Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PROVING THE CAR SECURITY SYSTEM MODEL USING CTL AND LTL

[1]RAFAT ALSHORMAN
[1] Computer Science Department, Faculty of Information Technology & Computer Sciences, Yarmouk University, Irbid-Jordan
E-mail: r.alshorman@yu.edu.jo

ABSTRACT
One of the most important legal issues that troubles the courts is car theft. Therefore, car companies race to build security systems to reduce this problem. In this research, we propose a system based on remote control and a keypad to decrease the risk of car theft. To prove the system, firstly, we described it using a finite-state machine. Secondly, a set of correctness conditions are introduced. These correctness conditions are encoded in temporal logic CTL and LTL. Lastly, the NuSMV model checker is used to verify the correctness conditions. The importance of the proposed system verification is to ensure that the system is correct under these conditions. Additionally, this research also found that CTL and LTL can be used to specify and verify systems of this kind. The main contribution of this research is to demonstrate that CTL and LTL model checking can be used to specify and verify the proposed system. Moreover, this can be a stepping-stone to prove similar systems that their behaviors generate complex finite-state machines and it is difficult to trace all possible states of the system.
Keywords: *CTL, model checking, NuSMV, LTL, Car Security.*

1. INTRODUCTION

Nowadays, security has become one of the requirements of most electronic and information systems, in addition to being a requirement for continuity and guarantee of providing services and maintaining systems. Car theft is one of the most significant issues for judicial and security authorities in most countries. This issue costs individuals, countries, and insurance companies. In the United States, the FBI reports $7.4 billion lost to car theft in 2020 [1]. Therefore, car manufacturers sought to develop security and protection systems. With the advent of smart car technology, car security has become an important feature of smart cars [5]. This research proposes a proving technique for hypothetical car security systems with remote control units and keypad based on smart car requirements using Computational Tree Logic (CTL), Linear-Time Logic (LTL), and NuSMV Model Checker.

This research aims to propose an appropriate proving technique for car security systems with formal specifications and fully automatic verifications. Choosing temporal logic-based techniques are advantageous for these purposes [3], [4], [7], [8], [9]. A set of synchronous and asynchronous processes, protocols, and conditions can be described and demonstrated using temporal logic specifications. The temporal logic types with temporal restrictions and correctness conditions model the proposed system implementation.

In Figure 5, we show how the proposed technique is used to prove a system's correctness conditions and generate a counterexample when one of the conditions is not met. This will lead to modifying the proposed system to meet the car manufacturing company requirements.

2. THE PROPOSED CAR SECURITY SYSTEM

In this research, we proposed a car security system that is based on remote control unit and smart car key in addition with keypad. Most modern cars contain these elements. The issue here is to propose a simple protocol to make sure that these elements together will reduce the possibility of car theft. Figure 1 shows the finite state machine of the proposed system. The operations of how the driver can turn the car on are described by as follows rules or instructions:

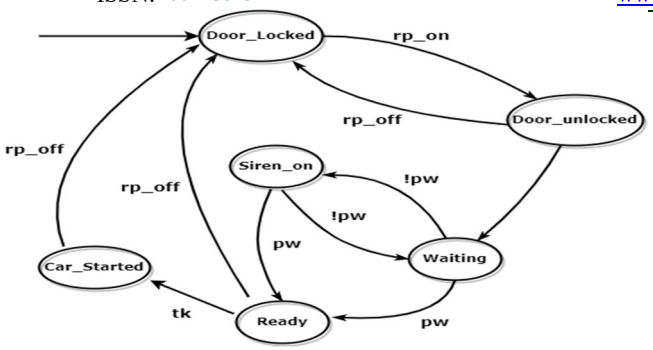

Figure 1: Finite state machine of the proposed system

1. He/she should press on remote control unit to unlock the car doors ($Door\_locked$ state). This operation is represented by rp_on input which transits to a new state $Door\_unlocked$.
2. After that, He/she can return to the initial state ($Door\_locked$ state) by pressing on remote control unit. This operation is represented by rp_off input which transits to the initial state $Door\_locked$. This operation could happen if the driver opens the car doors by mistake or brings something from the car without any intent to turn the car on.
3. To activate the keypad, the car key should be in the ignition hole. However, the car will not start even if the driver tries to move the key. The result of this operation will activate the keypad ($keypad\_activated$ state).
4. Now, the car is in waiting state ($Waiting$ state). This represents the case where the car waits for the driver to enter password.
5. If the driver entered a correct password (represented by pw input), the car moves to $ready$ state. This state means that the driver is now ready to move the car key to start the car.
6. If the driver entered an incorrect password (represented by $!pw$ input), the car moves to $Siren\_on$ state. This state means that the siren will start to notify the owner that there is an unusual operation happening to the car. In this case, we have two scenarios. First, the driver is the real owner of the car, but he/she typed the password on the keypad wrongly. Now, the driver can enter a correct password again to move the car into $ready$ state. This represented by transition edge from $Siren\_on$ state to $ready$ state). Second, the driver is not the real owner of the car, but he/she typed the password on the keypad wrongly and he/she keeps entering wrong password (This represented by a cycle in $Waiting$ state and $Siren\_on$ state in Figure 1). In this case the car will stay in $Siren\_on$ state to notify the real owner that there is unusual operation happening to the car.
7. After the driver enters a correct password (the car into $ready$ state), he/she can now turn the ignition switch further by car key to the start the car. This represents by tk input and state $Car\_started$ state.
8. After entering correct password, at states $Car\_started$ or $ready$, the driver can press on remote control unit to lock the car doors kind of personal security because it is better to lock the door before moving the car forward or backward. This represented by edges labeled by rp_off from states $ready$ and $Car\_started$ to the initial state, respectively.

