Security Analysis of a Key Exchange Protocol under Dolev-Yao Threat Model Using Tamarin Prover

Singam Bhargav Ram
IIIT Sri City, Chittoor
Sri City - 517 646, AP, India
bhargavram.s18@iiits.in
ORC ID: 0000-0001-7984-7623

Vanga Odelu
IIIT Sri City, Chittoor
Sri City - 517 646, AP, India
odelu.vanga@iiits.in
ORC ID: 0000-0001-6903-0361

***Abstract*—In recent days, security and privacy is becoming a challenge due to the rapid development of technology. In 2021, Khan et al. proposed an authentication and key agreement framework for smart grid network and claimed that the proposed protocol provides security against all well-known attacks. However, in this paper, we present the analysis and shows that the protocol proposed by Khan et al has failed to protect the secrecy of the shared session key between the user and service provider. An adversary can derive the session key (online) by intercepting the communicated messages under the Dolev-Yao threat model. We simulated Khan et al.'s protocol for formal security verification using Tamarin Prover and found a trace for deriving the temporary key. It is used to encrypt the login request that includes the user's secret credentials. Hence, it also fails to preserve the privacy of the user's credentials, and therefore any adversary can impersonate the user. As a result, the protocol proposed by Khan et al. is not suitable for practical applications.**

***Index Terms*—Security, Privacy, User Anonymity, Key Secrecy, Tamarin Prover, Smart Grid**

I. Introduction

The Smart Grid(SG) is slated to replace conventional grids soon. The primary objective of a smart grid is to ensure power supply to the end users in a dependable and steady manner. The bidirectional communication structure provided by a smart grid will ensure a more reliable, secure and efficient way for the transmission and distribution of power resources [1]. In smart grids, the power utilization of consumers is regularly shared between end users and one or more power suppliers. This requires a secure method for enabling such communications, and in order to achieve this, various key agreement protocols have been introduced. Security is an important aspect of smart grids as the compromise of the secrecy and privacy of transmitted messages can lead to the failure of the smart grid infrastructure[2]. The security requirements for key distribution schemes for SGs have been put forward in [3]. The user facing devices in the smart grid are called smart meters(SM) and are typically resource constrained(i.e. have low computational and memory capacity), therefore the authentication mechanism used should be designed to not put any burden on these resources.

Several key agreement protocols were proposed to handle the security requirements of a smart grid communication network. Tsai-lo [4] (2016) presented an authentication scheme for a smart grid environment that used identity based encryption techniques to ensure mutual authentication with privacy without involving a trusted authority. However, Odelu et al.[5] showed that Tsai-Lo's scheme does not provided secrecy of the session key and smart meter credentials in the event of a temporary session secret leakage. Further a key agreement scheme has been proposed for the smart grid environment. Chen et al.[6] proposed an authentication protocol based on elliptic curve cryptography and bi-linear pairings. However, Wu et al.[7] showed that the protocol fails to hold against session specific temporary information attack and user impersonation attack. Khan et al.[8] proposed a key agreement framework using password, namely *PALK*, that uses a secure hash functions and ECC. However, Chaudary[9] showed that the protocol's login and authentication phase is invalid because of superficial ECC operations, and proposed a refined scheme.

In Khan et al.[10]'s work (2021), a new authentication and key agreement framework named as LAKAF has been proposed for smart grid networks. LAKAF is proposed to authenticate and establish a shared session key between an end user U and a service provider S. They had provided a formal analysis of their protocol using AVISPA and showed that the protocol proposed by them is secure against the attacks that are well known in the literature. Khan et al.[10] also gave a comparative analysis with related schemes to show that LAKAF has lower computation and communication costs. However, we have analyzed the protocol and found that the scheme fails to provide user anonymity, temporary key secrecy, shared session key secrecy and is susceptible to user and service provider impersonation attacks. The contributions of the paper are listed below:

- We analyze Khan et al.'s[10] authentication protocol and have shown that an adversary can impersonate the user

with the intercepted communicated messages from that user.
- We also verified for formally using the Tamarin Prover, a formal automated security verification tool, by modeling the protocol under the Dolev-Yao threat model.

This paper is organized as follows: We discussed the required cryptographic primitives in Section II. In Section III, we have reviewed the protocol proposed by Khan *et* al.[10]. In Section IV, we discussed the security flaws in Khan et al.'s[10] protocol. Further, in Section V, we presented simulation results of the formal security analysis using the Tamarin prover. Finally, Section VI concludes the paper with the future work.

II. Mathematical Preliminaries

We describe in brief the basic definitions and notations required to analyze the protocol and security issues.

