Formal Verification of Forward-Secure Authenticated Key Exchange Scheme for Location-based Service Application

Mahesh Kumar K.M, Pradeep R, Sunitha N.R

Abstract:A Location-based service (LBS) is a popularinformation service which uses the geographical position of theuser to provide service. Major challenges for wide deploymentof such services is security and privacy, in our paper we propose a generic model of authenticated key exchange (AKE)protocol termed as forward-secure authenticated key exchangeprotocol (FSAKE) which uses elliptic curve cryptosystem. TheFSAKE protocol supports concurrent sessions and is used fortheexchange of secure seed values which are used in forward-securepseudo-random number generators to generate secret keys formessage authentication and symmetric encryption. The FSAKEprotocol is a key evolving scheme which updates the long-termkeys (LTKs) at regular intervals and guarantees the security ofthe past keys and mitigates the damage caused by exposure ofthe current key. We make use of Scyther model checking tool toprove the correctness of FSAKE protocol security.

Keywords*: Authenticated Key Exchange, EllipticCurve Cryptography, Forward-Security, Formal Verification,Location-Based Services, Symmetric Key Evolving Systems.

I. INTRODUCTION

An LBS is a service offered to users who can access the information with the help of mobile devices such as smart phones, pocket PCs, GPS devices. Mobile network forms the backbone to LBS, which offers services to the users based on the mobile device's geo graphical position. LBS has several applications like navigation, asset tracking, gaming, location-based queries (discovering the nearest hospitals, restaurant, fuel stations, etc.).

Three main stakeholders can be identified in LBS see Fig.1, first the Location Provider: the source of the location information, second the Location-Based Service Provider: the source of the location-specific information and lastly the User who avails the service of location provider and location-based service provider [1].

A Forward-Secure Authenticated Key Exchange (FSAKE) protocol should withstand key exposure attacks, even if the adversary learns information about the current key, he should not be able to derive past keys used for communication. Elliptic curve cryptography (ECC) [2, 3, 4] provides an efficient public-key system which can support the forward-secure AKE.

LBS provider maintains a point of interest database; user queries are answered by the LBS server by retrieving Pointsof Interest (POIs) from the database. There are several areas of concerns for the wide deployment of such services; major challenges are privacy and security assurance. It is hard to preserve confidentiality and integrity of data when we are transmitting it over a public network (wireless network) which is insecure, thus establishing a secure channel plays a pivotal role in network communication.

Establishing secure channels without trust among the communicating parties is a risk; AKE provides a standard solution to solve this problem by authenticating the communicating parties and establishing trust among them.

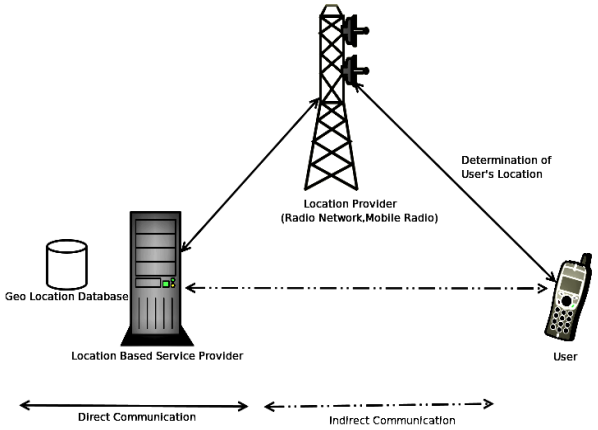


Fig.1. Location Based Service

Although we could make use of LTKs to preserve the confidentiality and integrity of data, in general, it is not wise due to the risk of key exposure. The past communication recorded by the passive listener can now be compromised with the exposure of the LTK. A good strategy is to use temporary session keys to secure the channels, and also to provide forward-security, i.e., exposure of the current key compromises only the communication that was encrypted using it and the past communications encrypted using previous keys remain secure.

Revised Manuscript Received on December 12, 2019.
***** Correspondence Author
Mahesh Kumar K.M*, Dept. of CSE, Siddaganga Institute of Technology, Tumakuru, India. Email: maheshkumarkm87@gmail.com
Pradeep R, Dept. of CSE, Siddaganga Institute of Technology, Tumakuru, India. Email: pradeepr@sit.ac.in
N.R. Sunitha, Dept. of CSE, Siddaganga Institute of Technology, Tumakuru, India. Email: nrsunitha@sit.ac.in

Retrieval Number: B10731292S19/2019©BEIESP
DOI: 10.35940/ijitee.B1073.1292S19

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

A. Related work

Early researchers in their work on secure authentication adopted zero-knowledge model [5] proposed by Feige et al.,[6] or they followed the ad hoc approach, these models were tailored to meet the needs of smart card identification.

Bellare and Rogaway in their work [7] provided the formalized model and definitions for AKE protocols and entity authentication. They were the first to recognize that an individual entity can be part of multiple instances of the protocol called "session". Each session was modelled as Oracle and adversaries were given access to it through the standard three queries: send, reveal and test. An adversary in a session can send a specific message through send query, he can obtain session key of the specific session through reveal query, he can obtain random value or the actual session key depending on the coin flip by the test query.

In this paper, we follow two-party authentication or AKE using symmetric keys and use random oracles similar to [7]. However, unlike [7] we support the concept of forward-security and allow an adversary to obtain the LTKs. Bellare-Rogaway model [7] was adopted to an asymmetric key setting by Blake-Wilson et al., [8, 9] and they provided forward-security and protection against Unknown Key-Share (UKS) and Key-Compromise Impersonation (KCI) attacks. Three-party AKE model was introduced by Bellare and Rogaway [10], which included a corrupt query, allowing an adversary to obtain and set the LTKs.

