Chapter 1

Introduction

Fog computing is defined as an augmentation of cloud computing and its services to the edge of the network, i.e., in vicinity of the user. Just like cloud, fog computing also offers data, storage and usage of various other platform and application based services. The main objective is that fog computing along with providing all benefits of cloud, also fills the holes in remote data centers and IoT devices. Apart from this, it also provides various perks including enhanced security, decreased bandwidth and also comparably lesser latency. Thus, fog computing looks like a promising technology for various IoT services. Other advantages of the same, include better real time interaction and local awareness support for mobility. Fig. 1 illustrates a typical fog computing based IoT environment.

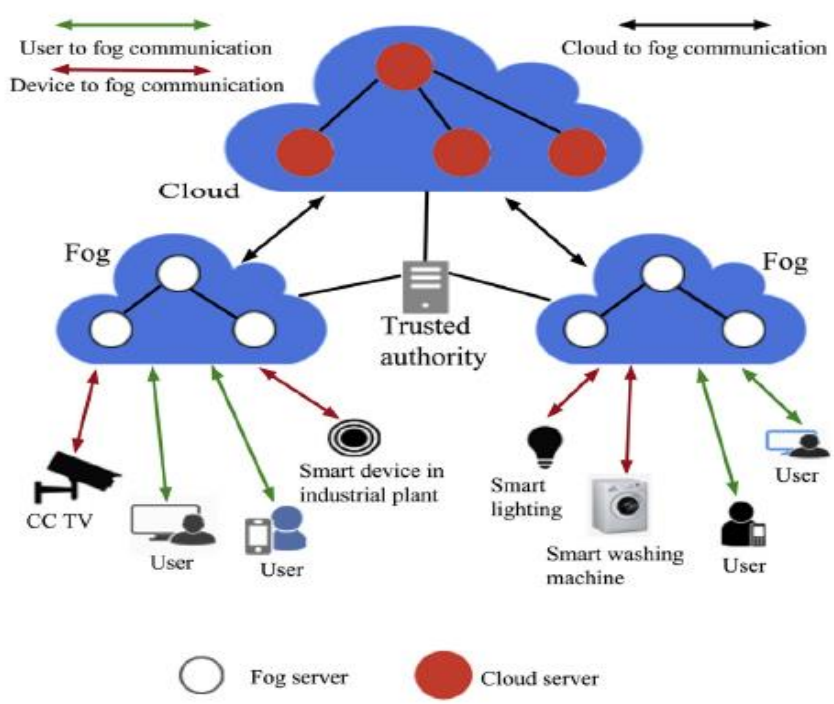

Fig. 1.1 A typical model of fog computing environment [1]

There are many users who try to access data of the smart devices without a long delay. Since, fog computing is an augmentation of cloud computing it also inherits the security and privacy challenges of the latter, which in turn causes some serious security issues to be tackled in fog computing environment. For example, spoofing, impersonation, insider attack, password guessing and many more to mention. Thus, this calls for well-built key management and user authentication scheme to work over all secure as well as insecure channels. Hence, only a legitimate user could gain access to system and valuable information. This report covers deep analysis of one such scheme SAKA-FC developed by M. Wajid et al. [2], shows security flaws after it's cryptanalysis and proposes a new User Authentication and Key Management scheme for the discussed environment.

This report also covers informal security analysis of the proposed scheme, it's testing on Automated Validation of Internet Security Protocols and Applications (AVISPA) and also it's communication cost and compares these aspects with the studied scheme, SAKA-FC.

1.1 Literature review

Hu et al. [13] had proposed various schemes for user identity authentication, data encryption and also data integrity thus to satisfy confidentiality, integrity and availability for face identification and face resolution. Biometric security mechanism for face images, using visual cryptography and zero watermarking is developed by Abdul et al. [14]. Stojmenovic et al. [1] discussed the advantages of fog computing and its applications, such as smart traffic lights in vehicular networks, smart grid and software defined networks. They further discussed the security and privacy issues as per the current fog computing environment.

A privacy-preserving de-duplication technique for efficient ownership management in fog computing is being designed by Koo et al. [15]. Also, Wang et al. [16] presented a technique for anonymous and secure aggregation scheme in fog-based public cloud computing in which a fog node aggregates the data from terminal nodes, and then forwards the aggregated data to the public cloud server. Hu et al. [17] surveyed various fog computing model architectures and discussed the key technologies, applications, challenges and open issues related to fog computing.

Keeping in view of the previous study done in the related field, authentication protocols in cloud computing and also to erase the security loopholes, M. Wajid et al [1] proposed a new three-factor key management and authentication protocol which is suitable for fog computing environment, called SAKA-FC applies user password and biometrics along with the mobile device as the three factors in the user authentication protocol. In addition, the fuzzy extractor technique was also applied there for local biometric verification of a legal user by his/her mobile device. In SAKA-FC, the lightweight operations, such as one-way cryptographic hash function and XOR operation are used for the resource-constrained devices.

But after deep study of SAKA-FC, it was found there were few security flaws in that scheme, those flaws are being discussed in detail in later part of this report.

1.2 System Model

The fog computing network model used in SAKA-FC and our proposed scheme is shown in Fig. 1. The above given model, assumes that there are initially cloud servers CS, fog servers FS, smart devices are being deployed in the network and there are users who try to access the smart device information.

Following type of communications take place in the environment:

1. User to fog server communication.
2. Device to fog server communication.
3. Cloud server to fog server communication.

Thus for these communications, secret keys are needed for their secure communication on insecure channel where otherwise an adversary can easily tamper with the data. Thus, to handle such issue, user needs to authenticate to the smart device with the help of fog server. After this successful authentication, user and smart device calculate their session key for their secure communication. Also initially a trusted agency (TA) is responsible for registration of smart devices, fog and cloud servers.

1.3 Security Goals

Following security requirements are being kept in mind while developing the proposed scheme:

- ***Authentication:*** It makes authentication of smart devices, users, cloud servers and fog servers before allowing access to a restricted resource, or revealing important information.
- ***Integrity:*** By integrity, the message or the entity under consideration must not be altered during communication.
- ***Confidentiality:*** Confidentiality (privacy) means the information must not be disclosed to unauthorized entities except those who are authorized to use it.

- ***Availability:*** The essential network services should be available to the authorized users or smart devices even under Denial-of-Service (DoS) attacks.
- ***Non-repudiation:*** It prevents an unauthorized entity from hiding his/her activities.
- ***Authorization:*** It assures that only the authorized smart devices can provide information to network services.

1.4 Organization of Report

Report is organized in the following manner: Chapter 2 contains preliminaries of ECC (Elliptic Curve Cryptography). Chapter 3 gives a brief review of Wajid et. al. scheme SAKA-FC [1] along with it's security flaws. Chapter 4 contains the detailed discussion of the proposed scheme. Chapter 5 contains the informal security analysis of the proposed scheme along with its simulation results on AVISPA. Chapter 6 contains detailed cost comparison of proposed scheme with Wajid et. al. scheme SAKA-FC [1] and other previous schemes. Finally, chapter 7 concludes the report.

Chapter 2

Preliminaries

The studied and the proposed scheme has extensive use of ECC. It's basic knowledge, computational problems and benefits are being discussed in the following chapter.

2.1 Elliptic Curve Cryptography (ECC)

Neal Koblitz and Victor Miller anticipated ECC in 1985 [8]. Later on it was adopted in public key cryptography system. Due to hardness of ECC, it has rapidly risen as the standard organized and inexpensive public key cryptosystem which is secure, proficient and efficient in terms of computation and communication overhead.

2.2 Benefits of ECC:

Following are the briefly discussed advantages of using ECC:

- As compared to other similar cryptosystems ECC uses smaller key size. For example RSA cryptosystem requires 1024 bits key, contrasting to which ECC just requires 160 bits key to ensure same security level [8].

- Solving the ECC based computational problem such as elliptic curve discreet logarithm problem (ECDLP) within polynomial time is impossible and intricate [9]. Approximately $\sqrt{p}$ steps are required to solve the ECC computational problem, where p is the cardinality of that elliptic curve and is a large prime no [10].

- ECC has smaller and compressed message size to run ECC-based protocols due to smaller key size. Faster operations allows less power consumption and hence, ECC-based protocols are suitable for the resource constrained devices like IoT nodes [11].

2.3 Definition:

A typical elliptic curve is being defined by the following equation:

$$y^2 \bmod q = (x^3+ax+b) \bmod q \qquad \ldots(1)$$

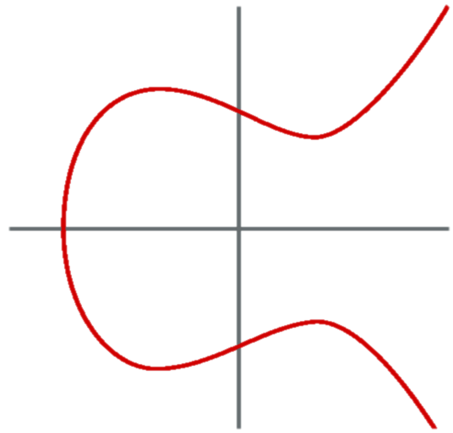

Fig. 2.1 Elliptic curve of equation $y^2 = x^3 – x + 1$

This equation defines the elliptic curve E/F_q on a prime finite field F_q where, $(a, b) \epsilon F_q$, $(4a^3 + 27b^2) mod\ q \neq 0$ and elliptic curve additive group G_q is defined as $\{(x, y): x, y \epsilon E/F_q \}$ U $\{O\}$ and $\{(x, y) q\}$ where O is called *point of infinity*.

2.4 Basic Group Operations:

Following group operations are commonly being performed on ECC.

- ***Point addition:*** Any two points for example P, Q that are on the elliptic curve of eq. *1* then, $P + Q = R$, where the straight-line joining P and Q intersects the curve is at $–R$, which is reflection of R with respect to x-axis [8].
- ***Point of subtraction:*** If, P, Q be the points on the elliptic curve of eq. *1* where $P = -Q$, and $P + P = P – P = O$ then, the intersecting line of the curve and the points P and Q join at an abstract point O (point of infinity) [9].
- ***Point doubling*:** Addition of a point P on the elliptic curve with the same point to get a new point Q on the same curve in equation *1* such that $2P=Q$ which is the point of reflection of the point of intersection of the tangent line at P with respect to x-axis [10].
- ***Scalar point multiplication:*** For any point on elliptic curve such that $t.P= +P+P+P+\ldots P$ (t times) $=\Sigma_i^n P$, where $t \epsilon Z_p^*$ and t is a scalar number in the cyclic group G_q [11].

2.5 Computational Problems of ECC:

Following computational problems of ECC are commonly used.

- ***Elliptic curve discrete logarithm problem (ECDLP)*:** Given that, two points *P and Q* lie on $E_q(a, b)$ such that, $P,Q \epsilon G_p$ and $Q=k.P$ where $k \epsilon Z_p^*$. In polynomial time it is hard to find k if Q is given [12].
- ***Computational Diffie-Helmen Problem (CDHP)*:** For any random occurrences *a, b* and *P* finding *a, b* is hard when the value of *abP* is known where $(P, aP, bP) \epsilon E/F_q$ and $a, b \epsilon Z_p^*$ [11].

Chapter 3

Cryptanalysis of Wajid et. al. scheme

This chapter discusses a brief review of Wajid et al. scheme SAKA-FC and also discusses its security flaws by doing its cryptanalysis.

3.1 Review of Wajid et al. scheme

The secure key management and user authentication scheme developed by Wajid et. al. has 8 phases:

i. Pre-Deployment phase.
ii. Key management phase.
iii. User registration phase.
iv. Login phase.
v. Authentication and key agreement phase.
vi. Password and biometric update phase.
vii. New device addition phase.
viii. Mobile Device revocation phase.

3.1.1 Pre Deployment Phase:

i. *Registration of smart devices:*

The TA picks a unique real identity ID_k and its corresponding temporary identity TID_k for each smart device D_k and calculates its pseudo-identity $RID_k = h(K // ID_k)$ where the TA's secret key is K . TA further calculates the temporal credential $TC_k = h(K // RTS_k // ID_k)$ of each D_k , where RTS_k is D_k's registration timestamp. In addition, TA computes a polynomial share $T_{mn}\, F(TID_k, y) = \sum^{t}_{m,n=0} [\, a_{m,n} (TID_k)^m \,]\, y^n$, which is clearly a univariate polynomial of the same degree t. Finally, the TA stores the credentials $\{RID_k, TID_k, TC_k, F(TID_k, y)\}$ in D_k 's memory prior to the placement of smart devices.