A. Cryptographic Hash Function

One-way cryptographically secure collision-resistant hash functions [11] are deterministic functions that considers a random binary string of any length as the input and will produce a fixed length binary string as the output.

B. Biometric Fuzzy Extractor

A biometric fuzzy extractor [12] comprises of two methods, the generator and the reproducer, the generator is a probabilistic method that extracts two parameters, namely, one secret and one public parameters from a given biometric data. The reproduction method takes the public parameter along with a noisy biometric input that is closer to the original biometric input within a given error tolerance threshold, to recover the secret parameter.

A fuzzy extractor is defined as a tuple $(\mathcal{M}, l, \mathcal{T})$ that comprises of the following methods:

- *Gen*. The method that takes a $B_i \in \mathcal{M}$ as an input and produces a secret parameter, say, $\sigma_i \in \{0,1\}^l$ and a public parameter, say, τ_i as outputs, that is, *Gen*(B_i) = (σ_i, τ_i).
- *Rep*. The deterministic method that takes a noisy biometric input, say, $B_i^* \in \mathcal{M}$, and the corresponding public parameter τ_i and $\mathcal{T}$ related to B_i^* and recovers the secret key σ_i, where *Rep*(B_i^*, τ_i) = σ_i provided that $d(B_i^*, B_i) \leq \mathcal{T}$.

C. Elliptic Curve Cryptography

A non-singular elliptic curve $E_p(m,n)$ is defined by the equation $y^2 = x^3+mx+n$ over the finite field $GF(p)$ where p is a very large prime number, $m, n \in Z_p$ and $4m^3+27n^2 \neq 0\,(mod p)$. The point $\mathcal{O}$ is known as the *point at infinity* or *zero point*. If $R = (x_R, y_R)$ and $S = (x_S, y_S)$ are two points on the elliptic curve $E_p(m,n)$ then the addition of R and S is given by $T = (x_T, y_T) = R + S$ and can be computed as follows [13]:

$$x_T = (\lambda - x_R - x_S),$$
$$y_T = (\lambda(x_T - x_R) - y_R),$$
$$\text{where } \lambda = \begin{cases} \frac{y_S - y_R}{x_S - x_R}, if R \neq S \\ \frac{x_R^2 + m}{2y_R}, if R = S \end{cases}$$

In ECC, scalar multiplication is defined by repeated additions, if $P \in E_p(m,n)$, then kP is given by the equation $kP = P + P + \ldots + P(k\,\text{times})$.

The following mathematical problems are said to be computationally infeasible [13].

- *Elliptic Curve Discrete Logarithmic Problem(ECDLP)*: Given $P, Q \in E_p(m,n)$, where $Q = aP$ and $a \in Z_p^*$, it is computationally difficult to derive k from Q.
- *Elliptic Curve Computational Diffie-Hellman Problem (ECCDHP)*: Given $P, aP, bP \in E_p(m,n)$, where $a, b \in Z_p^*$, it is computationally difficult to compute abP.

III. Review of Khan et al.'s Scheme

In this section, we briefly review the proposed Khan et al.'s[10] authentication and key agreement scheme for smart grid. Notations used are listed in Table I.

TABLE I
Notations Used

Notations	Description
h(.)	Cryptographic Hash Function
$E_K(\dot{})/D_K(\dot{})$	Symmetric Enc/Decryption with key K
$Gen()$	Probabilistic Fuzzy Generator
$Rep()$	Deterministic Fuzzy Reproducer
$\|\|$	Concatenation Operator
$\oplus$	Bit-wise XOR operator
Δt	Valid Time Interval

A. TA Setup Phase

- The TA chooses a non-singular elliptic curve $E_q(m,n) : y^2 = x^3 + mx + n$ over a finite field $GF(q)$, where q is a very large prime number. TA selects a base point P in E_q and a collision resistant hash function $h(.)$.
- The TA generates its private key x, which is kept a secret and generates the corresponding public key $P_{pub} = xP$.

B. User Registration Phase

A user U registers with the TA as follows:

- User U, inputs its identity ID_U, a password PW_U and generates a random string $r_U \in Z_p^*$. U computes $(\sigma_U, \tau_U) = Gen(BIO_U)$ where BIO_U is the biometric imprint of the user and $\gamma_U = h(PW_U \| \sigma_U) \oplus r_U$. The parameters $\{ID_U, \gamma_U\}$ are sent to the TA via a secure channel
- After receiving $\{ID_U, \gamma_U\}$, the TA computes $A = h(ID_U \| x \| C_U)$ where C_U is the registration counter for

U generated by the TA. The TA then stores $\{ID_U, C_U\}$ in its database. Next, TA computes $B = A \oplus \gamma_U$ and stores the values $\{B, C_U\}$ in the database. Finally, TA sends $\{B, C_U\}$ to U via a secure channel.