In [11] authors present a definition and formal model for forward-secure authenticated key exchange using key-evolving schemes and proposed a FORSAKES protocol which demands less resource and realized by a hash function. Our approach is similar to [11] but unlike [11] our paper makes use of ECC to achieve authenticated key exchange, provides the solution to mitigate damage caused by LTK exposure.

D. Konidal et al., in their paper [12] propose a protocol to accomplish the task of secure key distribution between the LBS user and genuine service providers using a trust model which is convincing. This paper has achieved secure job handling at low cost using cryptography, they have been successful in reducing the burden on mobile devices in terms of communication and computation cost, also users are protected against privacy and replay attacks.

M. Lee et al., in their paper [13] propose a novel architecture for mobile users which allow them to create a dynamic policy and enforce it for secure and accurate LBS in a wireless network. Authors also list scenarios where in the proposed architecture is useful in keeping mobile user's location information private and secure.

In our work, we are focusing mainly on the security of LBS. The key idea is to make use of ECC for the secure authenticated exchange of "seed" (initial key) and temporary key between LBS provider and users. The seed along with the temporary key is then used by the forward-secure pseudo-random generator to generate secure keys timely for encryption and message authentication purposes.

B. Contributions

Our contributions are as follows:

• We propose a Forward Secure AKE protocol for LBS(FSAKE) using ECC, where the LTKs are updated overtime and also satisfy the forward-security property (refersection II).

• We explain the FSAKE protocol at a high level andalgorithmically describe it (refer section II-D).

• We formally verify the security of FSAKE protocol using formal verification tool Scyther (refer section III).

C. Organization

The rest of the paper is organized as follows: Section II presents the high-level overview of the proposed FSAKE protocol along with the details of the message generation and managing algorithms. Section III presents the formal verification of FSAKE protocol, Section IV presents the formal verification results and discussion, followed by Section V concluding the paper.

II. THE PROPOSED FSAKE PROTOCOL

The proposed FSAKE protocol works by exchanging three messages between the initiator and responder. We assume that the initiator and responders aresynchronized in time and maintain system time, an LTK and session states. The rest of the FSAKE protocol details are as follows:

A. System Time Stage

Current time stage of the system is denoted by a variable T and is equivalent to the system's epoch. T is represented as *64-bit* unsigned integer and can hold up to 2^{64} time stages, every τ seconds T gets incremented to new value.

B. Long-Term Keys

Let id_x and id_y represents the identity of FSAKE initiator and responder respectively, and K_{xy}^{θ}represents the long-term key agreed by the initiator and responder at time stage $T = \theta$ gets updated to $K_{xy}^{\theta+1}$(new LTK) when time stage gets incremented from $T = \theta$ to $T = \theta + 1$.

C. Session State

FSAKE protocol allows concurrent key exchanges to run at the same instance over time. To track each instance or session, FSAKE maintains session information in the form of session states, and are kept in memory. Each session makes use of independent randomness and is stored separately. To keep track of the internal session a designated counter ctr is used. To start with ctr is initialized to 0 and is incremented upon storing new session states. The variable $state_z^i$denotes the session state with session counter i on an entity $z \in \{x, y\}$, and its composition is as follows:

- $sid \in HASH(rnd_x^{ctr}, id_x, pid_y^{ctr})$: The session identifier. Resultant of the hash of the ordered concatenation of the random value rnd_x^{ctr}, Initiators identity (id_x) and the session partners identity (pid_y^{ctr}).
- $role_z^i \in \{'I', 'R'\}$: In a session, communicating entity has two possible roles the initiator ('I') responsible for initiating the session or the responder ('R').
- $pid_z^i \in \{0,1\}^l$: Binary sequence denoting session partner (Initiator/Responder).

- $T_z^i \in \{0,1\}^{64}$: Session initiation time stage. By design, FSAKE ensures the exchange of three standard messages in the permitted window interval. Failing which, results in deletion of session state.
- $EK_z^i \in \{0,1\}^{h/2}$: The Encryption key generation seed, where $h/2$ is the half the length of HASH value of shared secret *S*.
- $IK_z^i \in \{0,1\}^{h/2}$: The Integrity key generation seed, where $h/2$ is the half the length of HASH value of shared secret *S*.
- $acp_z^i \in \{\lambda, 0,1\}$: The session's acceptance decision. Acceptance is undecided (λ) in the beginning of the session. Upon session acceptance decision, acp_z^i will be either false (0) or true (1).

D. Session Initiation

The initiator session is assumed to be invoked by the application program, and no other party is expected to send any message to initiate the session. Call from the higher-level application program is treated as a zeroth message. The desired partner is identified by its identity by the caller. Let say id_x and id_y be the respective ids of the initiator and its partner. Algorithm 3 explains the details of session initiation by initiator and first protocol message generation process. The initiator first sets the session partner ID to id_y, and then increases the session counter *ctr*. Next, the session ID is set to rnd_x^{ctr} . Recall that rnd_x^{ctr} is the current session's randomness.

Algorithm 1: Digital Signature Generator (DSG)

Require: Incoming Parameters: MSG, d_A

1. $e \leftarrow HASH(MSG)$;
2. choose k randomly such that $0 < k < n$;
3. $R \leftarrow \binom{r_x}{r_y} = k * G$; //$G$ is a public known point on the curve.
4. $r \leftarrow r_x \bmod n$; $r \neq 0$;
5. if $r == 0$ then
6. repeat step 2;
7. else
8. $s \leftarrow \frac{e + d_A * r}{k} \bmod n$ //d_A is the private key of signer
9. if $s == 0$ then
10. repeat step 2;
11. else
12. return $(r; s)$; //Digital signature of MSG.
13. end if
14. end if