Table 3.1 Symbol table of Wajid et al. scheme [1]

Symbols	Abbreviations
U_i, MD_i	*i*th user and his/her mobile, respectively
D_k	*kth* smart device
TA	Trusted authority
ID_i, PW_i, BIO_i	U_i*'s* identity, password and biometric, respectively
FS_j, ID_j	*j*th fog server and its identity
CS_l, ID_l	*l*th cloud server and its identity
RID_i, RID_k	Pseudo identities of U_i and D_k, respectively
RID_j, RID_l	Pseudo identities of FS_j and CS_l, respectively
TID_i, TID_k	Temporary identities of U_i and D_k, respectively
TID_j, TID_l	Temporary identities of FS_j and CS_l, respectively
RTSi, *RTSk*	Registration timestamps of U_i and D_k, respectively
RTS_j, RTS_l	Registration timestamps of FS_j and CS_l, respectively
TS_u, TS_f , TS_k	Current timestamps
ΔT	Maximum transmission delay associated with a message
σ_i	Biometric secret key of U_i for BIO_i
$h(\cdot)$	Cryptographic one-way hash function
$E_p(a, b)$	$E_p(a, b) : y^2 = x^3 + ax + b$ (mod *p)* be an elliptic curve over prime field (Galois field) GF(p) (= Z_p), where *p* is a prime, $a, b \in Z_p^*$ with $4a^3 + 27b^2 \neq 0$ (mod p).
k.G	Elliptic curve point (ECC) point multiplication, $k \in Z_p^*$ and $G \in E_p(a, b)$
SK_{ik} (= SK_{ki})	Session key between U_i and D_k
$\|$, $\oplus$	Concatenation & bitwise XOR operations, respectively
$\Rightarrow, \rightarrow$	Secure and insecure channels, respectively

ii. *Registraiton of fog servers:*

Similar process takes place with the fog servers and they store $\{\{ RID_i \mid i = 1, 2, \ldots n_u \}, RID_j, TID_j , TC_j, F(TID_j, y)\}$

iii. *Registration of cloud servers:*

Similar process takes place with the cloud servers and they store *{{ RID_j |j = 1, 2, ... n_f }, RID_l, TID_l , TC_l, $G(TID_l, y)$}*

3.1.2 Key management phase:

(a) Key management between D_k and FS_j	
D_k	FS_j
Generate r_1 and TS_1.	
Calculate $r_1' = h(r_1 \|\| TC_k \|\| TS_1)$.	Check timeliness of TS_1.
$\xRightarrow{\{TID_k, r_1', TS_1\}}$	
	If valid, generate TS_2 and r_2. Calculate $F(TID_j, TID_k)$, $AA_j = h(r_2 \|\| TC_j) \oplus h(F(TID_j, TID_k) \|\| r_1' \|\| TS_2)$,
Check timeliness of TS_2.	$K_{jk} = h(F(TID_j, TID_k) \|\| r_1' \|\|$
Compute $F(TID_k, TID_j)$, $h(r_2 \|\| TC_j)$	$h(r_2 \|\| TC_j) \|\| TS_2)$, $BB_j = h(K_{jk} \|\| TS_2)$.
	$\xLeftarrow{\{TID_j, AA_j, BB_j, TS_2\}}$
$= AA_j \oplus h(F(TID_k, TID_j) \|\| r_1' \|\| TS_2)$, $K_{kj} = h(F(TID_k, TID_j) \|\| r_1' \|\| h(r_2 \|\| TC_j) \|\| TS_2)$, $BB_j' = h(K_{kj} \|\| TS_2)$.	
Check if $BB_j' = BB_j$?	
Both D_k and FS_j store the key K_{kj} $(= K_{jk})$.	

(b) Key management between FS_j and CS_l	
FS_j	CS_l
Generates r_3 and TS_3.	
Calculate $r_3' = h(r_3 \|\| TC_j \|\| TS_3)$.	Check timeliness of TS_3.
$\xRightarrow{\{TID_j, r_3', TS_3\}}$	
	If valid, generate TS_4 and r_4. Calculate $G_{j,l}(TID_l, TID_j)$, $CC_l = h(r_4 \|\| TC_l) \oplus h(G_{j,l}(TID_l, TID_j) \|\| r_3' \|\| TS_4)$,
Verify timeliness of TS_4.	$K_{lj} = h(G_{j,l}(TID_l, TID_j) \|\| r_3' \|\|$
Compute $G_{j,l}(TID_j, TID_l)$,	$h(r_4 \|\| TC_l) \|\| TS_4)$.
	$\xLeftarrow{\{TID_l, CC_l, DD_l, TS_4\}}$
$h(r_4 \|\| TC_l) = CC_l \oplus h(G_{j,l}(TID_j, TID_l) \|\| r_3' \|\| TS_4)$, $K_{jl} = h(G_{j,l}(TID_j, TID_l) \|\| r_3' \|\| h(r_4 \|\| TC_l) \|\| TS_4)$, $DD_l' = h(K_{jl} \|\| TS_4)$.	
Verify if $DD_l' = DD_l$?	
Both FS_j and CS_l store the key K_{jl} $(= K_{lj})$.	

Fig. 3.1 Summary of overall key management phase between *Di*, *FSj* and FS_j, CS_l [1]

3.1.3 User Registration:

U_i/MD_i	TA
Pick s and ID_i.	
Calculate $RID_i = h(s \|\| ID_i)$.	
Pick private key $d_i \in Z_p^*$.	Calculate
Compute public key $P_i = d_i.G$.	$TC_i = h(RID_i \|\| K \|\| RTS_i)$,
$\xRightarrow{\{RID_i, P_i\}}$	$\{h(TC_j) \| j = 1, 2, \cdots, n_f\}$.
	$\xLeftarrow{\{TC_i, \{(TID_j, h(TC_j)) \| j=1,2,\cdots,n_f\}\}}$
Select password PW_i.	
Input biometric BIO_i.	
Compute $Gen(BIO_i) = (\sigma_i, \tau_i)$,	
$TC_i^* = TC_i \oplus h(ID_i \|\| \sigma_i)$,	
$d_i^* = d_i \oplus h(ID_i \|\| PW_i \|\| \sigma_i)$,	
$RID_i^* = RID_i \oplus h(d_i \|\| \sigma_i)$,	
$RPB_i = h(ID_i \|\| TC_i \|\| PW_i \|\| \sigma_i)$,	
$TC_j^* = h(TC_j) \oplus h(RID_i \|\| \sigma_i)$,	
for $j = 1, 2, \cdots, n_f$.	
Store $\{RID_i^*, d_i^*, TC_i^*, RPB_i$,	
$\{(TID_j, TC_j^*) \| j = 1, 2, \cdots, n_f\}$, P_i, τ_i,	
$Gen(\cdot)$, $Rep(\cdot)$, $h(\cdot)$, $et\}$	
in MD_i's memory. Delete s, RID_i, d_i,	
TC_i, $\{h(TC_j) \| j = 1, 2, \cdots, n_f\}$ from MD_i.	

Fig 3.2. Summary of overall User registration process [1]

3.1.4 User Login and Authentication Phase

U_i/MD_i	FS_j	D_k
Inpute ID_i, PW_i and BIO'_i. Compute $\sigma'_i = Rep(BIO'_i, \tau_i)$, $TC_i = TC^*_i \oplus h(ID_i \| \sigma'_i)$, $d_i = d^*_i \oplus h(ID_i \| PW_i \| \sigma'_i)$, $RID_i = RID^*_i \oplus h(d_i \| \sigma'_i)$ $RPB_i = h(ID_i \|TC_i \| PW_i \| \sigma'_i)$, Check if $RPB'_i = RPB_i$? If so, Input TID_j, RID_k, & fetch TC^*_j. Calculate $h(TC_j) = TC^*_j \oplus h(RID_i \| \sigma'_i)$, Generate TS_u and r_u. Compute $R_u = r_u.G$ as $a_u = d_i + r_u \pmod p$, $RID'_i = RID_i \oplus h(h(TC_j) \| TS_u)$, $E_u = h(TC_i \|d_i \| TS_u) \oplus h(h(TC_j) \| RID_i)$, $F_u = RID_k \oplus h(h(TC_j) \|TS_u)$. $Msg_1 = \langle RID'_i, R_u, a_u, E_u, F_u, TS_u \rangle$ $\longrightarrow$ (to FS_j via open channel)	Check if $\lvert TS_u - TS^*_u \rvert \leq \Delta T$? If so, calculate $RID_i = RID'_i \oplus h(h(TC_j) \| TS_u)$. Check if $a_u.G = P_i + R_u$? If so, generate r_f, TS_f. Compute $K_{uf} = r_f.R_u = (r_u r_f).G$, $h(TC_i \| d_i \|TS_u) = E_u \oplus h(h(TC_j) \| RID_i)$, $P_f = r_f.G$, $RID_k = F_u \oplus h(h(TC_j) \|TS_u)$, $RID^*_i = h(RID_i) \oplus h(K_{jk} \| RID_k \|TS_f)$, $RID^*_k = RID_k \oplus h(K_{jk} \| TS_f)$, $G_j = h(K_{jk} \| RID_k \|TS_f) \oplus h(K_{uf} \|h(TC_i \|$ $d_i \|TS_u) \| h(RID_i))$, $H_j = h(h(RID_i) \| RID_k \|G_j \| P_f \| TS_f)$. $Msg_2 = \langle RID^*_i, RID^*_k, G_j, H_j, P_f, TS_f \rangle$ $\longrightarrow$ (to D_k via open channel)	Check $\lvert TS_f - TS^*_f \rvert \leq \Delta T$? If so, calculate $RID_k = RID^*_k \oplus h(K_{jk} \|TS_f)$, $h(RID_i) = RID^*_i \oplus h(K_{jk} \| RID_k \| TS_f)$. $H'_j = h(h(RID_i) \|RID_k \| G_j \| \|P_f \|TS_f)$, check $H'_j = H_j$, if so, compute $I_j = G_j \oplus h(K_{jk} \| RID_k \| TS_f)$ $= h(K_{uf} \|h(TC_i \| d_i \|TS_u) \| h(RID_i))$. Generate r_k, TS_k,
Check $\lvert TS_k - TS^*_k \rvert \leq \Delta T$? If so, calculate $RID_k = RID^{**}_k \oplus h(h(RID_i) \| TS_k)$, $h(TC_k \|r_k) = M_k \oplus h(RID_k \| h(RID_i) \| TS_k)$, $K_{uf} = r_u.P_f$, session key SK_{ik} $= h(h(K_{uf} \|h(TC_i \| d_i \|TS_u) \| h(RID_i))$ $\|h(TC_k \|r_k) \|TS_k)\ (= SK_{ki})$, $N'_k = h(SK_{ik} \| P_f \|TS_k)$, Verify if $N'_k = N_k$? If so, store SK_{ik}.		Calculate $RID^{**}_k = RID_k \oplus h(h(RID_i) \| TS_k)$, session key $SK_{ki} = h(I_j \|h(TC_k \| r_k) \|TS_k)$, $M_k = h(TC_k \|r_k) \oplus h(RID_k \| h(RID_i) \|TS_k)$, $N_k = h(SK_{ki} \|P_f \|TS_k)$. $Msg_3 = \langle RID^{**}_k, M_k, N_k, P_f, TS_k \rangle$ $\longleftarrow$ (to U_i via open channel)
Both U_i and D_k store the common session key $SK_{ik}\ (= SK_{ki})$.		

Fig. 3.3 Summary of user login and authentication process [1]

3.1.5 Password and Biometric Update Phase

i. U_i first provides his/her identity ID_i and old password PW^o_i at the interface of MD_i , and also imprints his/her old biometrics BIO^o_i at the sensor of MD_i. MD_i then retrieves the old biometric key $\sigma^o_i = Rep(BIO^o_i, \tau_i)$ with the restriction that the Hamming distance between the actual biometrics BIO_i at the registration time and just entered BIO^o_i is less than or equal to e_t. Moreover, MD_i calculates $TC_i = TC^*_i \oplus h(ID_i \| \sigma^o_i)$, $di = d^*_i \oplus h(ID_i \| PW^o_i \| \sigma^o_i)$, $RID_i = RID^*_i \oplus h(d_i \| \sigma^o_i)$, $h(TC^o_j) = TC^o_j \oplus h(RID_i \| \sigma^o_i)$ for $j = 1, 2, \ldots, n_f$, and $RPB^o_i = h(ID_i \| TC_i \| PW^o_i \| \sigma^o_i)$. MD_i checks whether $RPB^o_i = RPB_i$ holds. If it holds, U_i passes both old password and biometric verification, and can now proceed for new password and biometric update procedure. Otherwise, the procedure exits.

ii. U_i inputs new password PW^n_i and also imprints new biometric BIO^n_i at the sensor of MD_i . If U_i does not want to update biometrics, he/she may still keep old biometrics BIO^o_i . In such a case, BIO^n_i is considered as BIO^o_i . Otherwise, MD_i calculates $Gen(BIO^n_i) = (\sigma^n_i, \tau^n_i)$.

iii. MD_i further continues to calculate $TC^n_i = TC_i \oplus h(ID_i \| \sigma^n_i)$, $d^n_i = d_i \oplus h(ID_i \| PW^n_i \| \sigma^n_i)$, $RID^n_i = RID_i \oplus h(d_i \| \sigma^n_i)$, $RPB^n_i = h(ID_i \| TC_i \| PW^n_i \| \sigma^n_i)$ and $TC^n_i = h(TC_j) \oplus h(RID_i \| \sigma^n_i)$ for $j = 1, 2, \ldots, n_f$. Finally, MD_i replaces RID^*_i, d^*_i, TC^*_i, $RPBi$, $\{TC^*_j \mid j = 1, 2, \ldots, n_f\}$ and τ_i with the newly computed RID^n_i, d^n_i, TC^n_i, RPB^n_i, $\{TC^n_i \mid j = 1, 2, \ldots, nf\}$ and τ^n_i, respectively. Thus, MD_i holds the information $\{RID^n_i, d^n_i, TC^n_i, RPB^n_i, \{TC^n_i \mid j = 1, 2, \ldots, nf\}, P_i, \tau^n_i, Gen(\cdot), Rep(\cdot), h(\cdot), et\}$ in its memory.