- After receiving $\{B, C_U\}$, U computes $B_1 = B \oplus \sigma_U$ and $B_2 = h(ID_U \| PW_U \| B_1)$ and then stores $\{\tau_U, B, B_1, B_2\}$.

C. Service Provider Registration Phase

A service provider S registers with the TA as follows:

- Service Provider S, sends its identity ID_S to the TA via a secure channel.
- After receiving ID_S, TA computes $V = h(ID_S \| x \| C_S)$ where C_S is the registration counter for S generated by the TA and stores $\{ID_S, C_S\}$ in the database. Then, the TA sends $\{V, C_S\}$ to S via a secure channel.
- After receiving $\{V, C_S\}$, S generates a secret key $r_S \in Z_p^*$ and generates the corresponding public key $PK_S = r_S P$ and stores $\{V, C_S\}$.

D. Login, Authentication and Key Agreement Phase

In this phase, User U and Service Provider S, establish a common session key as follows:

- At User U:
  - U logs in with ID_U, PW_U, BIO_U and calculates $\sigma_U = Rep(BIO_U, \tau_U)$.
  - U computes $B_1 = B \oplus \sigma_U$ and $B_2^* = h(ID_U \| PW_U \| B_1)$.
  - U checks the condition, $B_2^* = B_2$, if it holds, selects $a \in Z_p^*$ and at time t_1, calculates $S_1 = h(ID_U \| a \| C_U)$, $I_1 = ID_U \oplus (h(t_1) \oplus t_1)$ and $K_1 = h(ID_U \| h(t_1 \oplus PK_S) \| t_1)$.
  - U encrypts $E_1 = E_{K_1}(a, S_1, C_U)$ with K_1 and sends $M_1 = \{E_1, I_1, t_1\}$ to S via a public channel.
- At Service Provider S:
  - On receiving $M_1 = \{E_1, I_1, t_1\}$ at time t_2 and S checks if $t_2 - t_1 \leq \Delta t$.
  - S then computes, $I_2 = I_1 \oplus (h(t_1) \oplus t_1)$, $K_2 = h(I_2 \| h(t_1 \oplus PK_S) \| t_1)$, verifies if $ID_U = I_2$ and decrypts E_1 with K_2 to get $(a, S_1, C_U) = D_{K_2}(E1)$.
  - S then checks if $S_1 = h(ID_U \| a \| C_U)$ holds.
  - S generates a random string $b \in Z_p^*$ and then computes the shared session key $SK_S = h(ID_U \| ID_S \| C_U \| C_S \| abP \| t_3)$.
  - S then computes $S_2 = h(ID_U \| ID_S \| S_1 \| V \| SK_S \| t_1)$, $V_1 = V \oplus h(C_U \| ID_U \| K_2)$, $N = ID_S \oplus h(b \| C_S \| C_U)$ and $K_3 = h(ID_U \| S_1 \| a \| C_U \| t_3)$.
  - S encrypts $E_2 = E_{K_2}(N, V_1, S_2, C_S, b)$ with K_3 and sends $M_2 = \{E_2, t_3\}$ to U via a public channel.
- At user U:
  - On receiving $M_2 = \{E_2, t_3\}$ at time t_4, U checks if $t_4 - t_3 \leq \Delta t$.
  - U computes $K_4 = h(ID_U \| S_1 \| a \| C_U \| t_3)$ and decrypts E_2 with K_4 to get $(N, V_1, S_2, C_S, b) = D_{K_4}(E2)$.
  - U then verifies if $ID_S = N \oplus h(b \| C_S \| C_U)$ and then computes its shared key $SK_U = h(ID_U \| ID_S \| C_U \| C_S \| abP \| t_3)$.
  - U then computes $V = V_1 \oplus h(C_U \| ID_U \| K_2)$ and verifies if
    $S_2 = h(ID_U \| ID_S \| S_1 \| V \| SK_U \| t_1)$.

Therefore, U and S agree on the same shared session key $SK_S = SK_U$ that will be used for further communications.