Session time stage gets set to *T* in the next step, the ephemeral keys i.e., integrity key and encryption key are set to empty, acceptance state is initialized to λ indicating session is undecided yet. The session state and M_1 are set in the last step. Session state consists of every other state variable:

$$state_z^i \leftarrow sid_x^{ctr} || role_x^{ctr} || pid_x^{ctr} || T_x^{ctr} || EK_x^{ctr} || IK_x^{ctr} || acp_x^{ctr};$$

Algorithm 2: Digital Signature Verifier (DSV)

Require: Incoming Parameters: $MSG, (r, s), Q_A$

1. $e \leftarrow HASH(MSG)$;
2. $w \leftarrow s^{-1} \bmod n$;
3. $u_1 \leftarrow e * w \bmod n$;
4. $u_2 \leftarrow r * w \bmod n$;
5. $P \leftarrow \binom{p_x}{p_y} \leftarrow u_1 * G + u_2 * Q_A$; //$Q_A$ is the public key of signer
6. if $p_x == r \bmod n$ then
7. return 1; //signature valid
8. else
9. return 0; //signature invalid
10. end if

Algorithm 3: Initiator: Session initiation

Require: Input: None (0)
Ensure: $pid_x^{ctr} \leftarrow id_y$

1. $ctr \leftarrow ctr + 1$;
2. $sid_x^{ctr} \leftarrow HASH(rnd_x^{ctr} || id_x || id_y)$;
3. $role_x^{ctr} \leftarrow 'I'$;
4. $T_x^{ctr} \leftarrow T$;
5. $EK_x^{ctr} \leftarrow IK_x^{ctr} \leftarrow \lambda$;
6. $acp_x^{ctr} \leftarrow \lambda$; // Session fate is undecided yet.
7. $state_z^i \leftarrow sid_x^{ctr} || role_x^{ctr} || pid_x^{ctr} || T_x^{ctr} || EK_x^{ctr} || IK_x^{ctr} || acp_x^{ctr}$;
8. Choose $rnd_{x_1}^{ctr}$ such that $0 < rnd_{x_1}^{ctr} < n$;
9. $R_x \leftarrow rnd_{x_1}^{ctr} * G$;
10. let K_{xy}^T be the LTK shared between id_x and pid_x^{ctr};
11. $K_{xy}^T \leftarrow rnd_{x_1}^{ctr} * Q_y$;
12. $MSG_1 \leftarrow 1 || id_x || pid_x^{ctr} || T_x^{ctr} || rnd_x^{ctr} || R_x$;
13. $AUTH_1 \leftarrow DSG(MSG_1, d_x)$// Digital signature of MSG_1.
14. return $MSG_1 || AUTH_1$;

It is sufficient to store only the session state since it consists of all other session variables. When we need individual variable value such as sid_x^{ctr}, it is assumed that from the. Session state it is possible to interpret the constituent parts. First protocol message construction takes place as follows:

$$MSG_1 \leftarrow 1 || id_x || pid_x^{ctr} || T_x^{ctr} || rnd_x^{ctr} || R_x;$$

The message starts with integer value 1, meaning it is the first message of the sequence. It is useful for a partner taking part in several simultaneous session to identify and distinguish the incoming message. It also helps in preventing typing attacks.

The identity of the sender (id_x) and partner (pid_x^{ctr}) is always part of the message, they follow the specific order in all messages of the FSAKE protocol. This is necessary to prevent parallel session attack [14]. Initiator ensures that MSG_1 contains the current time stage to prevent the receiver from accepting outdated messages. Initiator includes a nonce value (rnd_x^{ctr}), in order to prevent the replay attacks. Finally, MSG_1 includes R which is the resulting point obtained upon multiplying publicly known point on elliptic curve G with the randomly generated number $rnd_{x_1}^{ctr}$, R_x value is used in the computation of shared symmetric key by the session partner. The authentication code is computed using Algorithm 1 as follows:
$AUTH_1 \leftarrow DSG(MSG_1, d_x)$. $MSG_1 || AUTH_1$ is sent to the responder.

Algorithm 4: Responder: Managing first message and response generation.

$\overset{MSG_1}{\_\_\_\_\_}$

Require: Input: $m = 1||id_s||id_r||T_s||rnd_s||R_x||AUTH_1$

1. if $id_r \neq id_y$ *or* $T_s \neq T$ *or* $DSV(MSG_1,(r,s),Q_x) \neq 1$ *then*
2. return;
3. end if
4. $ctr \leftarrow ctr + 1$;
5. $pid_y^{ctr} \leftarrow id_s$;
6. $sid_y^{ctr} \leftarrow HASH(rnd_s||id_s||id_r)$;
7. $role_x^{ctr} \leftarrow$' R';
8. $T_y^{ctr} \leftarrow T$;
9. let K_{yx}^T be the LTK shared between id_y and pid_y^{ctr};
10. $K_{yx}^T \leftarrow d_y * R_x$;
11. choose n_R^{ctr} randomly such that $0 < n_R^{ctr} < n$;
12. $P_R \leftarrow n_R^{ctr} * G$; // G is a publicly known point.
13. $EK_x^{ctr} \leftarrow IK_x^{ctr} \leftarrow \lambda$;
14. $acp_x^{ctr} \leftarrow \lambda$; // Session fate is undecided yet.
15. $state_y^{ctr} \leftarrow sid_y^{ctr}||role_y^{ctr}||pid_y^{ctr}||T_y^{ctr}||EK_x^{ctr}||IK_x^{ctr}||acp_y^{ctr}$;
16. $MSG_2 \leftarrow 2||id_y||id_x||T_x^{ctr}||sid_y^{ctr}||P_R$;
17. $AUTH_2 \leftarrow DSG(MSG_2, d_y)$; // Digital signature of MSG_2.
18. return $MSG_2||AUTH_2$;

E. Managing the First Message

MSG_1 specifies the first message of the FSAKE protocol. First thing receiver verifies upon receiving this message is the syntax and upon successful verification follows Algorithm 4. $m = 1||id_s||id_r||T_s||rnd_s||R_x||AUTH_1$ denotes the incoming message, where *s* denotes the sender and *r* denotes the receiver. For example,id_s and id_r means the sender ID and receiver ID respectively.