3.1.6 New smart device addition phase:

i. TA picks a unique identity ID_k^{new} and its corresponding temporary identity TID^{new}_k, and calculates its pseudo-identity $RID_k^{new} = h(K \| ID_k^{new})$ *of* D_k^{new} using the secret key K of TA.

ii. TA calculates temporal credential $TC_k^{new} = h(K \| RTS_k^{new} \| ID_k^{new})$ of D_k^{new}, where RTS_k^{new} is the registration timestamp of D_k^{new}. TA then computes the polynomial share $F(TID_k^{new}, y))$ using its previously generated symmetric bivariate polynomial $F(x, y) = \sum^t_{i,j=0} a_{i,j} x^i y^j \in GF(p)[x, y]$ of degree t, for D_k^{new}.

iii. TA stores $\{RID_k^{new}, TID_k^{new}, TC_k^{new}, F(TID_k^{new}, y)\}$ in the memory of D_k^{new} and then D_k^{new} is deployed.

3.1.7 Mobile device revocation phase:

i. U_i first picks a new random secret s^n and computes its pseudo-identity $RID^n_i = h(s^n \| ID_i)$ corresponding to his/her previously chosen ID_i. U_i also generates a new private key $d^n_i \in Z^*_i$ and calculates its corresponding new public key as $P_i^n = d_i^n . G$. U_i dispatched the registration request $\{RID_i^n, P_i^n\}$ to TA securely.

ii. After receiving $\{RID_i^n, P_i^n\}$ from U_i, TA calculates U_i *'s* temporal credential $TC_i^n = h(RID_i^n \| K \| RTS_i^n)$, where the new registration timestamp of Ui is RTS_i^n. After this task, TA dispatches the registration reply $\{TC_i^n, \{(TID_j, h(TCj)) \mid j = 1, 2, \ldots, n_f\}\}$ to U_i securely.

iii. After receiving registration reply from TA, U_i chooses a password PW_i of his/her choice, and inputs biometric BIO_i at the sensor of MD_i^n. MD_i^n generates $Gen(BIOi) = (\sigma i, \tau i)$ and calculates $TC_i^* = TC_i^n \oplus h(ID_i \| \sigma_i)$, $d_i^* = d_i^n \oplus h(ID_i \| PW_i \| \sigma_i)$, $RID_i^n = RID_i^n \oplus h(d_i^n \| \sigma_i)$

and $RPB_i^n = h(ID_i \,/\!/\, TC_i^{\,n} \,/\!/\, PW_i /\!/ \sigma_i)$. In addition, MD_i calculates $TC_i^* = h(TCj) \oplus h(RID_i^n \,/\!/\, \sigma_i)$, $(j = 1, 2, \ldots, n_f)$. Finally, MD_i stores $\{RID_i^*, d_i^*, TC_i^*, RPB_i^n, \{(TID_j, TC_j^*)|j = 1, 2, \ldots, n_f\}, P_i^n, \tau_i, Gen(\cdot), Rep(\cdot), h(\cdot), et\}$ in its memory, and deletes sn, $RIDni$, dni, $TCni$ and $\{h(TC_j)|_j = 1, 2, \ldots, n_f\}$ from its memory.

Finally, *TA* sends the required information to the deployed fog servers securely after successful completion of this phase.

3.2 Cryptanalysis of Wajid et. al. scheme SAKA-FC

After cryptanalysis of secure key management and user authentication scheme (SAKA-FC) developed by Wajid et. al., it was found that it has some security flaws, thus it's vulnerable to some security attacks which as discussed below supported by mathematical analysis along with some practical assumptions.

- **Fog Server Insider Attack:**

If adversary somehow gains access to one of the fog server then he could easily acquire the information stored in it and then calculating the session key would not be hard to calculate.

Session key (at smart device):

$$SK_{ki}=h(I_j||h(TC_k||r_k)||TS_k)$$

I_j is calculated at smart device as:

$$I_j=G_j+(K_{jk}||RID_k||TS_f)$$

Here, G_j, RID_k is calculated by FS_j itself in step 5, and TS_f is public, thus could be calculated at FS_j. $h(TC_k||r_k)$ is calculated at mobile device/ user end as:

$$h(TC_k||r_k)=M_k+(RID_k||h(RID_i)||TS_k)$$

Here, M_k is public, RID_k is known by FS_j, RID_i is also known as it's being stored by FS_j in step 1 and TS_k is public.

Thus, whole session key could be calculated if a fog server is being captured, thus posing a great security risk.

- **Denial of Service (DoS) attack:**

At all login and authentication steps in the above scheme, a time stamp is being sent along with the message. If an adversary intercepts the message and changes the time stamp than that message won’t be accepted and further repeating the message could lead to flooding of computing resources thus hindering any legitimate message request to be processed.

Chapter 4

Proposed scheme

This chapter discusses the proposed scheme for key management and user authentication in fog computing system. This scheme has 6 phases:

i. Pre-negotiation phase.
ii. Pre deployment phase.
iii. Key management phase.
iv. User registration phase.
v. User login phase.
vi. User authentication and key agreement phase.

Description of the symbols used in the proposed scheme are given in table 4.1.

4.1 Pre Negotiation phase:

Here, *TA* selects and sets system parameters and the steps being used are as follows:

i. A finite field F_q is selected over $q > . 2^{160}$.

ii. An elliptic curve $E_q(a, b)$ is selected where $E_q(a, b)$: $y^2 \bmod q = (x^3 + ax + b) \bmod q$ of order n over F_q, where $(4a^3 + 27b^2)\ 0 \bmod q$ and $a, b \in F_q$.

iii. A symmetric encryption/decryption algorithm is selected $E_k()/D_k()$ here k depicts the symmetric key.

iv. A generator P is selected of order n over $E_q(a, b)$.

v. Now *TA* publishes $\{E_q(a, b), E_k()/D_k(), F(x,y), P\}$ as public parameters.

vi. Each communicating party for eg. *SD, FS, CS, U, TA* select their private key as α, γ, υ and β computes public keys as $V_{SD} = \alpha.P$, $V_{FS} = \gamma.P$, $V_{CS} = \upsilon.P$ and $V_{TA} = \beta.P$ respectively.

Table 4.1 Symbols used in the proposed scheme and their meaning

	Abbreviations
SD	**Smart Device**
FS	Fog Server
CS	Cloud Server
U	User
TA	Trusted Agency
(α, V_{SD})	Smart device's private and public key
(γ, V_{FS})	Fog server's private and public key
(υ, V_{CS})	Cloud server's private and public key
(η, V_U)	User's private and public key
(β, V_T)	Trusted agency's private and public key
K_{SDT}	Shared key of smart device and trusted agency
K_{FST}	Shared key of fog server and trusted agency
K_{CST}	Shared key of cloud server and trusted agency
K_{UT}	Shared key of user and trusted agency
K_{FSSD}	Shared key of fog server and smart device
K_{FSCS}	Shared key of fog server and cloud server
ID_{SD}, ID_{FS}, ID_{CS}	Identity of smart device, fog server, cloud server respectively
ID_U	64 bit Identity of user
PW_U, B_U	64 bit Password and biometrics of user
$F(x,y)$	Symmetric bivariate polynomial
r_i	128 bit Random nonce
TS	32 bit time stamp
$h(.)$	160 bit output hash function
$E_K(.)$	128 bit encrypted message with key K
ΔT	Maximum delay in time interval of sending and receiving a message
LM	Login message sent by user to fog server

4.2 Pre deployment phase:

This phase allows TA to register the smart devices, fog servers and cloud servers before they are being deployed in the network.

4.2.1 Registration of smart device

Step 1: SD $\rightarrow$ TA: $E_{K_{SDT}}(H_{SD} \| A_{SD} \| R_1)$

Smart device selects it's private key α and calculates it's public key $V_{SD} = \alpha.P$. Further , it makes it's registration request by calculating it's and trusted agency's shared key $K_{SDT} = \alpha.V_T$, now it selects a random nonce $r_1 \in Z_p^*$ and further calculates R_1 as, $R_1 = r_1.V_{SD}$. It also calculates $H_{SD}=h(ID_{SD}||V_{SD})$ and $A_{SD}=h(H_{SD}||R_1||K_{SDT})$. After performing these operations, it encrypts H_{SD}, A_{SD}, R_1 *as* $E_{K_{SDT}}(H_{SD} \| A_{SD} \| R_1)$ and sends this securely to TA.

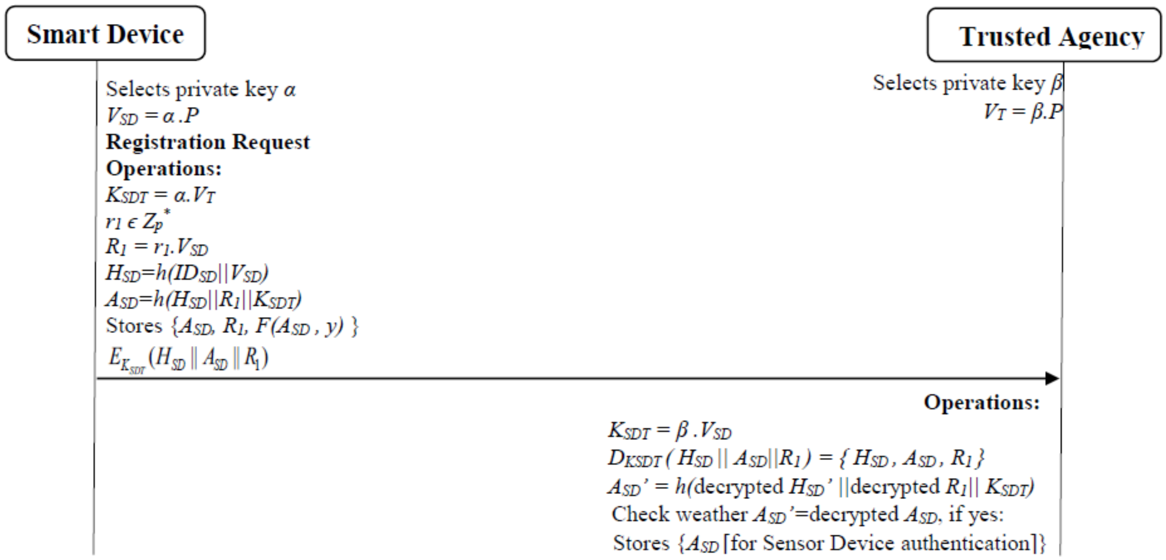

Fig. 4.1 Summary of smart Device registration

After receiving the registration request message, TA calculates their shared key as $K_{SDT} = \beta.V_{SD}$. It retrieves $\{H_{SD}, A_{SD}, R_1\}$ from the received message by using the K_{SDT} calculated. It further calculates $A_{SD}' = h($decrypted H_{SD}' ||decrypted R_1|| $K_{SDT})$ and checks if calculated A_{SD}'= received A_{SD}, if this satisfies then TA stores { A_{SD}} for smart device authentication for further purpose. Finally, smart device also stores $\{A_{SD}, R_1, F(A_{SD}, y)\}$.

4.2.2 Registration of fog server

Step 1: FS → TA: $E_{K_{FST}}(H_{FS} \| A_{FS} \| R_2)$

Fog server selects it's private key γ and calculates it's public key $V_{FS} = \gamma.P$. Further , it makes it's registration request by calculating it's and trusted agency's shared key $K_{FST} = \gamma.V_T$, now it selects a random nonce $r_2 \epsilon Z_p^*$ and further calculates R_2 as, $R_2 = r_2.V_{FS}$. It also calculates $H_{FS}=h(ID_{FS}||V_{FS})$ and $A_{FS}=h(H_{FS}||R_2||K_{FST})$. After performing these operations, it encrypts H_{FS}, A_{FS}, R_2 *as* $E_{K_{FST}}(H_{FS} \| A_{FS} \| R_2)$ and sends this securely to TA.