IV. Security Analysis of Khan et al.'s Scheme

The proposed scheme by Khan et al.[10] is weak against certain security properties of the smart grid environment in the Dolev-Yao model[14]. According to the Dolev-Yao model [14], the adversary controls the traffic of the public channels, the adversary can read all the communication between various parties, can modify or drop transmitted messages. The adversary can also send messages as an authorized party and can replay older messages. Further, the adversary can also encrypt or decrypt messages using known or derived secret keys. The adversary cannot read messages transmitted over the private channels and all other parties are considered to be honest.

A. User Anonymity

As per the security analysis of Khan et al.[10], the LAKAF scheme provides user anonymity. However, it fails to do so as the adversary can obtain the user's identity. In the login phase the user U computes its pseudo identity $I_1 = ID_U \oplus (h(t_1) \oplus t_1)$ and sends $M_1 = \{E_1, I_1, t_1\}$ to the service provider S. The adversary $\mathcal{A}$ can intercept the message and extract ID_U from $ID_U = I_1 \oplus (h(t_1) \oplus t_1)$. Therefore the real identity of the user is compromised.

B. Temporary Keys' Secrecy

The adversary $\mathcal{A}$, can eavesdrop the messages between a user U and service provider S, and derive the temporary session keys as follows:

- $\mathcal{A}$ intercepts all the transmitted messages $M_1 = \{E_1, I_1, t_1\}$ and $M_2 = \{E_2, t_3\}$ between U and S during the login and key agreement phase.
- $\mathcal{A}$ can first retrieve the identity of U as shown in Section IV-A and can then compute the temporary key $K_1 = h(ID_U || h(t_1 \oplus PK_S) || t_1)$ as the terms used to compute K_1, (ID_U, t_1, PK_S) are now available with $\mathcal{A}$. Further, $\mathcal{A}$ can decrypt E_1 to obtain (a, S_1, C_U).
- $\mathcal{A}$ can then compute $S_1' = h(ID_U \| a \| C_U)$ from the earlier obtained terms and verify the validity by checking the condition $S_1 = S_1'$. Now $\mathcal{A}$ can compute the temporary key $K_3 = h(ID_U \| S_1 \| a \| C_U \| t_3)$ using the derived S_1.

C. *Shared Session key Secrecy*

The adversary $\mathcal{A}$ can derive the session key of any ongoing session between U and S as follows:

- $\mathcal{A}$ intercepts all the transmitted messages $M_1 = \{E_1, I_1, t_1\}$ and $M_2 = \{E_2, t_3\}$ between U and S during the login and key agreement phase.
- $\mathcal{A}$ derives the values (K_1, K_3, S_1, a) as shown above in Section IV-B
- $\mathcal{A}$ then uses K_3 to decrypt E_2 to obtain the values (N, V_1, S_2, C_S, b) and also derives $ID_S = N \oplus h(b \| C_S \| C_U)$.
- Finally, using the obtained values, $\mathcal{A}$ can compute the shared session key $SK = h(ID_U \| ID_S \| C_U \| C_S \| abP \| t_3)$.

We can observe from the above analysis, any polynomial time adversary $\mathcal{A}$ can derive the shared session key SK from the intercepted messages without any difficulty under the Dolev-Yao threat model. Furthermore, if $\mathcal{A}$ has a dump of all the previous exchanged messages, then it can calculate the old session keys. This is a serious threat to the security of the smart grid environment.

D. *User Impersonation*

For the adversary $\mathcal{A}$ to impersonate any user, it needs the user's identity ID_U, which the adversary can obtain as mentioned above in Section IV-A. The user registration counter, C_U is not verified at the service provider side, therefore, $\mathcal{A}$ can use any arbitrary string as C_U, construct a valid login message and setup a valid shared session key with S impersonating any user U with ID_U.

E. *Service Provider Impersonation*

By using the parameters $\{ID_S, V_1, C_S, K_1, ID_U, C_U\}$, $\mathcal{A}$ has obtained as detailed in Section IV-B and Section IV-C. $\mathcal{A}$ can calculate $V = V_1 \oplus h(C_U \| ID_U \| K_1)$. Using $\{ID_S, C_S, V\}$, $\mathcal{A}$ can impersonate the Service Provider S, authenticate and setup a shared session keys with any user U.

Remark 1: In the password and biometric change phase of Khan et al's.[10] protocol, the TA does not verify the received credentials and therefore it is possible for an adversary who has obtained the user's identity, $ID_U$to send a valid password and biometric change request.

V. Simulation for formal security verification using Tamarin Prover

Tamarin Prover[15] is an automatic symbolic verification tool used for the formal analysis of security protocols. Tamarin performs an unbounded heuristics based backward search to generate counterexamples to the security claims and if such an example is found it generates a trace. Tamarin has built in theory for widely used cryptographic primitives such as Diffie-Hellman, bilinear pairings, hashing etc. Tamarin assumes the cryptography to be perfect, that is, their security properties cannot be broken by the adversary. Tamarin also allows us to specify our own theory by defining the functions and corresponding equations.