The receiver verifies whether ID of receiver is same as its own identity ($id_r = id_y$) meaning message is intended for him, checks whether time stage of message generation and its own time stage are same ($T_s = T$) and if $AUTH_1$signature verifies (i.e. $DSV(MSG_1,(r,s),Q_x) == 1$using Algorithm 2). If any of the above condition fails, the responder executes the return command, meaning that he is no longerinterested in this message.

Next, the responder increments his session counter *ctr*. The receiver sets the partner ID to id_s, the session ID to $h(rnd_s||id_s||id_r)$, '*R*' is set as the session role, and *T* set as the session time stage.

Responder computes the K_{yx}^T (LTK shared between the communicating parties) using public pointR_x and its private key d_y.Next, responder chooses n_R^{ctr} randomly such that $0 < n_R^{ctr} < n$ and computes public point P_R by multiplyingn_R^{ctr} with publicly known point *G*.

Both the ephemeral keys are not computed yet hence set to $\lambda(EK_x^{ctr} \leftarrow \lambda; IK_x^{ctr} \leftarrow \lambda;)$.

Next, to indicate the undecided state of the session we set the acceptance value of the session to λ, and we proceed to construct the session state. Second message generation takes place similar to the first message with a similar syntax.

The second message construction and the authenticator generation are as follows:

$$MSG_2 \leftarrow 2||id_y||id_x||T_x^{ctr}||sid_y^{ctr}||P_R \; AUTH_2 \leftarrow DSG(MSG_2, d_y)$$

Finally, the responder sends $MSG_2||AUTH_2$.

F. Managing the Second Message

The first thing initiator verifies upon receiving the second message is the syntax, and upon successful verification follows Algorithm 5. The initiator then checks for session ID of the incoming message to verify the existence of a matching session with an identical session ID. Next, $AUTH_2$ is verified, if no match is found or if $AUTH_2$ verification fails, the incoming message is rejected (via return).

Algorithm 5: Initiator: Managing second message and response generation.

$\overset{MSG_2}{\_\_\_\_\_}$

Require: Input: $m = 2||id_s||id_r||T_s||sid||P_R||AUTH_2$

1. if $sid_x^i \neq sid$ *for all* $i \in [ctr]$ *or* $DSV(MSG_2,(r,s),Q_y) \neq 1$ *then*
2. return;
3. end if
4. if $sid_x^i \neq sid$ *for more than one* $i \in [ctr]$ *then*
5. $state_x^i \leftarrow \lambda$; for all such $i's$;
6. return;
7. end if
8. $i \in [ctr]$ a unique number $sid_x^i = sid$;
9. *if* $id_s \neq pid_y^i$ *or* $id_r \neq id_x$ *or* $T_s \neq T$ *or* $T_s \neq T_x^i$ *or* $role_x^i \neq$' I' *or* $acp_x^i \neq \lambda$ *then*
10. $state_x^i \leftarrow \lambda$;
11. return;
12. end if
13. *let* K_{yx}^T *be the LTK shared between* id_x *and* pid_x^i;
14. *choose* n_I^{ctr} *randomly such that* $0 < n_I^{ctr} < n$;
15. $P_I = n_I^{ctr} * G$ *// G is a publicly known point.*
16. *let S be the secret,* $S = n_I^{ctr} * P_R$;
17. $H_{fh}||H_{sh} \leftarrow H \leftarrow HASH(S)$ *//* H_{fh}&H_{sh} *denotes 1st and 2nd half.*
18. $EK_x^i \leftarrow H_{fh}$;
19. $IK_x^i \leftarrow H_{sh}$
20. $acp_x^i \leftarrow 1$;
21. $state_x^i \leftarrow sid_x^i||role_x^i||pid_x^i||T_x^i||EK_x^i||IK_x^i||acp_x^i$;
22. $MSG_3 \leftarrow 3||id_x||pid_x^i||T_x^i||sid_x^i||P_I$;
23. $AUTH_3 \leftarrow DSG(MSG_3, d_x)$; *// Digital signature of* MSG_3.
24. *return* $MSG_3||AUTH_3$;

It is expected that there exists exactly one match. If more than one session with the same session ID as in incoming message exist, all such sessions will be deleted, the incoming message will be rejected (via return). Once we find a unique session that matches the session ID of the incoming message, we proceed with six additional verifications as follows:

1. The sender ID and session partner ID matches.
2. The receiver ID and the current party ID matches.
3. The current and senders time stage matches.
4. The current and the session time stage matches.
5. The current session role is *'I'*, indicating anticipation of the second message.
6. The session did not accept or reject previously.

Failing to meet any of the above conditions, the session state gets wiped securely, and we reject the incoming message (via *return*).

The initiator chooses n_I^{ctr} randomly such that $0 < n_I^{ctr} < n$,P_I is a public point computed by multiplying n_I^{ctr} with *G*. Shared secret *S*, is computed by multiplying n_I^{ctr} with point P_R received. The ephemeral keys are updated as follows: $H_{fh}||H_{sh} \leftarrow H \leftarrow HASH(S)$; $EK_x^i \leftarrow H_{fh}$; $IK_x^i \leftarrow H_{sh}$;

If all the verifications are passed, the acceptance state is set to true, and we proceed to construct the session state.