After receiving the registration request message, TA calculates their shared key as $K_{FST} = \beta.V_{FS}$. It retrieves $\{H_{FS}, A_{FS}, R_2\}$ from the received message by using the K_{FST} calculated. It further calculates $A_{FS}' = h($decrypted H_{FS}' ||decrypted R_2|| $K_{FST})$ and checks if calculated A_{FS}'= received A_{FS}, if this satisfies then TA stores { A_{FS}} for fog server authentication for further purpose and sends $\{A_{SD}\}$

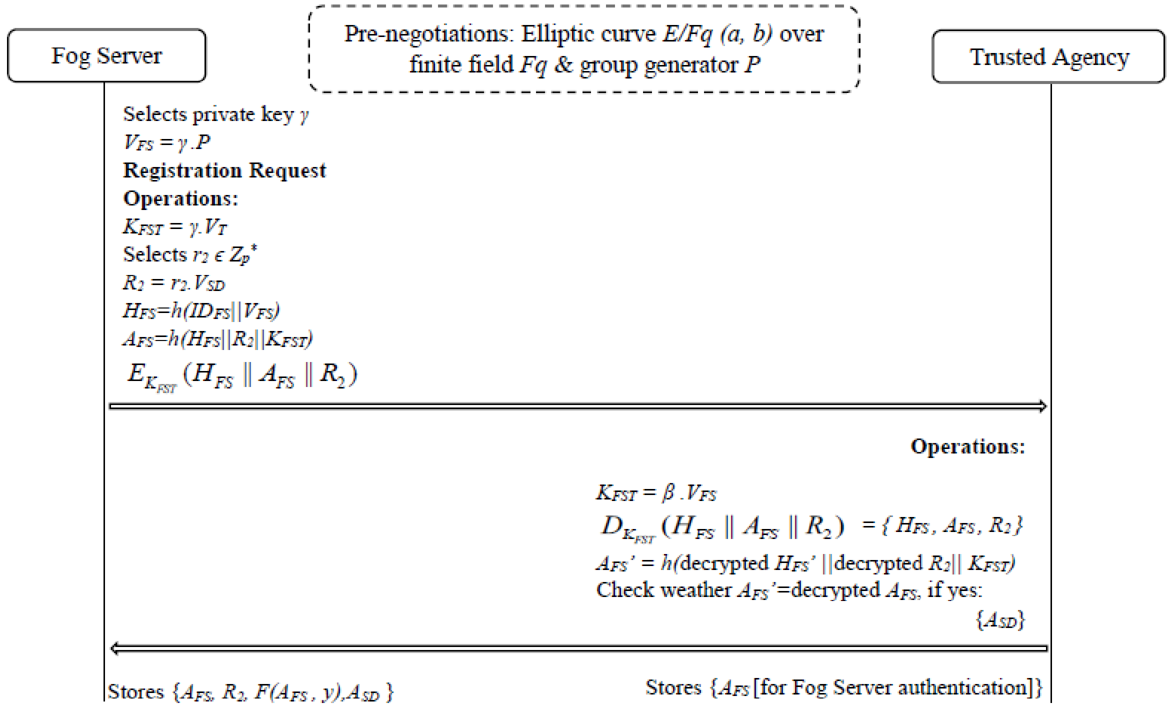

Fig. 4.2 Summary of fog server registration

calculated in step 4.2.1 back to fog server securely. Finally, fog server also stores { A_{FS}, R_2, $F(A_{FS}$, $y)$, A_{SD} }

4.2.3 Registration of cloud server

Step 1: CS ⟶ TA: $E_{K_{CST}}(H_{CS} \| A_{CS} \| R_3)$

Cloud server selects it's private key υ and calculates it's public key $V_{CS} = \upsilon .P$. Further, it makes it's registration request by calculating it's and trusted agency's shared key $K_{CST} = \upsilon.V_T$, now it selects a random nonce $r_3 \epsilon Z_p^*$ and further calculates R_3 as, $R_3 = r_3.V_{CS}$. It also calculates $H_{CS}=h(ID_{CS}||V_{CS})$ and $A_{CS}=h(H_{CS}||R_3||K_{CST})$. After performing these operations, it encrypts H_{CS}, A_{CS}, R_3 *as* $E_{K_{CST}}(H_{CS} \| A_{CS} \| R_3)$ and sends this securely to TA.

After receiving the registration request message, TA calculates their shared key as $K_{CST} = \upsilon .V_{CS}$. It retrieves $\{ H_{CS}, A_{CS}, R_3 \}$ from the received message by using the K_{CST} calculated. It further calculates $A_{CS}' = h($decrypted H_{CS}' ||decrypted R_3|| $K_{CST})$ and checks if calculated A_{CS}'= received A_{CS}, if this satisfies then TA stores { A_{CS}} for cloud server authentication for further purpose and sends {A_{FS}}

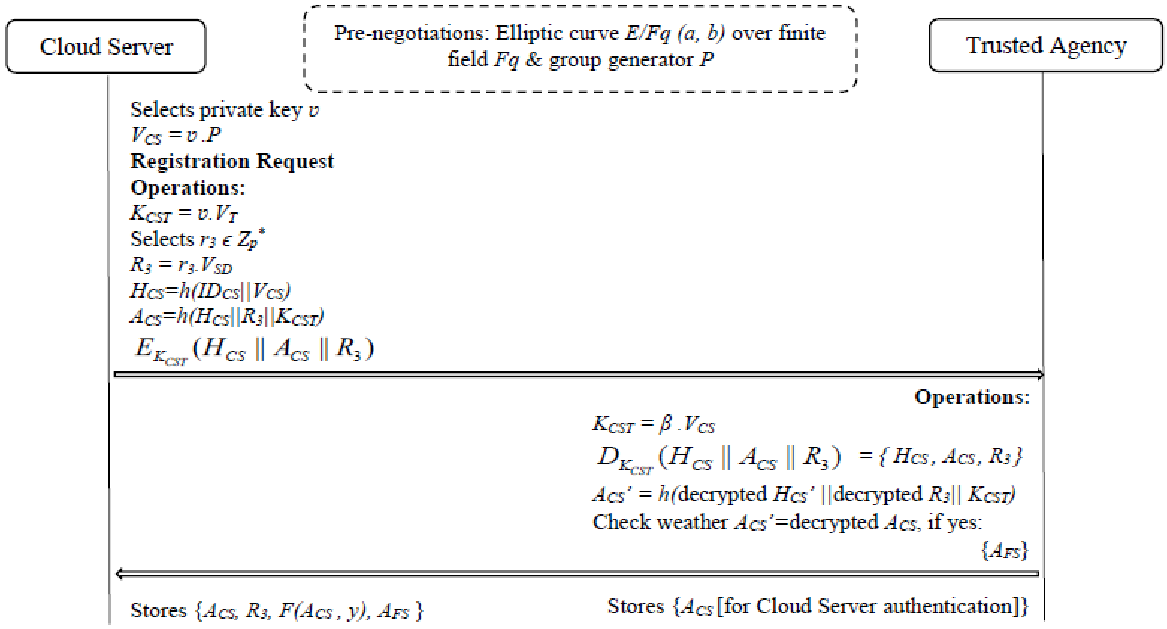

Fig. 4.3 Summary of cloud server registration

calculated in step 4.2.2 back to cloud server securely. Finally, fog server also stores { A_{CS}, R_3, $F(A_{CS}, y)$, A_{CS} }

4.3 Key management phase:

This phase helps to make secure communication between smart device and fog server, and also fog server and cloud server.

4.3.1 Key management between smart device and fog server

Smart device and fog server need to perform following steps to establish secret key between them for their secure communication.

Step 1: SD $\rightarrow$ FS: $\{r_4', TS_1\}$

Smart device selects a random nonce $r_4 \in Z_p^*$ and records the current time stamp TS_1 and calculates $r_4' = h(r_4||A_{SD}||TS_1)$. Now, it sends the message $\{r_4', TS_1\}$ to the fog server.

Step 2: FS $\rightarrow$ SD: $\{A_{FS}, r_5, BB_{FS}, TS_2\}$

On receiving the message, fog server records the current time stamp as TS_2 and checks if $|TS_2 - TS_1| <= et$. If the condition is satisfied, FS calculates another random nonce $r_5 \in Z_p^*$ and calculates $F($stored

A_{FS}, stored A_{SD}) and $K_a' = \gamma.V_{SD}$. After this it calculates Calculate $h(K_a'||r_5)$. Now, it calculates their secret key K_{FSSD} as $K_{FSSD}=h\ (F(A_{FS},A_{SD})||r_4'||h(K_a'||r_5\ )||TS_2)$.To check correctness of this key, it further calculates $BB_{FS}= h(K_{FSSD}||TS_2)$ and sends $\{A_{FS}, r_5, BB_{FS}, TS_2\ \}$ back to smart device.

On receiving the above message, smart device records the current time stamp TS' and Check if $|TS'-TS_2|<= et$, if yes then it calculates $F(A_{SD},A_{FS})$ and $K_a' = \alpha.V_{FS}$, further it calculates $h(K_a'||r_5)$ and finally obtains their secret key as $K_{FSSD}=h(F(A_{FS},A_{SD})||r_4'||h(K_a'||r_5)||TS_2)$. Lastly, it calculates $BB_{FS}'=h(K_{FSSD}||\ TS_2)$ and check if calculated $BB_{FS}' = received\ BB_{FS}$, if this criteria is satisfied then smart device has obtained the correct secret key.

4.3.2 Key management between fog server and cloud server

Fog server and cloud server need to perform following steps to establish secret key between them for their secure communication.

Step 1: FS $\rightarrow$ CS: $\{r_6', TS_3\}$

Fog server selects a random nonce $r_6 \in Z_p^*$ and records the current time stamp TS_3 and calculates $r_6' = h(r_6||A_{FS}||TS_3)$. Now, it sends the message $\{r_6', TS_3\}$ to the cloud server.

Step 2: CS $\rightarrow$ FS: $\{A_{CS}, r_7, BB_{CS}, TS_4\ \}$

On receiving the message, cloud server records the current time stamp as TS_4 and checks if $|TS_4-TS_3|<= et$. If the condition is satisfied, CS calculates another random nonce $r_7 \in Z_p^*$ and calculates $F($stored A_{FS}, stored $A_{CS})$ and $K_b' = v.V_{CS}$. After this it calculates Calculate $h(K_b'||r_7)$. Now, it calculates their secret key K_{FSCS} as $K_{FSCS}=h\ (F(A_{FS},A_{CS})||r_6'||h(K_b'||r_7\ )||TS_4)$.To check correctness of this key, it further calculates $BB_{CS}= h(K_{FSCS}||TS_4)$ and sends $\{A_{CS}, r_7, BB_{CS}, TS_4\ \}$ back to fog server.

On receiving the above message, smart device records the current time stamp TS' and Check if $|TS'-TS_2|<= et$, if yes then it calculates $F(A_{SD},A_{FS})$ and $K_a' = \alpha.V_{FS}$, further it calculates $h(K_a'||r_5)$ and finally obtains their secret key as $K_{FSSD}=h(F(A_{FS},A_{SD})||r_4'||h(K_a'||r_5)||TS_2)$. Lastly, it calculates $BB_{FS}'=h(K_{FSSD}||\ TS_2)$ and check if calculated $BB_{FS}' = received\ BB_{FS}$, if this criteria is satisfied then smart device has obtained the correct secret key.

4.4 User registration

Step 1: U → FS: $E_{K_{UFS}}(M_U \parallel H_U \parallel A_U \parallel R_8)$

User selects it's private key η, with which it calculates it's public key $V_U = \eta.P$. Further it calculates it's shared key with the fog server $K_{UFS} = \eta.V_{FS}$. After this, it generates it's registration request with the following operations. User choses it's identity ID_U, it's password PW_U and then imprints it's biometrics B_U. After this, it selects a random nonce $r_8 \in Z_p^*$ with which it calculated $R_8=r_8.P$, then $H_U=h(ID_U||V_U)$, $M_U=h(ID_U||PW_U||B_U)$ *and* $A_U=h(K_{UFS}||M_U||R_8)$. After completing these operations it encrypts $(M_U||H_U||A_U||R_8)$ with it's calculated shared key K_{UFS}.

Step 2: FS → U: $E_{K_{UFS}}(C_{FS} \parallel D_{FS})$

After receiving the registration request message, fog server calculates it's and user's shared key $K_{UFS}=\gamma.V_U$. After this it retrieves $\{M_U, H_U, A_U, R_8\}$ by decrypting it with the calculated shared key. It then calculates $A_U'=h(K_{UFS}||$decrypted $M_U||$decrypted $R_4)$ and checks if calculated $A_U'=A_U$, if the condition satisfies then the request is treated as legitimate. It now selects a random nonce $r_9 \in Z_p^*$ and $R_9=r_9.P$. Now it calculates $C_{FS}=h($decrypted $A_U||R_9)$ and $D_{FS}=h($decrypted $H_U||C_{FS}||$decrypted $R_8||K_{UFS})$ and sends $(C_{FS}|| D_{FS})$ by encrypting it with the calculated shared key K_{UFS}.

After receiving above message from fog server, it retrieves C_{FS} and D_{FS} checks for D_{FS} . If criteria holds true, it stores $\{M_U$ [To verify user's login], *DFS* [For user authentication]$\}$ and fog server also stores $\{D_{FS}$ [For user authentication] $\}$.

4.5 User login

Step 1: U → FS: **{*M', LM*}**

User inputs it's ID_U, PW_U and B_U . Now, it calculates $H_U=h(ID_U||V_U)$ and $M_{U'}=h($given $ID_U||$given $PW_U||$imprinted $B_U)$ and it checks if the calculated M_U'= stored M_U then it passes the security check and it's login request is forwarded to FS. *M'* is calculated as M'=h($PW_U||B_U$), now login message *LM* is calculated as $LM=h(M' \parallel K_{UFS}|| D_{FS})$ and $\{M', LM\}$ is sent to fog server.

After receiving the message, FS checks it for *LM' as* $LM'= h($received $M' \parallel K_{UFS}||$ stored $D_{FS})$, if it matches the received *LM t*hen user is logged in.