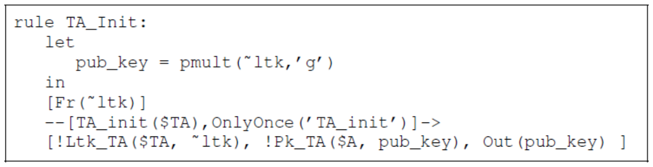
In Tamarin, the system state is represented by a finite multiset of facts. Facts are represented by $F(t_1, \ldots, t_m)$ for a fact symbol F, terms t_j and fixed arity n. The protocol is modeled as a set of rules for the multiset rewriting system. They describe the allowed transitions between the system states. A rule consists of a premise, conclusion and action. The premise represents the multiset of facts that are used up when the rule is applied, the conclusion represents the multiset of facts that are generated by the rule and the action represents a multiset of action facts that label the transition applied by the rule. The lemmas reference these action facts. Facts once consumed by a rule are removed from the system state. However, Facts can also be declared to be persistent, which means they can be used arbitrarily multiple times without being removed from the system state. Persistent facts are represented as $!F(t_1, \ldots, t_n)$. To execute a rule, instances of all the facts in the rule's premise should be present in the current system state. Tamarin has three special builtin facts: *In, Out*, which are use to model the interaction with the open channels that are controlled by the adversary based on the Dolev-Yao model, and *Fr*, which is used to generate fresh random values. The *K* fact represents the adversary's knowledge.

Security properties that are to be verified are specified as lemmas in first-order logic formulas over action facts and time-points. By default, they are intended to hold for all set of cases, which can also be indicated with the keyword *all-traces*, or should hold for at least one trace, indicated with the keyword *exists-trace*. The latter is generally used for checking protocol executabilty. Additionally, restrictions can be used to restrict the set of possible traces that are to be considered for the protocol analysis.

A. *Modelling in Tamarin*

The protocol has been modeled in Tamarin Prover to verify the security properties. Namely, the registration, login and final key agreement phase were modeled in Tamarin Prover. Figure 1 shows the rules for the initialization of the TA, the Public key of the TA is sent on the open channel using the `Out` fact.

Fig. 1. TA Initialization



```
rule TA_Init:
   let
      pub_key = pmult(~ltk,'g')
   in
   [Fr(~ltk)]
   --[TA_init($TA),OnlyOnce('TA_init')]->
   [!Ltk_TA($TA, ~ltk), !Pk_TA($A, pub_key), Out(pub_key) ]
```

The User registers with the TA over secure channels. In Figure 2, the rule `User_Registration` shows the registration of the user with the TA, the identity and secret parameters of the users which include the password and biometric secret key have been generated using the Fr fact. As a modeling simplification, the biometrics of the user have been considered as a randomly generated string. The user credentials along with the TA generated registration credentials are stored with the user as a persistent fact, `!User_state`. Similarly, the Service Provider registers with the TA over secure channels and then declares its public key over the open channel.

Fig. 2. User and Service Provider Registration

```
rule User_Registration:
   let
      g_U = h(<~PW,~fp_U>) XOR ~r_U
      A = h(<~ID_U,~ltk,~C_U>)
      beta = A XOR g_U
      beta1 = beta XOR ~fp_U
      beta2 = h(<~ID_U,~PW,beta1>)
   in
   [Fr(~PW),Fr(~r_U),Fr(~fp_U),Fr(~ID_U),Fr(~C_U), !Ltk_TA(
        $TA, ~ltk)]
   --[User_Reg_done($U),Privacy(~ID_U),OnlyOnce('User_init')
        ]->
   [!User_state($U,~fp_U,~PW,beta,beta1,beta2,~C_U),!User_ID
        ($U,~ID_U)]

rule Server_Registration:
   let
      public_key = pmult(~r_S,'g')
      V = h(<~ID_S,~ltk,~C_S>)
   in
   [Fr(~ID_S),Fr(~r_S),Fr(~C_S),!Ltk_TA($TA, ~ltk)]
   --[Server_Reg_done($S),OnlyOnce('Server_init')]->
   [!Server_state($S,~r_S,V,~C_S), !Server_public($S,
        public_key),
   !Server_ID($S,~ID_S),Out(public_key)]
```

Figure 3 shows the modeling of the user login phase, the user verifies its credentials and encrypts a message with a temporary key K_1 and sends it to the Service Provider on the open channel along with the user's pseudo identity and a timestamp.