Third message generation takes place similar to the second message with a similar syntax. The third message construction and the authenticator generation are as follows:

$$MSG_3 \leftarrow 3||id_x||pid_x^i||T_x^i||sid_x^i||P_I$$
$$AUTH_3 \leftarrow DSG(MSG_3, d_x);$$

Finally, the initiator sends $MSG_3||AUTH_3$

G. Managing the Third Message

The first thing the responder verifies upon receiving the third message is the syntax, and upon successful verification follows Algorithm 6. The responder then checks for session ID of the incoming message to verify the existence of a matching session with an identical session ID. Next, $AUTH_3$ is verified, if no match is found or if $AUTH_3$ verification fails, the incoming message is rejected (via return).

Algorithm 6: Responder: Managing third message

$\overline{MSG_3}$

Require: Input: $m = 3||id_s||id_r||T_s||sid||P_I||0||AUTH_3$

1. if $sid_y^i \neq sid$ *for all* $i \in [ctr]$ *or* $DSV(MSG_3, (r,s), Q_x) \neq 1$ *then*
2. return;
3. end if
4. if $sid_y^i \neq sid$ *for more than one* $i \in [ctr]$ *then*
5. $state_y^i \leftarrow \lambda$; for all such $i's$;
6. return;
7. end if
8. $i \in [ctr]$ a unique number $sid_y^i = sid$;
9. *if* $id_s \neq pid_y^i$ *or* $id_r \neq id_x$ *or* $T_s \neq T$ *or* $T_s \neq T_y^i$ *or* $role_x^i \neq' R'$ *or* $acp_y^i \neq \lambda$ *then*
10. $state_y^i \leftarrow EK_y^i \leftarrow IK_y^i \leftarrow \lambda$;
11. return;
12. end if
13. *let S be the secret,* $S = n_R^{ctr} * P_I$; // see algorithm 4 for n_R^{ctr}
14. $H_{fh}||H_{sh} \leftarrow H \leftarrow HASH(S)$ // H_{fh}& H_{sh} *denotes 1st and 2nd half.*
15. $EK_y^i \leftarrow H_{fh}$;
16. $IK_y^i \leftarrow H_{sh}$
17. $acp_y^i \leftarrow 1$;
18. $state_y^i \leftarrow sid_y^i||role_y^i||pid_y^i||T_y^i||EK_y^i||IK_y^i||acp_y^i$;

Next, it is expected that there exists exactly one match. If more than one session with the same session ID as in incoming message exist, all such sessions will be deleted, the incoming message will be rejected (via *return*). Once we find a *unique* session that matches the session ID of the incoming message, we proceed with six additional verifications as follows:

1. The sender ID and session partner ID matches.
2. The receiver ID and the current party ID matches.
3. The current and senders time stage matches.
4. The current and the session time stage matches.
5. The current session role is '*R*', indicating anticipation of the third message.
6. The session did not accept or reject previously.

Failing to meet any of the above conditions, the session state gets wiped securely, and we reject the incoming message (via return).

Shared secret *S*, is computed by multiplying n_R^{ctr} with the point P_I received. The ephemeral keys are updated as follows:

$$H_{fh}||H_{sh} \leftarrow H \leftarrow HASH(S);$$
$$EK_y^i \leftarrow H_{fh}; IK_y^i \leftarrow H_{sh};$$

Ultimately, the receiver accepts and updates the session state. This step concludes the description of FSAKE.

III. FORMAL VERIFICATION OF FSAKE PROTOCOL USING SCYTHER TOOL

In practice, we find several formal analysis tools in useto analyse security protocols. Scyther is a tool which workson the perfect cryptographic assumption; it evaluates securityprotocol based on the premise that without the valid decryptionkey no adversary can learn about the encrypted message[15]. Scyther is useful in detecting problems related to theconstruction of the protocol, these problems are in generalundecidable. However, several protocols have been eitherproved correct or attacks are identified in future [16]. Security properties or the security claims for the securityprotocol are written in the form of specifications using SecurityProtocol Description Language and stored in files with ".spdl".

A. Security Properties as Claim Events

To construct the security protocol, we should consider all the security properties to be satisfied by the protocol. In the Scyther tool, these security properties are encapsulated as claim events and made part of the protocol specification [16][17].

1) *Secrecy*: Secrecy is the first claim event, it claims that a message transmitted across an untrusted channel is not revealed to the adversary.

2) *Authentication*: In literature, authentication exists in several forms, but in simple terms, authentication is a guarantee that a communicating entity exists. By the protocol specification existence of at least two communicating entities is required.

Aliveness: Aliveness is a type of authentication claim event, where in the communicating entity proves it is "alive" which is evident by the execution of certain events. In literature, we find four types of aliveness: *weak aliveness*, *recent aliveness*, *weak aliveness in the correct role* and *recent aliveness in the correct role*. Among the four forms, recent aliveness in the correct role is the strongest one, hence in our FSAKE protocol, we aim to achieve it.

Recent aliveness in the correct role: This claim event is a strict form of aliveness claim, let *S* be a security protocol with roles I and R. $\gamma = claim_l(I, recent-alive-role, R)$, represents the claim event which requires the execution of some events to prove aliveness and also in the correct role(R) as claimed.

3) *Synchronisation*: Synchronisation claim event guarantees that two corresponding entities have exchanged the necessary messages as expected by the security protocol. Additionally, it confirms that information received was sent by the correct sender and all communications were received in the correct order by the receiver.

4) *Non-injective Synchronisation*: This claim event states that trace should contain all the events expected in the actual protocol description. The synchronisation is primarily focused on content and order of the content; hence it is prone to replay attacks. An adversary can capture the message and replay the same at a later instance to the recipient, this, in general, is not treated as an attack on synchronisation.

5) *Injective agreement or Agreement*: In [17] authors suggest that for an agreement between two communicating entities initiator *I* and responder *R* over a data item setd_s, it is expected that *I* and *R* have already completed a run of the protocol in Initiator and Responder role respectively (i.e., *R* acted as responder and agreed on the data set d_s and each run of initiator *I* has unique corresponding run with *R*).

