectively.

Table 9. DEFINITIONS OF PLACES - PARALLEL SESSION ATTACK MODEL

Place	Definition	Place	Definition
P_1	Client random number	P_{16}	Sent SYN/ACK
P_2	Client timestamp	P_{17}	Received SYN/ACK
P_3	SYN request	P_{18}	Fabricated Fake SYN
P_4	Login request	P_{19}	Sent fake SYN
P_5	Sent request	P_{20}	Received fake SYN
P_6	Intercepted MSG	P_{21}	Verification message
P_7	Forge MSG	P_{22}	Rejected request
P_8	Sent Forge MSG	P_{23}	Accepted request
P_9	Verification message	P_{24}	Server random number
P_{10}	Rejected request	P_{25}	Server timestamp
P_{11}	Accepted request	P_{26}	Session Key
P_{12}	Server random number	P_{27}	SYN/ACK
P_{13}	Server timestamp	P_{28}	Sent SYN/ACK
P_{14}	Session Key	P_{29}	Received SYN/ACK
P_{15}	SYN/ACK		

Table 10. DEFINITIONS OF TRANSITIONS - PARALLEL SESSION ATTACK MODEL

Trans.	Definition	Trans.	Definition
T_1	Compute login request + SYN	T_{10}	Send SYN/ACK
		T_{11}	Intercept MSG
T_2	Send MSG	T_{12}	Fabricate SYN
T_3	Intercept MSG	T_{13}	Send fake SYN
T_4	Send forge MSG	T_{14}	Receive forge SYN
T_5	Received Forge MSG	T_{15}	Split the packet and verify
T_6	Split the packet and verify	T_{16}	Drop the request
		T_{17}	Accept
T_7	Drop the request	T_{18}	Compute SYN/ACK and session key
T_8	Accept		
T_9	Compute SYN/ACK and session key	T_{19}	Send SYN/ACK
		T_{20}	Receive SYN/ACK

In the authentication phase of the proposed protocol, the adversary ***A*** can masquerade as an authorised client without prior knowledge of the password or biometric data. The exploit starts when ***A*** eavesdrops on the communication between ***C*** and ***S***. ***A*** intercepts and blocks the ***S*** response message: P_{16} = [ID_{C_i}, T_{S_i}, W_2, M_6, M_7, $MAC_k(ID_{C_i}, T_{S_i}, W_2, M_6, M_7)$]. Then, ***A*** instantly impersonates ***C*** and initiates a new session with ***S*** by sending a fabricated login request: P_{19} = [$ID_A = ID`_C$ , $T_A = T`_S$, $W_1 = W`_2$, $M_3 = M`_6$, M_7, $MAC`_k(ID_C, T_S, W_2, M_6, M_7)$], which is ***S*** original reply to ***C***.

Assume if the fabricated message arrives to ***S*** at time *T*, it will pass the verification check for the following reasons:

(1) The likelihood of correlation associated with $T^{`} - T_C \leq \Delta T$ will be high considering the time-delay in wide-area networks is unpredictable and varies most of the time. Thus, ΔT is often set higher than the timespan of a complete round-trip [16-18]

(2) The MAC integrity check will give a positive result since $MAC`_k(ID_C, T_S, W_2, M_6, M_7)$ is actually computed with the correct key *k* by ***S***.

Based on the above assumptions, ***S*** generates random number P_{24} and timestamp P_{25} to computes session key P_{26} and SYN/ACK response P_{28}, and sends it to ***A***.

D. *Analysis of Impersonation Attack*

One possible attack against the proposed model is impersonation attack. Based on the simulation of man-in-the-middle attack, reflection attack, and parallel session attack, the model reveals a potential risk and weakness that lead to impersonation attack. The adversary ***A*** can mount impersonation attack without knowing any other secret information or credentials by intercepting the login request [ID_C, T_C, W_1, M_3, $MAC_k(ID_C, T_C, W_1 ,M_3)$]. Hence, ***A*** can exploit the proposed protocol by using any of the methods explained previously and hijacking sessions transmitted between ***C*** and ***S***. Eventually, ***A*** succeeds to impersonating either the client or the server.