Fig. 3. User Login and Authentication Request

```
rule user_login_request:
   let
      B1 = beta XOR ~fp_U
      B2 = h(<ID_U,~PW,B1>)
      S1 = h(<ID_U,~a,~C_U>)
      I1 = ID_U XOR h(~t1) XOR ~t1
      K1 = h(<ID_U,h(~t1 XOR public_key),~t1>)
      E1 = senc(<~a,S1,~C_U>,K1)
      m4 = <E1,I1,~t1>
   in
   [!User_state($U,~fp_U,~PW,beta,beta1,beta2,~C_U), !
        User_ID($U,ID_U),
   !Server_public($S,public_key),Fr(~a), Fr(~t1)]
   --[Secret1(K1), Eq(beta2,B2),OnlyOnce('User_login'),
   User_Initiate($U,$S,<K1,E1>)]->
   [Out(m4),!User_state_2($U,S1,~a,K1,~t1)]
```

Next, the Service Provider receives decrypts and verifies the user's identity and request parameters and then calculates the shared session key. The Service Provider then sends a message encrypted with the temporary secret key $K3$ along with a timestamp as shown in Figure 4.

Fig. 4. Service Provider Shared Key Generation

```
rule server_user_agreement:
   let
      I2 = I1 XOR h(~t1) XOR ~t1
      K2 = h(<I2,h(~t1 XOR pmult(~r_S,'g')),~t1>)
      SK_S = h(<ID_U,ID_S,~C_U,~C_S,pmult(~a,pmult(~b,'g'))
           ,~t3>)
      S2 = h(<ID_S,ID_U,S1,V,SK_S,~t1>)
      V1 = V XOR h(<~C_U,ID_U,K2>)
      N = ID_S XOR h(<~b,~C_S,~C_U>)
      K3 = h(<ID_U,S1,~a,~C_U,~t3>)
      E2 = senc(<N,V1,S2,~C_S,~b>,K3)
      M5 = <E2,~t3>
   in
   [In(<senc(<~a,S1,~C_U>,K2),I1,~t1>),!User_ID($U,ID_U),
   !Server_state($S,~r_S,V,~C_S),!Server_ID($S,ID_S),Fr(~b),
        Fr(~t3)]
   --[Eq(I2,ID_U),Eq(S1,h(<ID_U,~a,~C_U>)),
   Secret2(K3),Server_recv($U,$S,<K2,senc(<~a,S1,~C_U>,K2)>)
        ,
   Server_send($U,$S,<K3,E2>),Secret_SK_S(SK_S),
   Server_User_Agreement($S,$U,SK_S),OnlyOnce('
        Server_Receive')]->
   [Out(M5)]
```

Figure 5 shows the rule for the final step in the keys' agreement phase, wherein the user receives the message from the open channel, decrypts, verifies and finally calculates the shared session key.

Fig. 5. User Shared Key Generation

```
rule user_server_agreement:
   let
      K4 = h(<ID_U,S1,~a,~C_U,~t3>)
      SK_U = h(<ID_U,ID_S,~C_U,~C_S,pmult(~b,pmult(~a,'g'))
           ,~t3>)
      V_Star = V1 XOR h(<~C_U,ID_U,K1>)
   in
   [In(<senc(<N,V1,S2,~C_S,~b>,K4),~t3>),
   !User_state($U,~fp_U,~PW,beta,beta1,beta2,~C_U),
   !User_ID($U,ID_U),!Server_ID($S,ID_S),!User_state_2($U,S1
        ,~a,K1,~t1)]
   --[
   Eq(ID_S,N XOR h(<~b,~C_S,~C_U>)), Eq(S2,h(<ID_S,ID_U,S1,
        V_Star,SK_U,~t1>)),
   Secret_SK_U(SK_U), User_Server_Agreement($S,$U,SK_U),
   User_recv($U,$S,<K4,senc(<N,V1,S2,~C_S,~b>,K4)>),
        OnlyOnce('User_Receive')
   ]->[]
```