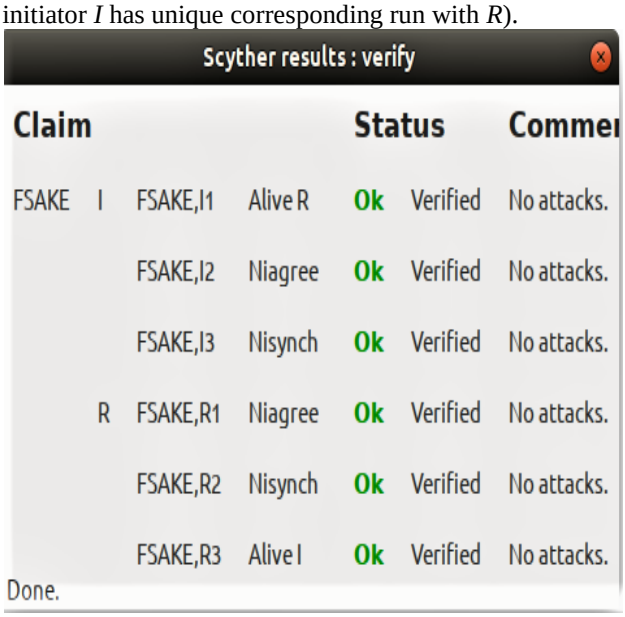


Fig.2. FSAKE Scyther Formal Verification Claim Results.

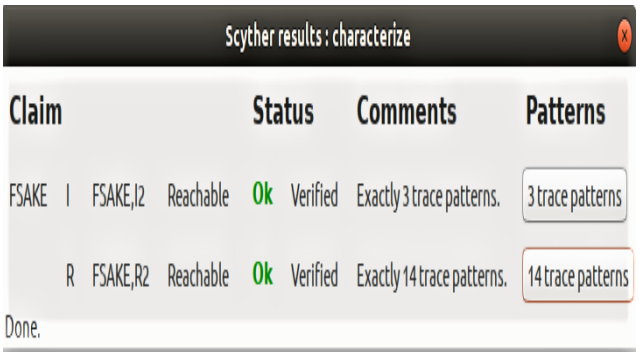


Fig.3. FSAKE Scyther Characterize Roles.

Table I: Attacks Handled

List of Attacks	Mitigated	Prevented
Message Fabrication	Yes	Yes
Man in the Middle Attack	Yes	Yes
Parallel Session Attack	Yes	Yes
Typing Attack	Yes	Yes
Replay Attack	Yes	Yes
Brute Force Attack	Yes	No
System Break-in	Yes	No

IV. RESULTS AND DISCUSSIONS

The security properties of FSAKE protocol was modelledin spdl language using claim events discussed in theprevious section. In this section we present the results anddiscussion of FSAKE protocol, Fig. 2, presents the formalverification results of FSAKE protocol claiming the aliveness,non-injective synchronisation and non-injective agreement.

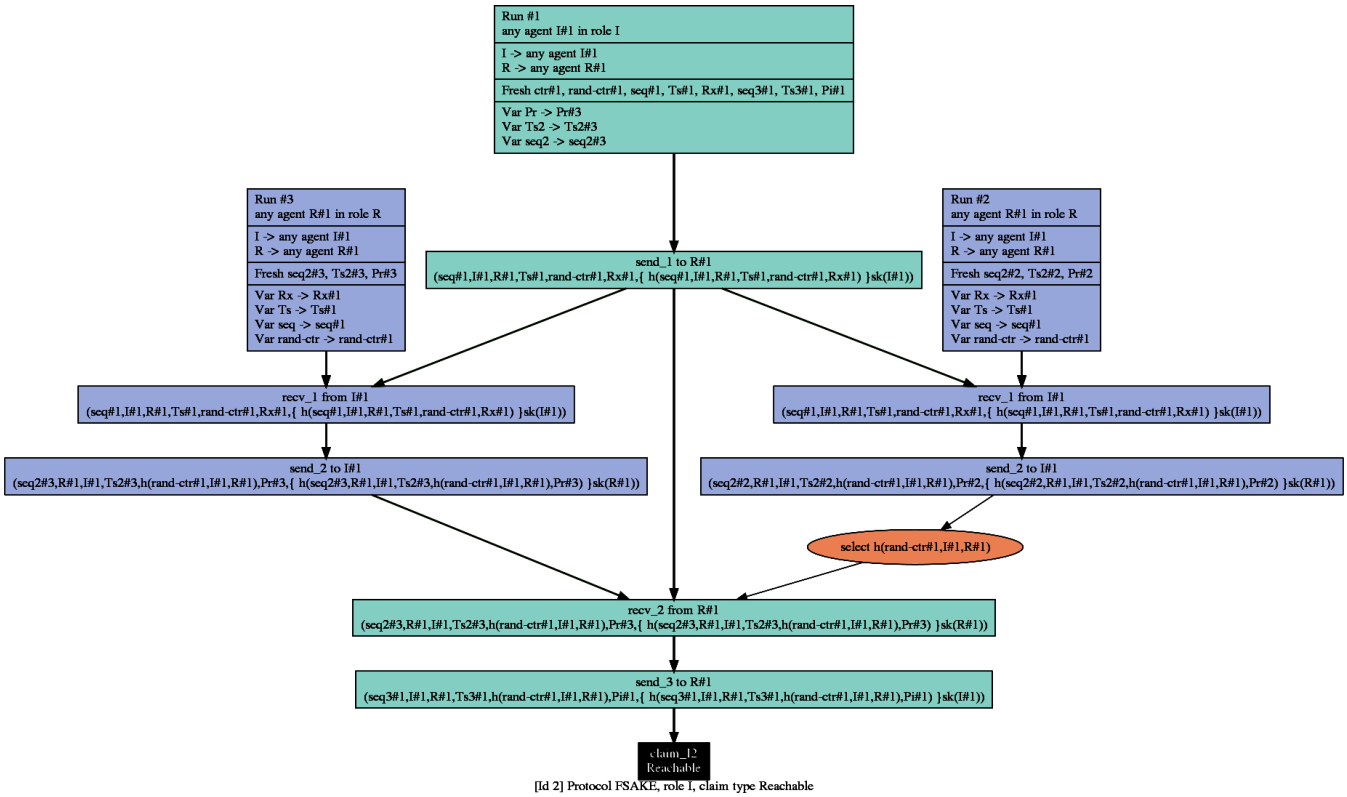
Fig. 3 presents the role characterization and all possibletraces for the role *I* (initiator) and *R* (responder), there are3 possible trace patterns for the role *I* and 14 patterns forrole *R*. Due to page constraints, out of 17 possible tracepatterns we have listed one sample trace pattern for each role.Fig. 4 presents the trace pattern for the role *I* and Fig.5 presents the trace pattern for role *R*, which guaranteesthe reachability of the claims non-injective agreement andnon-injective synchronisation respectively.