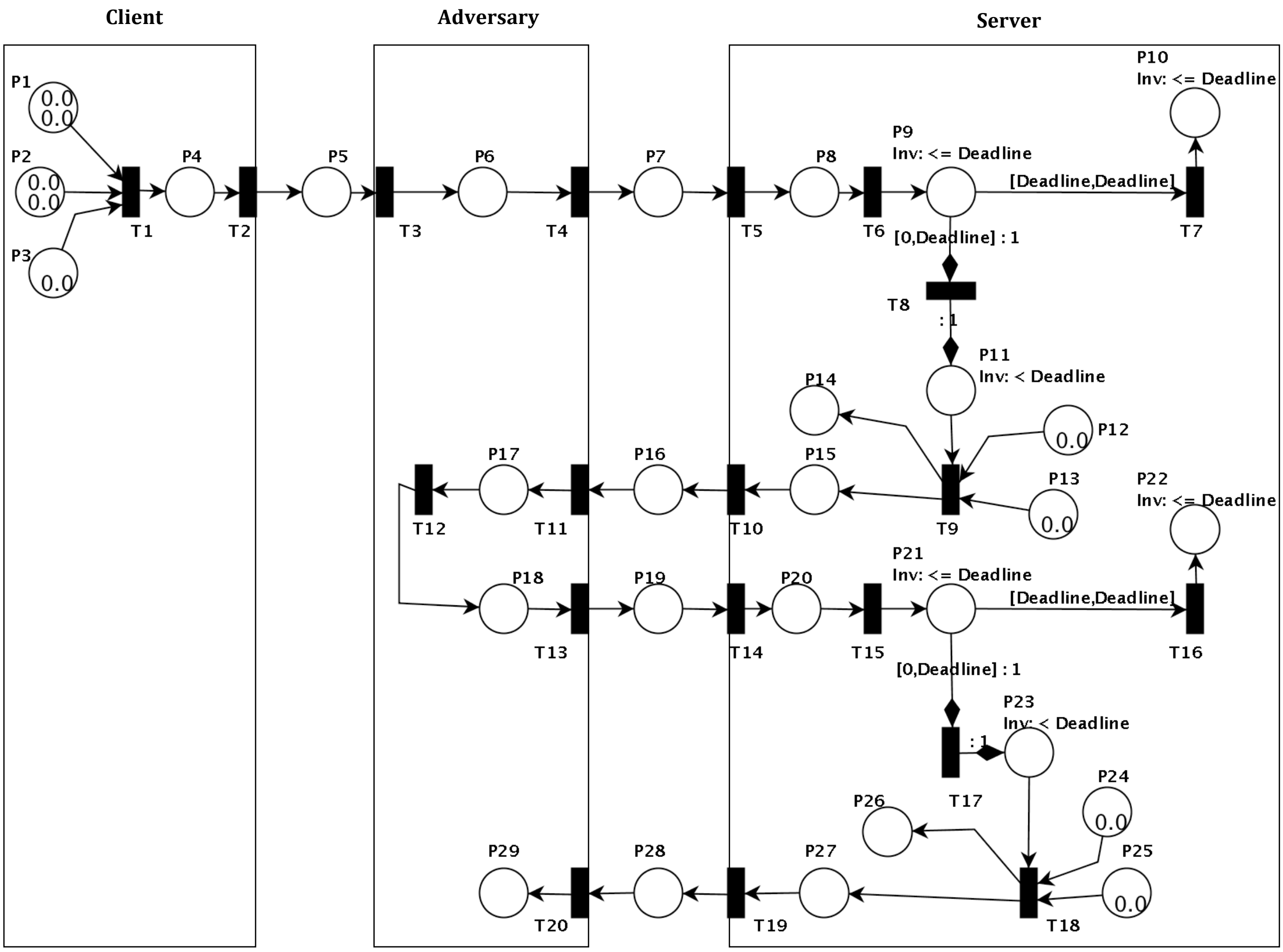


Figure 6. Modelling parallel session attack

V. SECURITY ANALYSIS AND DISCUSSION

Security analysis is a crucial significant process in evaluating communication and cryptographic protocols. The flaws within the protocol can be quickly removed via two solutions. First, encrypting the traffic between client and server creates a private channel to transmit a confidential conversation and calculate the session key. This mechanism is the most cost-effective solution. It is insignificant if the adversary gets hold of an encrypted form of sensitive data as long as it does not obtain the corresponding decryption key. In the second solution, the absence of server identity allows an adversary to simply masquerade as a trusted server. It is possible to optimise the protocol with a simple technique such as adding the server $ID_{S,}$ which can fix the problem. Encrypting traffic protects client's anonymity; user anonymity is one of the security features of remote login system.

Variations of these attacks can be modelled in all phases of mutual authentication and key agreement of cycle. Modelling and simulation revealed that the unencrypted traffic does not provide a full secure authentication and permit a sensitive credential information travel in clear.

Replay Attack. The security feature in the proposed protocol can withstand replay attack due to the use of freshness property. This is guaranteed by applying timestamps and random numbers for each authentication session. To validate the authenticity of messages exchanged between **C** and **S**, the freshness of timestamps is constantly checked. For example, the verification request will fail if $T^{`} – T_C > \Delta T$. This will cause the session to be terminated. Moreover, a new session key is constructed in every authentication cycle. It is worth to mentioning that, the adversary cannot compromise the old session key because it is never been transmitted in the protocol execution between the client and the server. One of the new protocol merits is that each entity computes the correct session key based on the information exchanged between them.