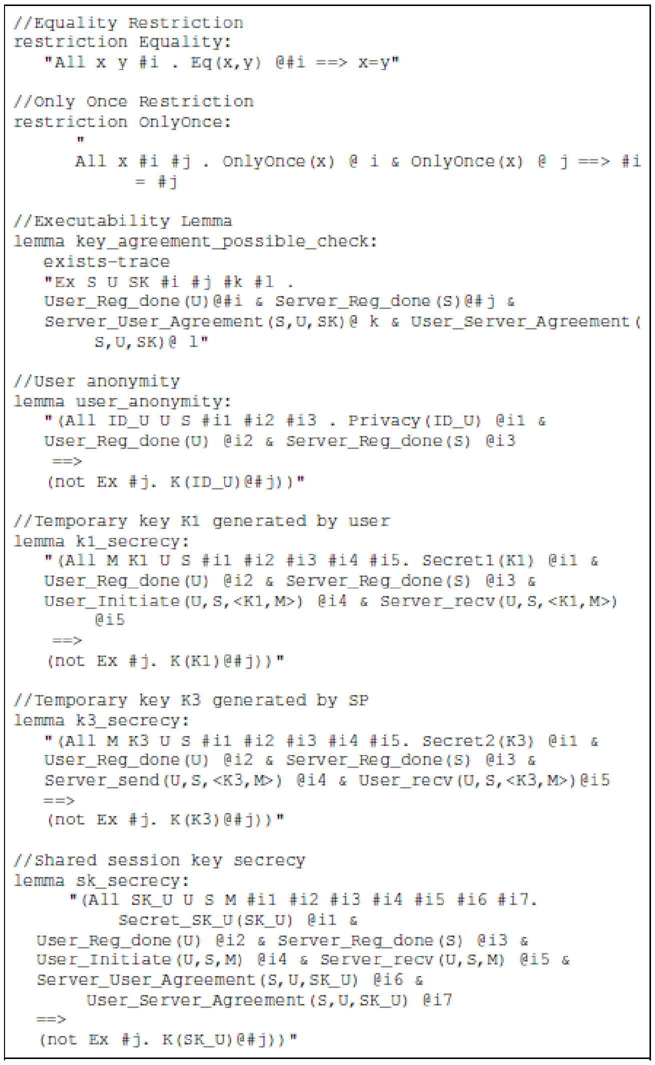
Fig. 6. Restrictions

```
//Equality Restriction
restriction Equality:
   "All x y #i . Eq(x,y) @#i ==> x=y"

//Only Once Restriction
restriction OnlyOnce:
      "
      All x #i #j . OnlyOnce(x) @ i & OnlyOnce(x) @ j ==> #i
           = #j
```

Figure 6 shows the restrictions that have been used, `Equality` restriction has been used for the verification of credentials in different phases of the protocols and the `OnlyOnce` restriction has been used to limit the number of times a rule is executed.

Figure 7 shows the lemmas specified for the key secrecy and user anonymity. The `key_agreement_possible_check` lemma is used to

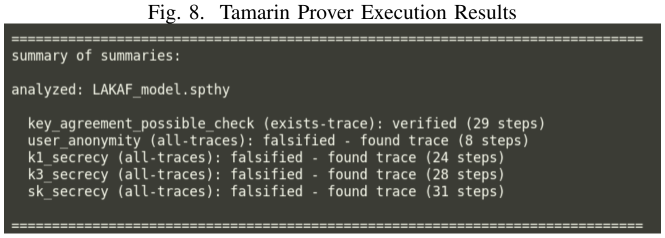
Fig. 7. Security Lemmas



```
//Equality Restriction
restriction Equality:
   "All x y #i . Eq(x,y) @#i ==> x=y"

//Only Once Restriction
restriction OnlyOnce:
      "
      All x #i #j . OnlyOnce(x) @ i & OnlyOnce(x) @ j ==> #i
            = #j

//Executability Lemma
lemma key_agreement_possible_check:
   exists-trace
   "Ex S U SK #i #j #k #l .
   User_Reg_done(U)@#i & Server_Reg_done(S)@#j &
   Server_User_Agreement(S,U,SK)@ k & User_Server_Agreement(
        S,U,SK)@ l"

//User anonymity
lemma user_anonymity:
   "(All ID_U U S #i1 #i2 #i3 . Privacy(ID_U) @i1 &
   User_Reg_done(U) @i2 & Server_Reg_done(S) @i3
    ==>
   (not Ex #j. K(ID_U)@#j))"

//Temporary key K1 generated by user
lemma k1_secrecy:
   "(All M K1 U S #i1 #i2 #i3 #i4 #i5. Secret1(K1) @i1 &
   User_Reg_done(U) @i2 & Server_Reg_done(S) @i3 &
   User_Initiate(U,S,<K1,M>) @i4 & Server_recv(U,S,<K1,M>)
        @i5
    ==>
   (not Ex #j. K(K1)@#j))"

//Temporary key K3 generated by SP
lemma k3_secrecy:
   "(All M K3 U S #i1 #i2 #i3 #i4 #i5. Secret2(K3) @i1 &
   User_Reg_done(U) @i2 & Server_Reg_done(S) @i3 &
   Server_send(U,S,<K3,M>) @i4 & User_recv(U,S,<K3,M>)@i5
   ==>
   (not Ex #j. K(K3)@#j))"

//Shared session key secrecy
lemma sk_secrecy:
     "(All SK_U U S M #i1 #i2 #i3 #i4 #i5 #i6 #i7.
          Secret_SK_U(SK_U) @i1 &
  User_Reg_done(U) @i2 & Server_Reg_done(S) @i3 &
  User_Initiate(U,S,M) @i4 & Server_recv(U,S,M) @i5 &
  Server_User_Agreement(S,U,SK_U) @i6 &
       User_Server_Agreement(S,U,SK_U) @i7
  ==>
  (not Ex #j. K(SK_U)@#j))"
```