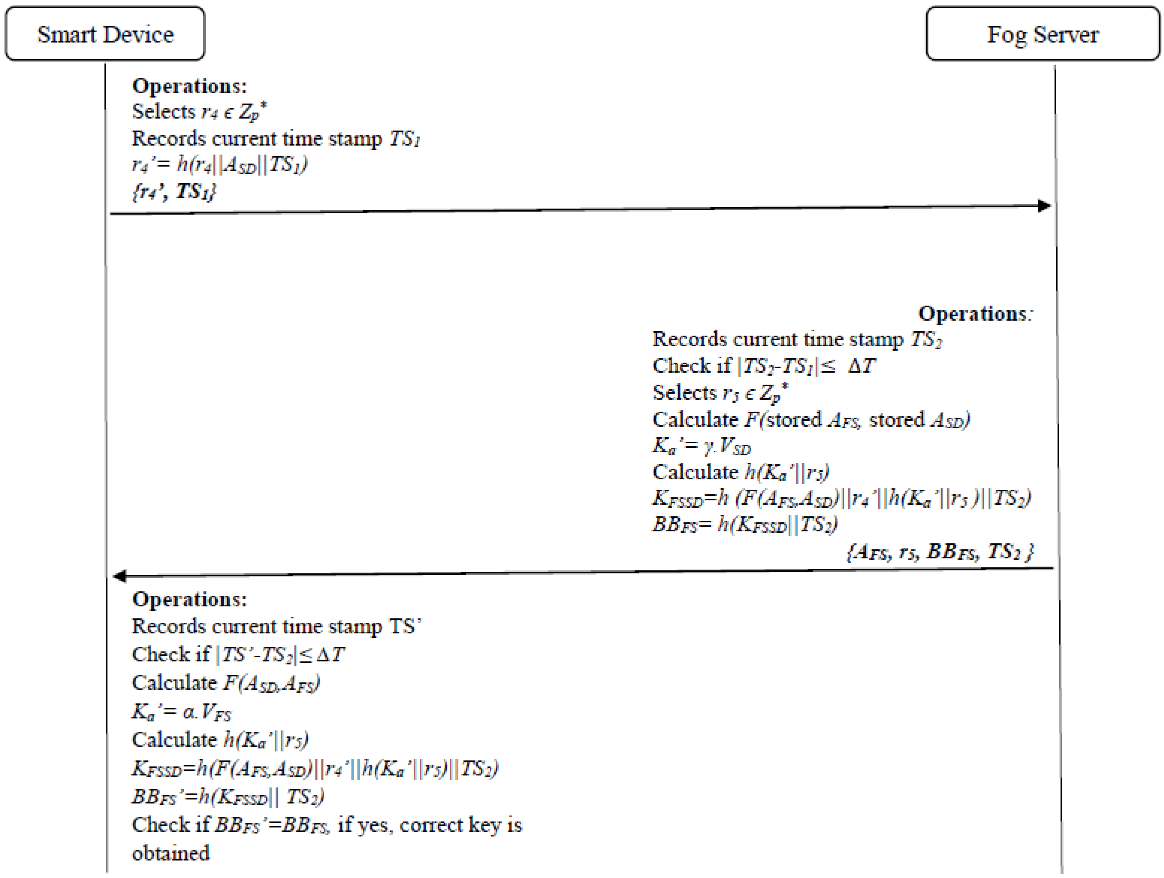

Fig. 4.4 Summary of key management between smart device and fog server

4.6 User Authentication and Key agreement

Step 1: U ⟶ FS: $\{M_{U1}, R_{10}, TS_5\}$

After being logged in, user selects $r_{10} \epsilon Z_p^*$ and calculates $R_{10} = r_{10}.\eta.P$. It also records the current time stamp TS_5 and calculates $M_{U1} = h(D_{FS}||R_{10}||TS_5||K_{UFS})$. Now it sends $\{M_{U1}, R_{10}, TS_5\}$ to fog server.

Step 2: U ⟶ FS: $\boldsymbol{\{H_U, M_{FS3}, R_{11}, R_{10}, TS_6\}}$

After receiving the authentication request from user, FS records the current time stamp TS_6 and checks $|TS_6 - TS_5| \leq \Delta T$, if yes then calculates $M_U' = h($stored $D_{FS}||$recieved $R_{10}||$recieved $TS_5||K_{UFS})$, now if calculated M_U'= received M_U then user is authenticated. Now, FS selects $r_{11} \epsilon Z_p^*$ and calculates $R_{11} = r_{11}.P$. It also calculates $M_{FS2} = h(H_U||ID_{FS}||A_{SD}||TS_6)$ and $M_{FS3} = h(M_{FS2}||R_{11}||R_{10}||K_{FSSD})$. Now it sends $\{H_U, M_{FS3}, R_{11}, R_{10}, TS_6\}$ to smart device.

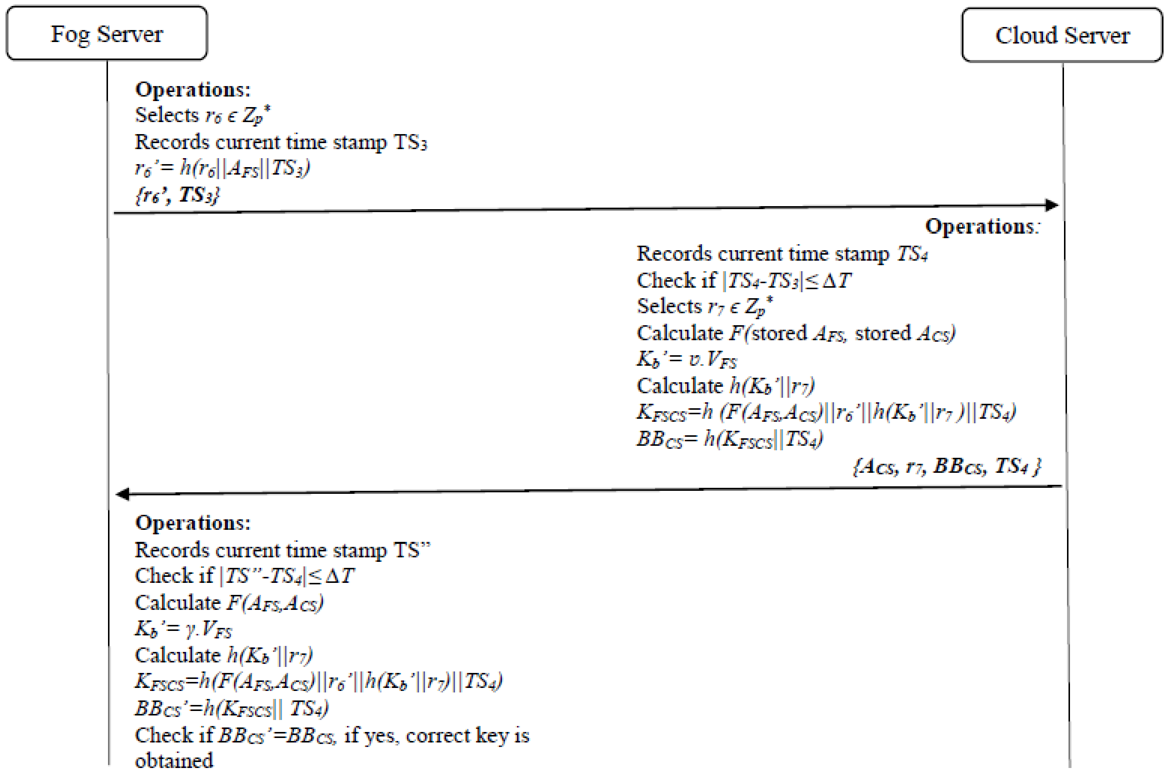

Fig. 4.5 Summary of key management between fog server and cloud server

Step3: FS ⟶ U: $\{M_{SK},K'',TS_7\}$

On receiving the message from FS, SD records the current time stamp TS_7 and checks $|TS_7\text{-}TS_6| \leq \Delta T$, if yes then M_{FS2}' is calculated as $M_{FS2}'=h($received $H_U||ID_{FS}||A_{SD}||$received $TS_6)$ and M_{FS3}' as $M_{FS3}'=h(M_{FS2}'||$received $R_{11}||$received $R_{10}||K_{FSSD})$, if calculated M_{FS3}'=received M_{FS3}, then fog server request is treated as legitimate and session key is calculated as $SK_{SDU}=\alpha.r_1.R_{10}=\alpha.r_1.r_{10}.\eta.P$ and $K''=r_1.V_{SD}$ and $M_{SK}=h(SK_{SDU}||R_{10}||TS_7)$. Now, $\{M_{SK},K'',TS_7\}$ *is sent back to the user.*

On receiving the above message, user records the current time stamp as TS'' and checks $|TS''\text{-}TS_7| \leq \Delta T\ t$, if yes then session key is calculated as $SK_{SDU}=r_{10}.\eta.K''=r_{10}.\eta.r_1.\alpha.P$, now to check it's correctness $M_{SK}'=h(SK_{SDU}||R_{10}||TS_7)$ is calculated if it matches the received M_{SK} then it is ensured that correct session key is established.

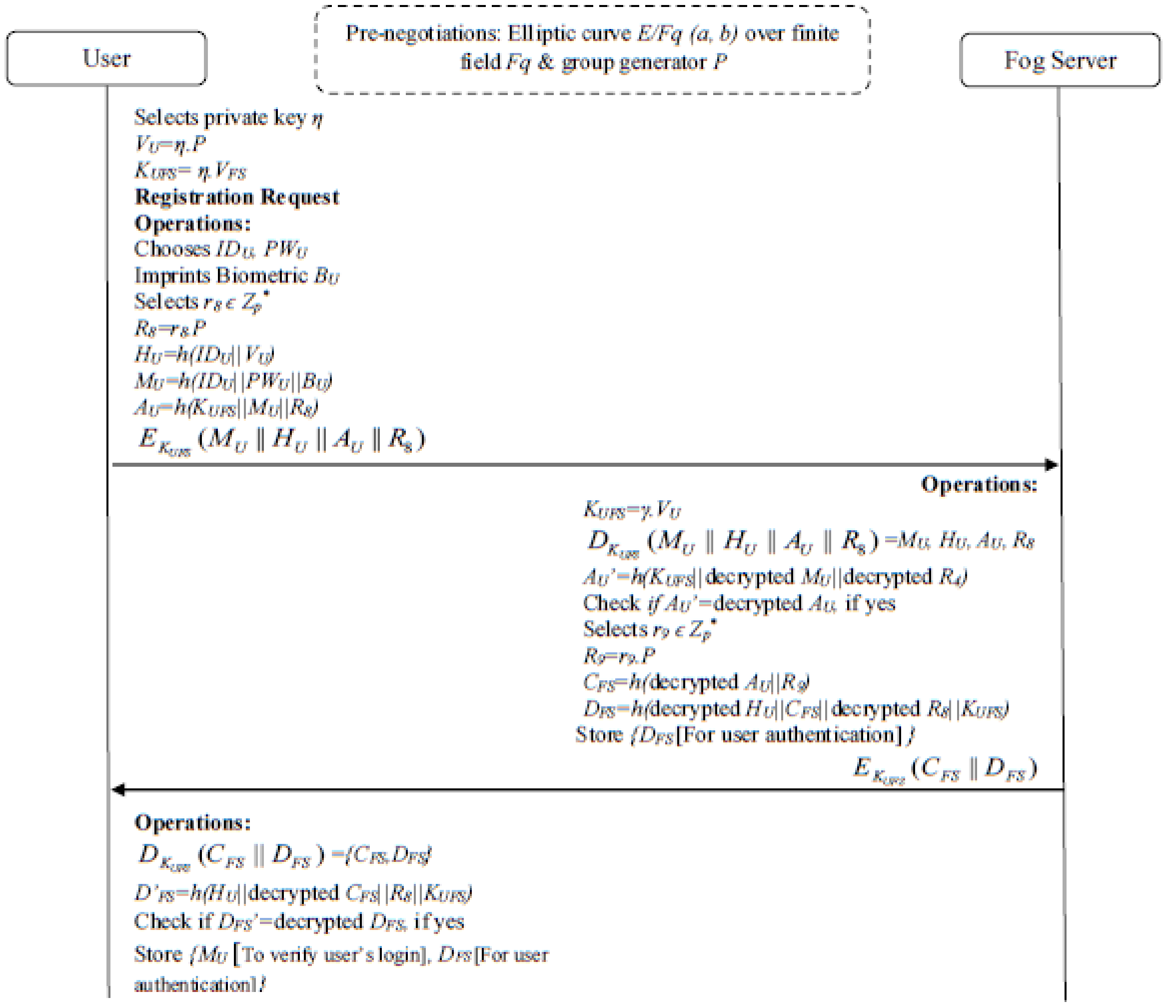

Fig. 4.6 Summary of user registration phase.