Forgery Attack. The adversary ***A*** cannot create a valid login from scratch without knowing the secret value and the private key of the client. Thus, the adversary cannot act as a legal client so the attack is not feasible.

VI. Conclusion and Future Work

This paper presents a formal approach for enumerating the vulnerabilities and flaws in our protocol and determining suitable countermeasures to fix them. First, PN is used to model the client-server trust model. Then, an adversary entity is added to trust model to analyse various attacks and understand possible behaviours of the adversary. Each attack scenario has been simulated using PN to exploits vulnerabilities in case if the symmetric encryption was not applied to our new protocol.

Adding an adversary to the model encourages discovering and discussing scenarios where the system was under malicious attack. The range of attacks tested the behaviour of the protocol and helped to understand possible behaviours of the adversary during attacks. Each attack scenario has been simulated using PN to exploits vulnerabilities in case if the symmetric encryption was not applied to our new protocol.

It is evident that the most viable countermeasure to defend authentication attacks is to encrypt the message exchange between the client and server. Since the traffic is encrypted between the client and server, this proves that our new protocol is resistant to man-in-the-middle attack, reflection attack, parallel session attack, and impersonation attack. Also, this paper shows that replay attack and forgery attack are not effective because of the freshness property and the difficulty of creating a login request without learning any prior credentials. This analysis shows that our protocol is efficient and provides secure communication over insecure channels.

Future work will include examining ciphertext attack where the adversary can eavesdrop and intercept encrypted messages. PN will be used for modelling and simulating the attack. Once the security analysis is completed, any modification will be considered to enhance our protocol, such as including server ID in the protocol. Consequently a complex client-server trust model will be simulated and validated via PN.

Acknowledgment

This research has been funded by Saudi Arabian Cultural Bureau in London and King Abdul Aziz University in Saudi Arabia.

References

[1] Ryan, P. and Schneider, S.A., 2001. The modelling and analysis of security protocols: the csp approach. Addison-Wesley Professional

[2] Nieh, B.B. and Tavares, S.E., 1993. Modelling and analyzing cryptographic protocols using Petri Nets, *Advances in Cryptology—AUSCRYPT'92* 1993, Springer, pp. 275-295.

[3] Al-Azzoni, I., Down, D.G. and Khedri, R., 2005. Modeling and verification of cryptographic protocols using coloured petri nets and design/CPN. *Nordic Journal of Computing,* **12**(3), pp. 201.

[4] Dresp, W., 2005. Security analysis of the secure authentication protocol by means of coloured petri nets, *Communications and Multimedia Security* 2005, Springer, pp. 230-239.

[5] Permpoontanalarp, Y. and Sornkhom, P., 2009. A new Coloured Petri net methodology for the security analysis of cryptographic protocols, *Proceedings of the 10th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark* 2009.

[6] Xu, Y. and Xie, X., 2011. Modeling and analysis of security protocols using colored petri nets. *Journal of Computers,* **6**(1), pp. 19-27

[7] Petri, C.A., 1962. Kommunikation mit Automaten. Ph. D. Thesis, University of Bonn.

[8] Silva, Manuel. "50 years after the PhD thesis of Carl Adam Petri: A perspective." In Discrete Event Systems, vol. 11, no. 1, pp. 13-20. 2012

[9] Peterson, J.L., 1981. Petri Net Theory and the Modeling of Systems. Prentice-Hall.

[10] Murata, T., 1989. Petri nets: properties, analysis and applications. Proceedings

[11] Bobbio, A., 1990. System modelling with Petri nets. *Systems reliability assessment.* Springer, pp. 103-143.

[12] Aljeaid, D., Ma, X. and Langensiepen, C., 2014. Biometric identity-based cryptography for e-Government environment, *Science and Information Conference (SAI), 2014* 2014, IEEE, pp. 581-588.

[13] Shamir, A., 1985. Identity-based cryptosystems and signature schemes, *Advances in cryptology* 1985, Springer, pp. 47-53.

[14] Aljeaid, D., Ma, X. and Langensiepen, C., Modelling and Simulation of a Biometric Identity-Based Cryptography. *International Journal of Advanced Research in Artificial Intelligence (IJARAI),* **3**(10),.

[15] TAPAAL 2.4.3 Petri nets simulation and verfication of timed-arc Petri nets. Available ar: www.tapaal.net.

[16] Mills, D.L., 1991. Internet time synchronization: the network time protocol. Communications, IEEE Transactions on, 39(10), pp. 1482-1493

[17] Giridhar, A. and Kumar, P., 2006. Distributed clock synchronization over wireless networks: Algorithms and analysis, Decision and Control, 2006 45th IEEE Conference on 2006, IEEE, pp. 4915-4920.

[18] Han, J. and Jeong, D., 2010. A practical implementation of IEEE 1588-2008 transparent clock for distributed measurement and control systems. Instrumentation and Measurement, IEEE Transactions on, 59(2), pp. 433-439.