In Fig. 4 we can notice that attacker trying to modifythe message content but fails to do so because the attackercannot produce the valid signature of the message and inFig. 5 attacker tries to replay the signed message but failssince the message can be delivered only once and within thetime stage since it contains time stamp. Similarly, remainingtrace patterns can be obtained by running FSAKE spdl code.Use of digital signature for message exchange thwarts manin the middle attack; key evolving scheme makes it hard forguessing the keys or brute force attack. System break-in will compromise the current session keys but the forward security
property thwarts the past key compromise. Table I summarizesthe attacks handled by FSAKE protocol.

V. CONCLUSIONS

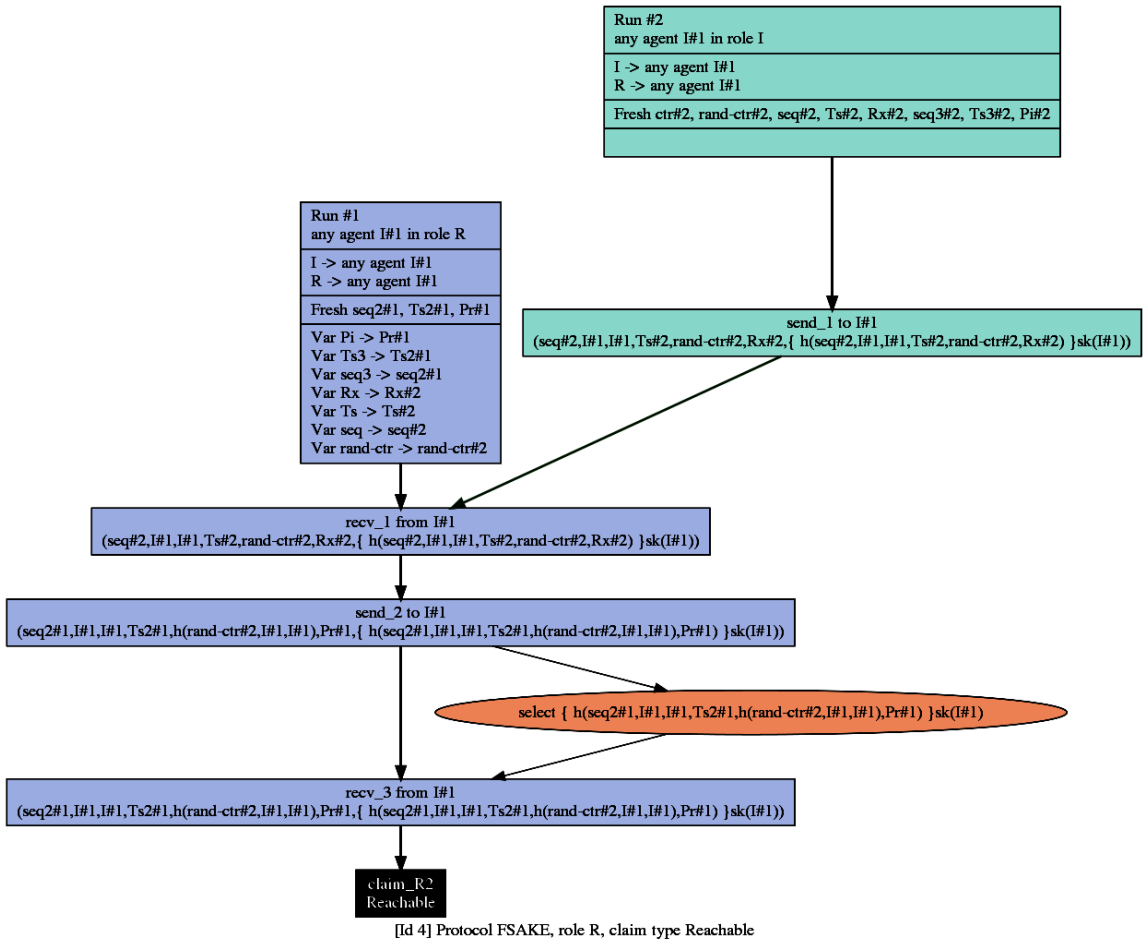
We can enhance the trust among the LBS stakeholders byaddressing security and privacy concerns of LBS applications, which can lead to more revenue and enhanced customer satisfaction. Our paper highlighted the significance of AKEand forward-security. Also, we presented an FSAKE protocol and key-evolving symmetric key cryptosystem to cater to themessage authentication and encryption needs of LBS. Wehave provided formal proof for the proposed FSAKE protocol.