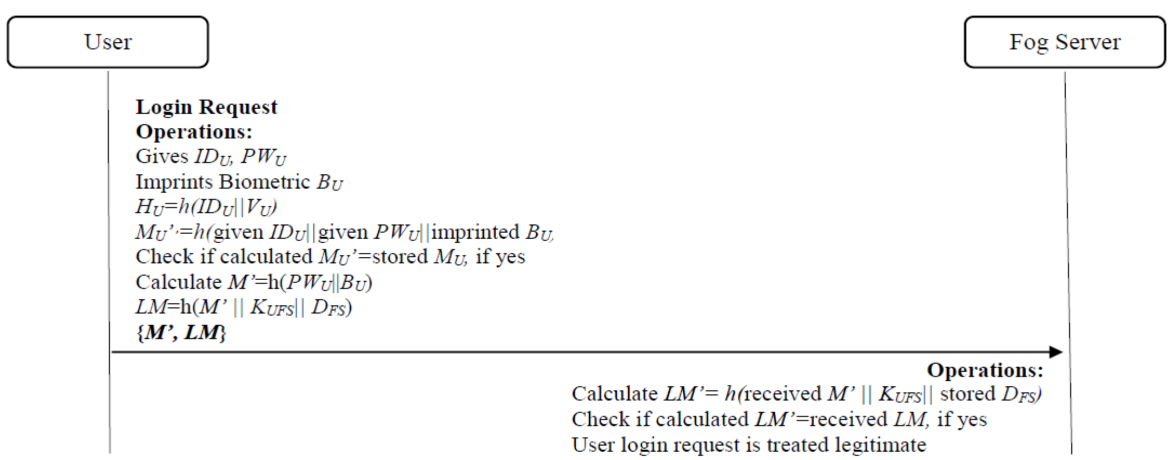

Fig. 4.7 Summary of user login phase.

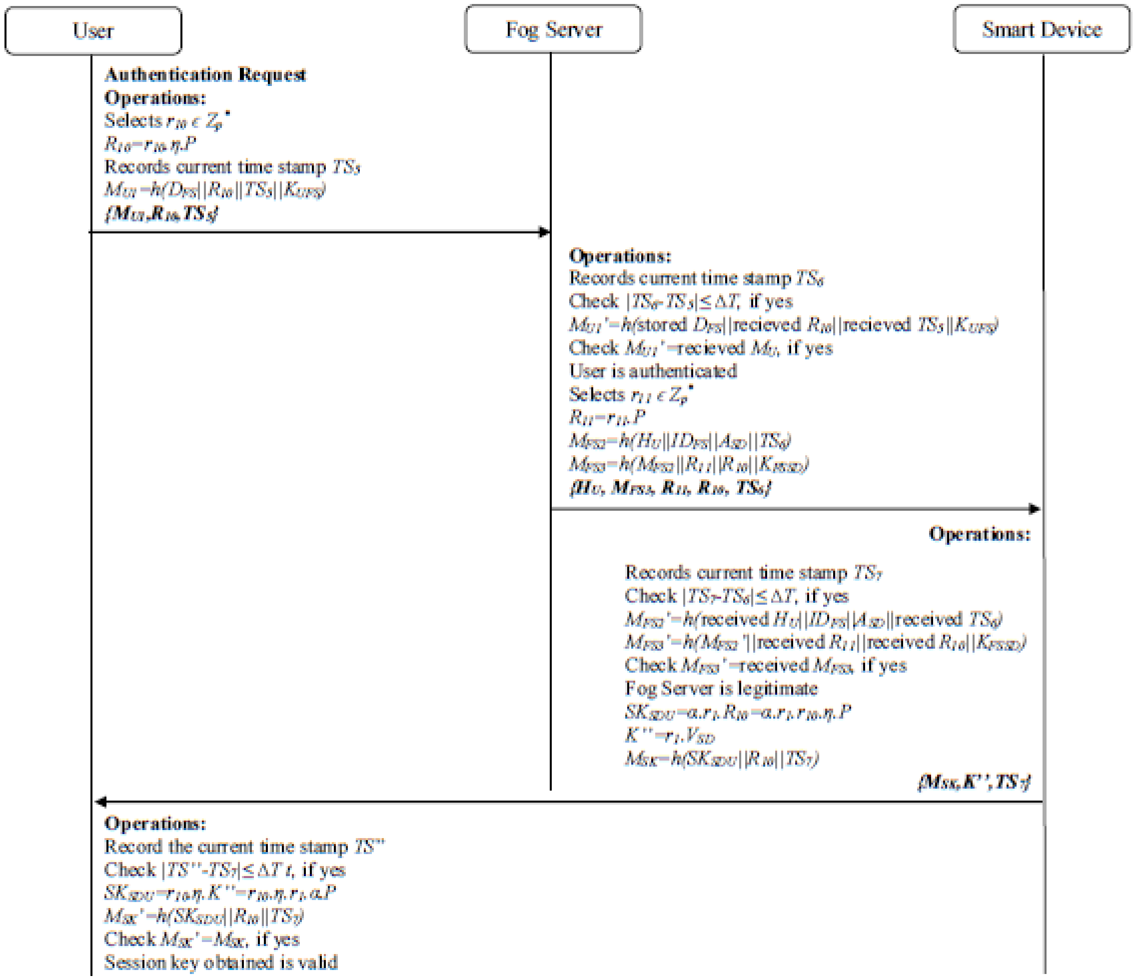

Fig. 4.8 Summary of user authentication phase

Chapter 5

Security Analysis

Security of any key management and user authentication protocol for any network is the most important although not easy to achieve. This chapter discusses some of those standards and analyzes how our scheme performs in those scenario. This chapter contains the informal security analysis of the proposed scheme and performs it's simulation on AVISPA (Automated Validation of Internet Security Protocols and Applications).

5.1 Informal Security Analysis

To show that the proposed scheme provides resilient against relevant security attacks, some security subjects are conferred below.

Proposition 1: The proposed scheme has three-factor security.

Proof: The proposed scheme requires three factors identity (ID_U), password (PW_U) and biometrics (B_U) to authenticate to fog server to access any kind of data of a smart device. Even if somehow adversary guesses the password, identity or both, he still won't be able to login because in login phase, check at user's end is laid on M_U' which is calculated as $M_{U'}=h($given $ID_U||$given $PW_U||$imprinted $B_U)$, only if calculated M_U' matches the stored $M_{U'}$ then only login request is forwarded to FS.

Proposition 2: Proposed scheme is resilient against stolen mobile device attack.

Proof: Suppose the mobile device is somehow stolen and adversary, through power analysis, guesses $H_U=h(ID_U||V_U)$, $M_U=h(ID_U||PW_U||B_U)$ and $A_U=h(K_{UFS}||M_U||R_8)$. Thus he would pass the login phase but still won't be able to authenticate with FS. As in authentication phase it needs to generate message $M_{U1}=h(D_{FS}||R_{10}||TS_5||K_{UFS})$ which won't be feasible to calculate as it needs ECDH based symmetric key K_{UFS}, also further it would need to generate R_{10} which too can't be calculated as it requires private key η and random nonce r_{10} which too would change after each session. Thus, adversary won't be able to generate a legitimate authentication request.

Proposition 3: Proposed scheme also provides resilient against impersonation attack.

Proof: Suppose an adversary intercepts a valid authentication request message $\{M_{U1}, R_{10}, TS_5\}$ where M_{U1} is calculated as $M_{U1}=h(D_{FS}||R_{10}||TS_5||K_{UFS})$. Now if somehow he guesses D_{FS} and tries to modify M_{U1} such that it is treated as a genuine request he still needs to guess K_{UFS} which will be computationally infeasible due to hardness of ECPM.

Proposition 4: Proposed scheme provides resilient against Fog Server insider attack

Proof: Suppose somehow adversary gains access to a fog server and comes to know all the secrets stored in fog server like r_{11} , M_{FS2} , M_{FS3} . It still won't be able to guess the session key. Session key could be calculated in 2 ways:

At smart device, session key is calculated as $SK_{SDU}=\alpha.r_1.R_{10}$ where both α and r_1 are still unknown, thus it would be infeasible to calculate session key from this method.

At user/mobile device end session key is calculated as $SK_{SDU}=r_{10}.\eta.K''$, here also r_{10} and η are still unknown, thus from this way also it won't be possible to calculate the session key.

Proposition 5: Proposed scheme is resilient against replay attack.

Proof: An adversary might intercept the authentication message and keep on sending it again and again, proposed scheme provides resilient against the same as it contains time stamps and whenever same message would be repeated it would be discarded automatically by FS as it won't be able to satisfy the timeliness condition at FS.

Proposition 6: Proposed scheme ensures user anonymity.

Proof: User anonymity means that user's identity is kept anonymous during the whole phase of communication. Proposed scheme keeps the user's identity safe as at no phase, User login or authentication, user's identity is never sent directly on the public channel. Thus preventing any kind of tracking of user's activity.

Proposition 7: Proposed scheme provides perfect forward secrecy.

Proof: The scheme proposed provides perfect secrecy, i.e. even if any long term secret key like K_{UFS} (shared key between user and fog server) or K_{FSSD} (shared key between FS and SD) but that information still won't help adversity in calculating $SK_{SDU}=\alpha.r_1.R_{10}=\alpha.r_1.r_{10}.\eta.P$ as it still would require short term secret r_{10} and secret keys α and η to calculate the key.

5.2 AVISPA (Automated Validation of Internet Security Protocols and Applications)

Automated Validation of Internet Security Protocols and Applications (AVISPA) is a simulation tool, to check whether a security scheme is SAFE or UNSAFE against all the existing security attacks. AVISPA is well accepted as formal security verification tool in recent day's research [13]. The proposed scheme is also simulated in AVISPA for its formal security analysis.

High Level Protocol Specification Language (HLPSL) is used for writing this simulation code, which is converted through a translator hlpsl2if into a low-level code i.e. a code into an intermediate format (IF). The low-level language IF, can be understood by the four back-ends of AVISPA which are (i) On-the-fly Model-Checker (OFMC), (ii) Constraint Logic based Attack Searcher (CL-AtSe), (iii) SAT-based Model-Checker (SATMC), (iv) Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). As AVISPA is a role oriented language each participant is implemented as a separate role. On the basis of the four back ends, the Output Format (OF) is produced which describes the results, whether the scheme is safe or unsafe.

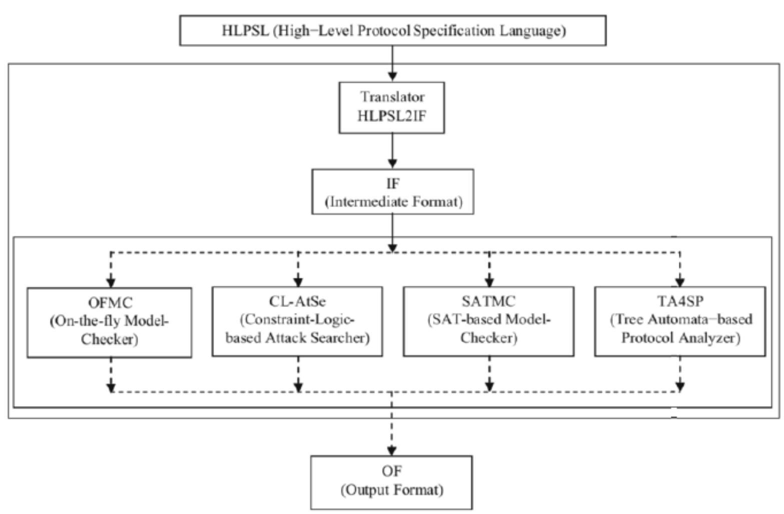

Fig. 5.1 Architecture of AVISPA

role Fog Server

```
role FS(SDj,FSj,CSj,Ui,TA:agent,
  Ksdt:symmetric_key,
  Kfst: symmetric_key,
  Kcst: symmetric_key,
  Kfssd : symmetric_key,
  Kfscs: symmetric_key,
  Kufs: symmetric_key,
  H,MUL,F:hash_func,
   Snd,Rcv(dy))
played_by FSj
def=
local State:nat,
a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,Kbd,K
fscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r11,Mfs2,SKs
du,TSddd:text,
Hsd,Asd,R1,Hfs,Afs,R2,Asd,Hcs,Acs,R3,Afs,r4d,TS1,r5,
BBfs,TS2,r6',TS3,r7,BBcs,TS4,Mu,Hu,Au,R8,Cfs,Dfs,M
d,LM,Mu1,R10,TS5,Mfs3,R11,TS6,Msk,Kdd,TS7:messag
e,
Inc:hash_func
const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs,CS_
FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_LM,U_FS_
Mu1,FS_SD_Mfs3,SD_U_Msk,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
init State:=0
transition
const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs,CS_
FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_LM,U_FS_
Mu1,FS_SD_Mfs3,SD_U_Msk,
subs1,subs2:protocol_id
init State:=0
transition
1.State=0/\Rcv(start)=|>
State':=1/\g'=new()
/\Vfs':=MUL(g'.P)
/\Kfst':=MUL(g'.Vt)
/\r2':=new()
/\R2':=MUL(r2'.Vfs')
/\Hfs':=H(IDfs.Vfs')
/\Afs':=H(Hfs'.R2'.Kfst')
/\Snd({Hfs'.Afs'.R2'}_Kfst')
/\witness(FSj,TAj,FS_TA_Afs,Afs)
/\secret({g'},subs3,FSj)
/\secret({Kfst'},subs4,{FSj,TAj})
/\rcv(Asd')
/\r6':=new()
/\TS3':=new()
/\R6':=H(r6'.Afs'.TS3')
2.State=1/\Rcv(r4'.TS1')=|>
State':=2/\TS2':=new()
/\r5':=new()
/\Ffssd=F(Afs'.Asd')
/\Ka':=MUL(g.Vsd')
/\hkar5':=H(Ka'.r5')
/\Kfssd':=H(Ffssd'.r4'.hkar5'.TS2')
/\BBfs':=H(Kfssd'.TS2')
Snd(Afs',r5',BBfs',TS2')
3.State=2/\Rcv(Acs',r7',BBcs',TS4')=|>
State':=3/\Kb':=g.Vfs'
/\Ffscs':=F(Afs'.Acs')
/\hKbr7':=H(Kb'.r7')
/\Kfscs':=H(Ffscs'.r6'.hKbr7'.TS4')
4.State=3/\Rcv({Mu',Hu',Au',R8'}_Kufs')=|>
State':=4/\Kufs':=MUL(g'.Vu)
/\r9':=new()
/\R9':= MUL(r9'.P)
/\Cfs':=H(Au'.R8')
/\Dfs':=H(Hu'.Cfs'.R8'.Kufs')
/\Snd({Cfs'.Dfs'}_Kufs')
3.State=2/\Rcv(Mu1',R10',TS5')=|>
State':=3/\TS6':=new()
/\r11':=new()
/\R11':=MUL(r11'.P)
/\Mfs2':=H(Hu.IDfs.Asd'.TS6')
/\Mfs3':=H(Mfs2'.R11'.R10'.Kfssd')
Snd({Hu'.Mfs3'.R11'.R10'.TS6'})
```