check if a successful trace of the protocol execution can take place. The other lemmas are the secrecy lemmas on the user identity, temporary keys $K1$ and $K3$, and final shared session key SK. The lemmas fail to hold as Tamarin successfully shows a trace in which the adversary gets access to the secret parameters. The results from the Tamarin Prover execution are shown in Fig 8. The complete source code and the traces for the lemmas is available on Github at https://github.com/SBhargav15/LAKAF-Tamarin-Model.

Fig. 8. Tamarin Prover Execution Results



```
==============================================================================
summary of summaries:

analyzed: LAKAF_model.spthy

  key_agreement_possible_check (exists-trace): verified (29 steps)
  user_anonymity (all-traces): falsified - found trace (8 steps)
  k1_secrecy (all-traces): falsified - found trace (24 steps)
  k3_secrecy (all-traces): falsified - found trace (28 steps)
  sk_secrecy (all-traces): falsified - found trace (31 steps)

==============================================================================
```

VI. Conclusion

In this paper, we have analyzed the security weakness of Khan et al.'s proposed lightweight authentication protocol. We have shown that the protocol has failed to protect the session key against any adversary under the Dolev-Yao threat model. In addition, we also simulated Khan et al.'s protocol using Tamarin Prover for secrecy of user identity, temporary keys and shared session key and found a trace. As a conclusion, Khan et al.'s protocol is not practical.

In the future work, we aim to present an enhancement over Khan et al.'s protocol to address analyzed security weakness and provide a more robust solution for smart grid network.

References

[1] O. M. Butt, M. Zulqarnain, and T. M. Butt, "Recent advancement in smart grid technology: Future prospects in the electrical power network," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 687–695, 2021.

[2] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security & Privacy*, vol. 7, no. 3, pp. 75–77, 2009.

[3] D. He, H. Wang, M. K. Khan, and L. Wang, "Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography," *IET Communications*, vol. 10, no. 14, pp. 1795–1802, 2016.

[4] J.-L. Tsai and N.-W. Lo, "Secure anonymous key distribution scheme for smart grid," *IEEE transactions on smart grid*, vol. 7, no. 2, pp. 906–914, 2015.

[5] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1900–1910, 2016.

[6] Y. Chen, J.-F. Martínez, P. Castillejo, and L. López, "A bilinear map pairing based authentication scheme for smart grid communications: Pauth," *IEEE Access*, vol. 7, pp. 22 633–22 643, 2019.

[7] T.-Y. Wu, Y.-Q. Lee, C.-M. Chen, Y. Tian, and N. A. Al-Nabhan, "An enhanced pairing-based authentication scheme for smart grid communications," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2021.

[8] A. A. Khan, V. Kumar, M. Ahmad, S. Rana, and D. Mishra, "Palk: Password-based anonymous lightweight key agreement framework for smart grid," *International Journal of Electrical Power & Energy Systems*, vol. 121, p. 106121, 2020.

[9] S. A. Chaudhry, "Correcting "palk: Password-based anonymous lightweight key agreement framework for smart grid"," *International Journal of Electrical Power & Energy Systems*, vol. 125, p. 106529, 2021.

[10] A. A. Khan, V. Kumar, M. Ahmad, and S. Rana, "Lakaf: Lightweight authentication and key agreement framework for smart grid network," *Journal of Systems Architecture*, vol. 116, p. 102053, 2021.

[11] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *International workshop on fast software encryption*. Springer, 2004, pp. 371–388.

[12] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 523–540.

[13] W. Stallings, *Cryptography and network security, 4/E*. Pearson Education India, 2006.

[14] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[15] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 696–701.