In our future work, we plan to work on issues related to synchronization of shared symmetric keys and providing novel solutions to preserve privacy in different LBS application like PoI based, Friend Finder/Locator, People Finder (Finding Unknown parties) based on the proximity of the mobile users.



[Id 2] Protocol FSAKE, role I, claim type Reachable

Fig.4. FSAKE Scyther Role 'I'.



[Id 4] Protocol FSAKE, role R, claim type Reachable

Fig.5. FSAKE Scyther Role 'R'.

REFERENCES

1. H. Jonker, S. Mauw, and J. Pang, "Location-basedservices: Privacy, security and assurance," DigitalEnlightenment Yearbook 2012, pp. 235–244, 2012.
2. V. S. Miller, "Use of elliptic curves in cryptography," in Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings, 1985, pp. 417–426.
3. A. H. Koblitz, N. Koblitz, and A. Menezes, "Elliptic curve cryptography: The serpentine course of a paradigm shift," IACR Cryptology ePrint Archive, vol. 2008, p. 390, 2008.
4. N. Christin and R. Safavi-Naini, Eds., FinancialCryptography and Data Security - 18th InternationalConference, FC 2014, Christ Church, Barbados,March3-7, 2014, Revised Selected Papers, ser. Lecture Notesin Computer Science, vol. 8437. Springer, 2014.
5. M. S. Dousti and R. Jalili, "Efficient statisticalzero-knowledge authentication protocols for smart cardssecure against active & concurrent quantum attacks,"IACR Cryptology ePrint Archive, vol. 2013, p. 709, 2013.
6. U. Feige, A. Fiat, and A. Shamir, "Zero-knowledgeproofs of identity," J. Cryptology, vol. 1, no. 2, pp. 77–94,1988.
7. M. Bellare and P. Rogaway, "Entity authentication andkey distribution," in Advances in Cryptology – CRYPTO'93, 13th Annual International Cryptology Conference,Santa Barbara, California, USA, August 22-26, 1993,Proceedings, 1993, pp. 232–249.
8. S. Blake-Wilson, D. Johnson, and A. Menezes, "Keyagreement protocols and their security analysis," inCryptography and Coding, 6th IMA InternationalConference, Cirencester, UK, December 17-19, 1997,Proceedings, 1997, pp. 30–45.
9. S. Blake-Wilson and A. Menezes, "Entity authenticationand authenticated key transport protocols employingasymmetric techniques," in Security Protocols, 5thInternational Workshop, Paris, France, April 7-9, 1997,Proceedings, 1997, pp. 137–158.
10. M. Bellare and P. Rogaway, "Provably secure session keydistribution: the three-party case," in Proceedings of theTwenty-Seventh Annual ACM Symposium on Theory ofComputing, 29 May-1 June 1995, Las Vegas, Nevada,USA, 1995, pp. 57–66.
11. M. S. Dousti and R. Jalili, "Forsakes: A forward-secureauthenticated key exchange protocol based on symmetrickey-evolving schemes," Advances in Mathematics ofCommunications (AMC), vol. 9, no. 4, pp. 471–514,2015.
12. D. Konidala, C. Y. Yeun, and K. Kim, "A secure andprivacy enhanced protocol for location-based services inubiquitous society," in IEEE Global TelecommunicationsConference, 2004. GLOBECOM '04. IEEE, 2004.
13. M. Lee, J. Kim, S. Park, J. Lee, and S. Lee, "A secureweb services for location-based services in wireless networks," in Lecture Notes in Computer Science.Springer Berlin Heidelberg, 2004, pp. 332–344.
14. R. Dojen, A. Jurcut, T. Coffey, and C. Gyorodi, "Onestablishing and fixing a parallel session attack in asecurity protocol," in Intelligent Distributed Computing,Systems and Applications, C. Badica, G. Mangioni,V. Carchiolo, and D. D. Burdescu, Eds. Berlin,Heidelberg: Springer Berlin Heidelberg, 2008, pp.239–244.
15. "Scyther tool," https://people.cispa.io/cas.cremers/scyther/, last accessed: 2019-08-06.
16. C. Cremers and S. Mauw, Operational Semanticsand Verification of Security Protocols. Springer BerlinHeidelberg, 2012. [Online]. Available: https://doi.org/10.1007/978-3-540-78636-8
17. G. Lowe, "A hierarchy of authentication specification," in10th Computer Security Foundations Workshop (CSFW'97), June 10-12, 1997, Rockport, Massachusetts, USA,1997, pp. 31–44.

AUTHORS PROFILE

Mahesh Kum K.M is a full-time research scholar in the department of computer science and engineering at Siddaganga Institute of Technology Tumkur, India. His work focuses specifically on forward-security and private information retrieval schemes for location-based service application in the information security domain. He obtained his B.E and M.Tech degree from the VTU Belagavi in the year 2009 and 2011 respectively. He has 6 years of research and 3 years of IT industry experience.

Pradeep R is a full-time research scholar in thedepartment of computer science and engineering atSiddaganga Institute of Technology Tumkur, India. Hiswork focuses specifically on formal verification ofsecurity protocols in the information security domain.He obtained his B. E and M. Tech degree from theVTU Belagavi in the year 2012 and 2014 respectively.He has 2 years of research and 2 years of Academicexperience.

Dr. N.R. Sunitha is a professor in the departmentof computer science and engineering at SiddagangaInstitute of Technology Tumkur, India. Her workfocuses specifically on forward-security and digitalsignature schemes for banking and finance applicationin the information security domain. She has 22years of research and academics experience and has62 international conference articles, 20 internationaljournals, 12 book chapters, 5 sponsored projects ofvarious industries and 6 patents (filed) 1 (awarded)to her credits. She is guiding 5 PhD students and 1student has been awarded a PhD.