role User

```
role U(SDj,FSj,CSj,Ui,TA:agent,
 Ksdt:symmetric_key,
 Kfst: symmetric_key,
 Kcst: symmetric_key,
 Kfssd: symmetric_key,
 Kfscs: symmetric_key,
 Kufs: symmetric_key,
 H,MUL,F:hash_func,

   Snd,Rcv(dy))

played_by Ui

def=
local State:nat,

a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,Kbd,
Kfscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r11,Mfs2,S
Ksdu,TSddd:text,

Hsd,Asd,R1,Hfs,Afs,R2,Asd,Hcs,Acs,R3,Afs,r4d,TS1,r5,
BBfs,TS2,r6',TS3,r7,BBcs,TS4,Mu,Hu,Au,R8,Cfs,Dfs,
Md,LM,Mu1,R10,TS5,Mfs3,R11,TS6,Msk,Kdd,TS7:mes
sage,

Inc:hash_func

const

SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs,CS
_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_LM,U_F
S_Mu1,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2,subs3,subs4,subs5,subs6:protocol_id

init State:=0

transition
const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs,CS
_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_LM,U_F
S_Mu1,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2:protocol_id
init State:=0

transition

1.State=0/\Rcv(start)=|>
State':=1/\n'=new()
/\Vu':=MUL(n'.P)
/\Kufs':=MUL(n'.Vfs)
/\IDu':=new()
/\PW':=new()
/\Bu':=new()
/\r8':=new()
/\R8':=MUL(r8'.P)
/\Hu':=H(IDu'.Vu')
/\Mu':=H(IDu'.PWu'.Bu')
/\Au':=H(Kufs'.Mu'.R8')

Snd({Mu',Hu',Au',R8'}_Kufs')
/\M':=H(PWu'.Bu')
/\LM':=H(M'.Kufs'.Dfs')

Snd(M',LM')
/\r10':=new()
/\R10':=MUL(r10'.n.P)
/\TS5':=new()
/\Mu1':=H(Dfs'.R10.TS5'.Kufs')

Snd(Mu1',R10',TS5')

2.State=1/\Rcv(Msk,K2',TS7')=|>
State':=2/\TS3':=new()
/\SKsdu'=MUL(r10'.n.K2')
```

Fig. 5.2 Role Fog Server and Role User role Trusted Agency

```
role TA(SDj,FSj,CSj,Ui,TA:agent,
  Ksdt:symmetric_key,
  Kfst: symmetric_key,
  Kcst: symmetric_key,
  Kfssd: symmetric_key,
  Kfscs: symmetric_key,
  Kufs: symmetric_key,
  H,MUL,F:hash_func,

    Snd,Rcv(dy))

played_by TA

def=
local State:nat,

a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,
Kbd,Kfscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r
11,Mfs2,SKsdu,TSddd:text,

Hsd,Asd,R1,Hfs,Afs,R2,Asd,Hcs,Acs,R3,Afs,r4d,T
S1,r5,BBfs,TS2,r6',TS3,r7,BBcs,TS4,Mu,Hu,Au,R8
,Cfs,Dfs,Md,LM,Mul,R10,TS5,Mfs3,R11,TS6,Msk
,Kdd,TS7:message,

Inc:hash_func

const

SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBf
s,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_
LM,U_FS_Mul,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2,subs3,subs4,subs5,subs6:protocol_id

init State:=0
transition

const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBf
s,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_
LM,U_FS_Mul,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2:protocol_id

init State:=0

transition

1.State=0/\Rcv(start)=|>
State':= 1/\b'=new()
/\Vt':=MUL(b'.P)

2.State=1/\Rcv({Hsd',Asd',R1'}_Ksdt')=|>
State':=2/\Ksdt':=MUL(b'.Vsd')

3.State=2/\Rcv({Hfs',Afs',R2'}_Kfst')=|>
State':=3/\Kfst':=MUL(b'.Vfs')
Snd(Asd)

4.State=3/\Rcv({Hcs',Acs',R3'}_Kcst')=|>
State':=4/\Kcst':=MUL(b'.Vcs')
Snd(Afs)
```

role Smart Device

```
role SD(SDj,FSj,CSj,Ui,TA:agent,
  Ksdt:symmetric_key,
  Kfst: symmetric_key,
  Kcst: symmetric_key,
  Kfssd: symmetric_key,
  Kfscs: symmetric_key,
  Kufs: symmetric_key,
  H,MUL,F:hash_func,
   Snd,Rcv(dy))
played_by SDj

def=
local State:nat,
a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,K
bd,Kfscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r11,
Mfs2,SKsdu,TSddd:text,
Hsd,Asd,R1,Hfs,Afs,R2,Asd,Hcs,Acs,R3,Afs,r4d,TS
1,r5,BBfs,TS2,r6',TS3,r7,BBcs,TS4,Mu,Hu,Au,R8,C
fs,Dfs,Md,LM,Mul,R10,TS5,Mfs3,R11,TS6,Msk,Kd
d,TS7:message,
Inc:hash_func

const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs
,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_L
M,U_FS_Mul,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
init State:=0
transition

const
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs
,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_L
M,U_FS_Mul,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2:protocol_id
init State:=0
transition
1.State=0/\Rcv(start)=|>
State':=1/\a'=new()
/\Vsd':=MUL(a'.P)
/\Ksdt':=MUL(a'.Vt)
/\r1':=new()
/\R1':=MUL(r1'.Vsd')
/\Hsd':=H(IDsd.Vsd')
/\Asd':=H(Hsd'.R1'.Ksdt')
/\Snd({Hsd'.Asd'.R1'}_Ksdt')
/\witness(SDj,TAj,SD_TA_Asd,Asd)
/\secret({a'},subs1,SDj)
/\secret({Ksdt'},subs2,{SDj,TAj})
/\r4':=new()
/\TS1':=new()
/\R4':=H(r4'.Asd'.TS1')
2.State=1/\Rcv(Afs',r5',BBfs',TS2')=|>
State':=2/\TS':=new()
/\Fsdfs':=F(Asd'.Afs')
/\Ka':=a.Vfs'
/\ha5':=H(Ka'.r5')
/\Kfssd':=H(Fsdfs'.r4'.ha5'.TS2)
/\request(FS,SD, FS_SD_BBfs,BBfs)

3.State=2/\Rcv(Hu',Mfs3',R11',R10',TS6')=|>
State':=3/\TS7':=new()
/\SKsdu':=MUL(a'.r1'.R10')
/\K2':=MUL(r1'.Vsd')
/\Msk':=H(SKsdu'.R10'.TS7')
Snd(Msk',K2',TS7')
end role
```

Fig. 5.3 Role Trusted Agency and Role Smart Device role Session/Environment

```
role session(SDj,FSj,CSj,Ui,TA:agent,
  Ksdt:symmetric_key,
  Kfst: symmetric_key,
  Kcst: symmetric_key,
  Kfssd: symmetric_key,
  Kfscs: symmetric_key,
  Kufs: symmetric_key,
  H,MUL,F:hash_func)
def=
local SI,SJ,RI,RJ,TI,TJ:channel(dy)
composition
U(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,Kufs,
H,MUL,F,SI,RI)
/\SD(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,Kufs
, H,MUL,F,TI,TJ)
/\FS(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,Kufs,
H,MUL,F,SJ,RJ)
/\CS(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,Kufs
, H,MUL,F,SJ,RJ)
end role
role envirionment()
def=
const SDj,FSj,CSj,Ui,TA:agent,
Ksdt:symmetric_key,
Kfst: symmetric_key,
Kcst: symmetric_key,
Kfssd: symmetric_key,
Kfscs: symmetric_key,
Kufs: symmetric_key,
H,MUL,F:hash_func)
H,MUL,F:hash_func,
a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,Kbd,K
fscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r11,Mfs2,SKs
du,TSddd:text,
SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBfs,CS_
FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_LM,U_FS_
Mu1,FS_SD_Mfs3,SD_U_Msk,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
intruder_knowledge={r4d,TS1,Afs,r5,BBfs,TS2,r62,TS3,
Acs,r7,BBcs,TS4,Md,LM,Mu1,R10,TS5,Hu,Mfs3,R11,R1
0,TS6,Msk,Kd,TS7}
composition
session(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,K
ufs, H,MUL,F)
/\session(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst, Kcst, Kfssd,
Kfscs, Kufs, H,MUL,F)
/\session(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,
Kufs, H,MUL,F)
/\session(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,
Kufs, H,MUL,F)
/\session(SDj,FSj,CSj,Ui,TA,Ksdt,Kfst,Kcst,Kfssd,Kfscs,
Kufs, H,MUL,F)
end role
goal
secrecy_of subs1
secrecy_of subs2
secrecy_of subs3
secrecy_of subs4
secrecy_of subs5
secrecy_of subs6
authentication_on SD_TA_Asd
 authentication_on FS_TA_Afs
authentication_on CS_TA_Acs
authentication_on FS_SD_BBfs
authentication_on  CS_FS_BBcs
authentication_on  U_FS_Au
authentication_on FS_U_Dfs
authentication_on U_Mu
authentication_on U_FS_LM
authentication_on U_FS_Mu1
authentication_on FS_SD_Mfs3
authentication_on SD_U_Msk
end goal
environment()
```

role Cloud Server

```
role CS(SDj,FSj,CSj,Ui,TA:agent,

  SDj:symmetric_key,
  FSj: symmetric_key,
  CSj: symmetric_key,
  Ui: symmetric_key,
  TA: symmetric_key,

  H,MUL,F:hash_func,

   Snd,Rcv(dy))

played_by CSj

def=

local State:nat,

a,Ksd,r1,b,g,Kfst,rd,v,Kcst,r3,r4,Kad,Kfssd,TSd,r6,
Kbd,Kfscs,TSdd,n,Kufs,IDu,PWu,Bu,r8,r9,R9,r10,r
11,Mfs2,SKsdu,TSddd:text,

Hsd,Asd,R1,Hfs,Afs,R2,Asd,Hcs,Acs,R3,Afs,r4d,TS
1,r5,BBfs,TS2,r6',TS3,r7,BBcs,TS4,Mu,Hu,Au,R8,
Cfs,Dfs,Md,LM,Mu1,R10,TS5,Mfs3,R11,TS6,Msk,
Kdd,TS7:message,

Inc:hash_func

const

SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBf
s,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_
LM,U_FS_Mu1,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2:protocol_id

init State:=0

transition
const

SD_TA_Asd,FS_TA_Afs,CS_TA_Acs,FS_SD_BBf
s,CS_FS_BBcs,U_FS_Au,FS_U_Dfs,U_Mu,U_FS_
LM,U_FS_Mu1,FS_SD_Mfs3,SD_U_Msk,

subs1,subs2:protocol_id

init State:=0
transition

1.State=0/\Rcv(start)=|>
State':=1/\b'=new()
/\Vt':=MUL(b'.P)

2.State=1/\Rcv({Hsd',Asd',R1'}_Ksdt')=|>
State':=2/\Ksdt':=MUL(b'.Vsd')

3.State=2/\Rcv({Hfs',Afs',R2'}_Kfst')=|>
State':=3/\Kfst':=MUL(b'.Vfs')
Snd(Asd)

4.State=3/\Rcv({Hcs',Acs',R3'}_Kcst')=|>
State':=4/\Kcst':=MUL(b'.Vcs')

Snd(Afs)
```

Fig. 5.4 Role session/environment and Role Cloud Server

Simulation results

In this chapter simulation result of HLPSL code has been represented in Figs. 5.5 and 5.6 which depict the results for the back-ends OFMC and CL-AtSe of AVISPA. The simulation results authorize that the proposed scheme is SAFE and able to withstand all active and passive attacks like man-in-the-middle attack, replay attacks etc.

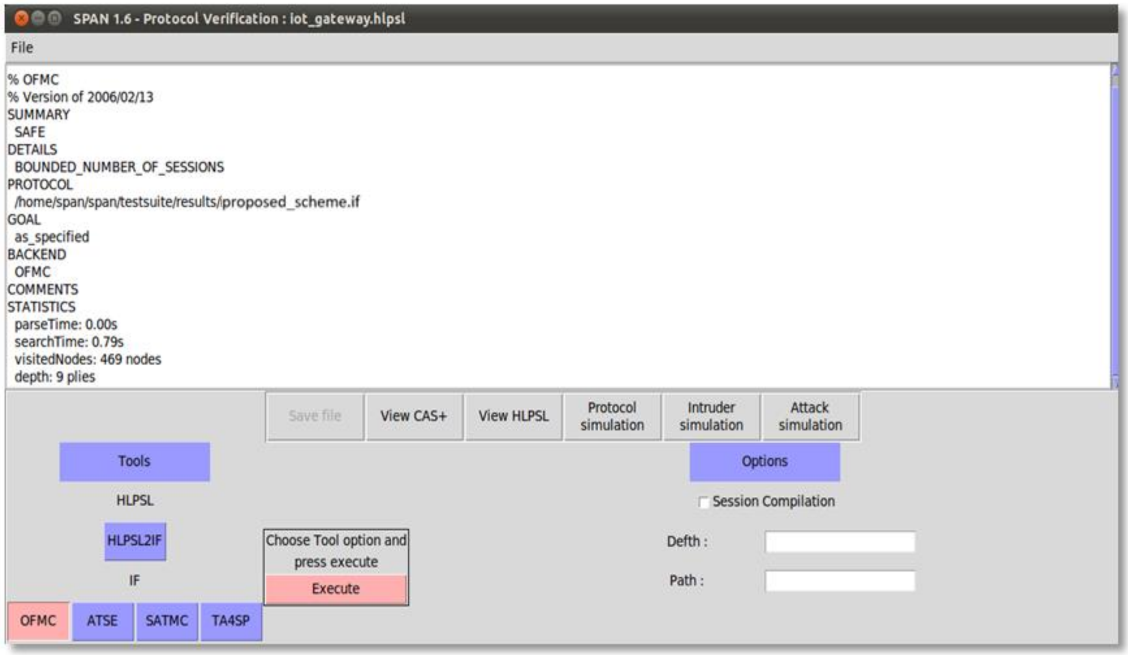

Fig. 5.5 Result analysis of the proposed scheme using OMFC back-end of AVISPA

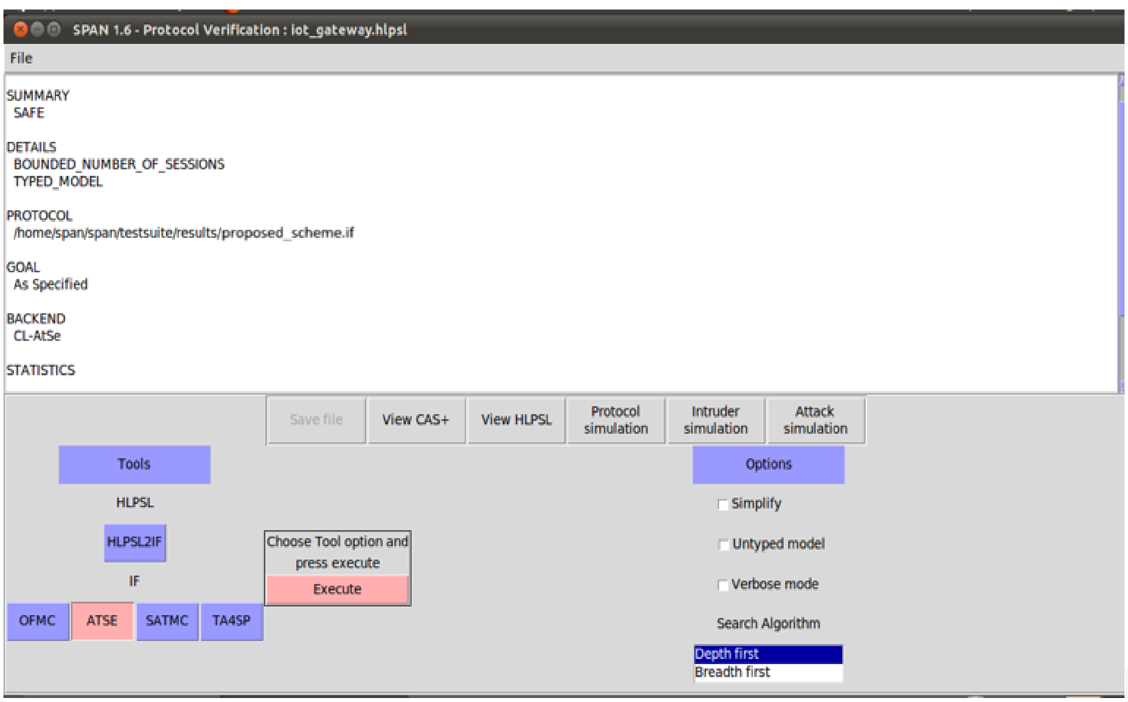

Fig. 5.6 Result analysis of the proposed scheme using CL-AtSe back-end of AVISPA

Chapter 6

Performance Analysis

In this chapter, the proposed scheme is analyzed with respect to its performance in comparison with other relevant schemes on the basis of different performance parameters.

6.1 Computation overhead comparison

Table 6.1 shows the comparison of the computational cost of the scheme in execution time (in milliseconds) with other relevant schemes. Let T_{ecm}, T_{eca}, T_{sed}, T_h, T_{fe}, $T_{enc-ibe}$, $T_{dec-ibe}$, $T_{sig-ibe}$, $T_{ver-ibe}$, T_{exp}, T_{pke} and T_{pkd} denote the computational time required for an elliptic curve point multiplication, an elliptic curve point addition, a symmetric encryption/decryption, a cryptographic one-way hash function h(.), a fuzzy extraction operation (Gen(.)/Rep(.)), identity-based encryption (IBE), identity based decryption, identity-based signature generation, identity based signature verification, modular exponentiation, public key encryption and public key decryption, respectively. Since, login and authentication phases are only frequently executed, so only those 2 phases are used for calculation.

Results of cryptographic operations as reported in [17-19] are used here. The experiment values reported in [17-19] for T_{ecm}, T_{eca}, T_{sed} and T_h are 0.063075 s, 0.010875 s, 0.0087 s and 0.0005 s, respectively. As in [19], it is assumed that the execution time needed for a fuzzy extractor is approximately equal to that for an elliptic curve point multiplication time at most, that is, $T_{fe} \approx T_{ecm}$ = 0.063075 s. Moreover, as reported in [17], we have $T_{exp} \approx 60T_{sed}$ = 0.522 s, and $T_{pke}/T_{pkd} \approx 100T_{sed}$ = 0.87 s.

Table 6.1 Comparison of computational cost of different schemes in related field

SCHEME	*U/MD*	*FS*	*CS*	*SD*	TOTAL	EXECUTION TIME
Ours	$6T_H+2T_{ECM}$	$4T_H+1T_{ECM}$	-	$3T_H+2T_{ECM}$	$13T_H+5T_{ECM}$	**316.025ms**
Ref [1]	$16T_H+2T_{ECM}+1T_{FE}$	$10T_H+3T_{ECM}$	-	$9T_H$	$35T_H+5T_{ECM}+1T_{FE}$	395.95ms
Ref [3]	-	$2T_{exp}+2T_{pke}+T_{PKD}$	$2T_{exp}+2T_{pke}+2T_{PKD}$	-	$4T_{exp}+3T_{pke}+3T_{PKD}$	7308 ms
Ref [15]	$6T_H+2T_{ECM}$	-	$7T_H+3T_{ECM}+4T_{SED}$	-	$13T_H+5T_{ECM}+4T_{SED}$	356.68 ms
Ref [14]	$3T_H+4T_{ECM}$	-	$4T_H+6T_{ECM}+4T_{ECA}$	-	$6T_H+2T_{ECM}$	677.75 ms

Table 6.2 Comparison of communication cost.

SCHEME	TOTAL NO. OF MESSAGES	TOTAL NO. OF BITS
Ours	4	1600
Ref [1]	3	2816
Ref [3]	3	7168
Ref [15]	4	4160
Ref [4]	2	2688

6.2 Communication overhead comparison

In Table 6.2, communication overhead is compared for the proposed and other relevant schemes. For computing the communication cost, length of password, identity (user, fog server and smart device) and random nonce as *128* bits each whereas timestamp and identity of timestamp are of 32 bits each. The symmetric-key encryption or decryption message and hash function are 128 and 160 bits respectively.

Not only our proposed algorithm is resilient to all relevant security threats but also has much less communication cost as compared to its counterparts. Thus, this proposed scheme maintains an excellent tradeoff between performance and security.

Chapter 7

Conclusion

Wajid et. al. developed a secure key management and remote user authentication scheme for fog computing. Furthermore, Wajid et. al. claimed that the protocol is provably protected against all possible cryptographic security attacks. After analyzing the protocol in detail, it is found that Wajid et al. scheme is vulnerable to fog server insider attack and Denial of Service (DoS) attack. To overcome the aforementioned security loopholes, an enhanced and efficient protocol for secure key management and remote user authentication scheme for fog computing has been proposed. The proposed scheme is lightweight, since it uses Elliptic Curve Cryptography, hash functions and symmetric bivariate polynomial function. Further, it maintains a good tradeoff between security and performance which is hard to achieve. The state-of-art informal security analysis of the proposed scheme and the formal security analysis using AVISPA shows that the scheme is resilient to all relevant malicious attacks. The performance analysis of the proposed scheme in comparison with other relevant schemes shows that the scheme outperforms the other pre-existing schemes. Further, the simplicity of the scheme makes it easily implementable in practical scenarios.

References

1. M. Wazid, A.K. Das, N. Kumar, A. Vasilakos. Design of secure key management and user authentication scheme for fog computing services, Future Generation Computing Systems 2018, 475-492
2. B. Martini, K.K.R. Choo, An integrated conceptual digital forensic framework for cloud computing, Digit. Investigation 9 (2) (2012) 71–80.
3. P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, X. Yao, Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things, IEEE Internet Things J. (2017) http://dx.doi.org/10.1109/JIOT.2017.2659783.
4. W. Abdul, Z. Ali, S. Ghouzali, B. Alfawaz, G. Muhammad, M.S. Hossain, Biometric security through visual encryption for fog edge computing, IEEE Access 5(1) (2017) 5531–5538.
5. D. Koo, J. Hur, Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing, Future Gener. Comput. Syst. 78 (2018) 739–752.
6. H. Wang, Z. Wang, J. D.-F.errer, Anonymous and secure aggregation scheme in fog-based public cloud computing, Future Gener. Comput. Syst. 78 (2018) 712–719.
7. P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on fog computing: architecture, key technologies, applications and open issues, J. Netw. Comput. Appl. 98 (2017)27–42.
8. S. Ray, G.P. Biswas, M. Dasgupta, Secure multi-purpose mobile-banking using elliptic curve cryptography, Wireless Personal Communications 90, no. 3 (2016) 1331-1354.
9. H.S.K. Islam, R. Amin, G.P. Biswas, M.S. Farash, X. Li, S. Kumari, An improved three party authenticated key exchange protocol using hash function and elliptic curve cryptography for mobile-commerce environments, Journal of King Saud University-Computer and Information Sciences 29, no. 3 (2017) 311-324.
10. S. Ray, G.P. Biswas, Establishment of ECC-based initial secrecy usable for IKE implementation, In Proc. of World Congress on Expert Systems (WCE), 2012.
11. W. Stallings, Cryptography and network security: principles and practices, Pearson Education India, 2006
12. S. Ray, G.P. Biswas, An ECC based public key infrastructure usable for mobile applications, Proceedings of the second international conference on computational science, engineering and information technology, ACM (2012) 562-568.

13. K. Mahmood, S.A. Chaudhry, H. Naqvi, S. Kumari, X. Li, A.K. Sangaiah, An elliptic curve cryptography based lightweight authentication scheme for smart grid communication, Future Generation Computer Systems 81 (2018) 557-565.
14. H. Sun, Q. Wen, H. Zhang, Z. Jin, A novel remote user authentication and key agreement scheme for mobile client–server environment, Appl. Math. 7 (4) (2013) 1365–1374.
15. H. Li, F. Li, C. Song, Y. Yan, Towards smart card based mutual authentication schemes in cloud computing, KSII Trans. Internet Inform. Syst. 9 (7) (2015) 2719–2735.
16. H. Li, Y. Dai, L. Tian, H. Yang, Identity-based authentication for cloud computing, in: Cloud Computing, in: Lecture Notes in Computer Science, vol. 5931, Springer Berlin Heidelberg, 2009, pp. 157–166
17. C.T. Li, M.S. Hwang, Y.P. Chu, A secure and efficient communication scheme with authenticated key establishment and privacy preserving for vehicular adhoc networks, Comput. Commun. 31 (12) (2008) 2803–2814.
18. W. Li, Q. Wen, Q. Su, Z. Jin, An efficient and secure mobile payment protocol for restricted connectivity scenarios in vehicular ad hoc network, Comput. Commun. 35 (2) (2012) 188–195.
19. D. He, N. Kumar, J.H. Lee, R.S. Sherratt, Enhanced three-factor security protocol for consumer USB mass storage devices, IEEE Trans. Consum. Electron. 60 (1) (2014) 30–37.