Generally, Information and Communication Technology (ICT) systems consider authentication to be strong when at least two factors of identity are satisfied (see [10] and [11]):

1. something you are (or can do)
2. something you have
3. something you know.

We inherited this authentication process in the proposed protocol for car security system mentioned above. So, the driver cannot start the car on without having the car key, which is something you have, and entering a correct password which is something you know. Therefore, two factors of identity are satisfied. Hence, we have a strong authentication in this proposed system. For example, if the thief stole the car key (something you have), he/she could not start the car because it must know the password (something you know).

All the above rules of the proposed protocol (rules 1-8) will be modeled by NuSMV model checkers. To ensure that the NuSMV model matches the proposed system, a set of correctness conditions will be introduced. Moreover, a set of correctness conditions will be introduced to prove the correctness of the proposed protocol using Computational Tree Logic (CTL) along with Linear Temporal Logic (LTL), see Section 4.

3. TEMPORAL LOGIC

Temporal logic is a branch of mathematical logic that deals with events and time. Thus, it is used to analyze the behavior of a system over time. Temporal logic is indispensable, as shown in [12], [13], [14], and [15]. It is possible for temporal logic operators to reason about synchronous and asynchronous events and scenarios. In addition, temporal logic formulas can be used to express complex system behavior and properties. In the following subsections, we shall introduce two types of temporal logic: CTL and LTL.

3.1. Syntax of CTL

We can use CTL to specify multiple possible futures or events in any system. As a result, one can determine whether a given property holds for all or some possible futures of the system behavior [15], [16], [17]. These abilities can be achieved via the following operators:

1. Ordinary Boolean operations¬,∨,∧, → , ⊤, ⊥, quantifiers Existential ($\boldsymbol{E}$), $\boldsymbol{A}$ (Universal).
2. Temporal operators $\boldsymbol{X}$, $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{U}$.

In this research, any CTL formula Φ consists of a set of Atomic Propositions (APs), that are used in the proposed system, such as rp_on, rp_off, pw, tk. This set of APs will refer them by $p_i, i = 0,1,2,\ldots$. Hence, the formulas in CTL can be written by:

$$\phi ::= p_i \mid \neg \Phi \mid \Phi_1 \vee \Phi_2 \mid$$

$$\Phi_1 \wedge \Phi_2 \mid \boldsymbol{AX}\Phi \mid \boldsymbol{E}[\Phi_1 \boldsymbol{U} \Phi_2] \mid \boldsymbol{EX}\Phi \mid \boldsymbol{AF}\Phi$$

$$\mid \boldsymbol{EF}\Phi \mid \boldsymbol{AG}\Phi \mid \boldsymbol{EG}\Phi \mid \boldsymbol{A}[\Phi_1 \boldsymbol{U} \Phi_2] .$$

3.2. Semantics of CTL

We shall determine that any atomic proposition p_i is true at any state or time s_i of the proposed system M such that: $M, s_i \vDash p_k$, for all $p_k \in p_i$. The system M is a structure that is defined by a tuple (S, I, R, AP, L) such that (see [16]):

S: is a finite set of states.

$I \subseteq S$: is a finite set of initial states.

R: is a total transition relation such that

$$R \subseteq S \times S.$$

AP: is the set of atomic propositions

L: is a function which determine to each state the set of atomic propositions that are true in that state

such that $L: S \to 2^{AP}$.

The semantics for the ordinary operators are defined as follows:

$$M, s_i \vDash \neg \phi \quad iff \quad M, s_i \nvDash \phi.$$

$$M, s_i \vDash \phi \wedge \psi \quad iff \quad M, s_i \vDash \phi \ and \ M, s_i \vDash \psi.$$

$$M, \vDash \phi \vee \psi \ iff \ M, s_i \vDash \phi \ or \ M, s_i \vDash \psi.$$

$$M, s_i \vDash \phi \Rightarrow \psi \ iff \ if \ M, s_i \vDash \phi \text{ then } M, s_i \vDash \psi.$$

Now, let λ be a path starting from a state s_i such that $\lambda = s$i, si+1, ... in the system model M. The following temporal operators can be interpreted over model M such that:

$M, s_i \vDash \boldsymbol{AX}\Phi iff \ \forall \lambda = (s_i, s_{i+1}, \ldots), \ M, s_{i+1} \vDash \Phi.$

$M, s_i \vDash \boldsymbol{EX}\Phi \ iff \ \exists \lambda = (s_i, s_{i+1}, \ldots), M, s_{i+1} \vDash \Phi.$

$M, s_i \vDash \boldsymbol{AF}\Phi \ iff \ \forall \lambda = (s_i, s_{i+1}, \ldots), \exists \ j \geq i,$
$M, s_j \vDash \Phi.$
$M, s_i \vDash \boldsymbol{EF}\Phi$ i$ff \ \exists \lambda = (s_i, s_{i+1}, \ldots), \exists \ j \geq$
$i, M, s_j \vDash \Phi.$

$M, s_i \vDash \boldsymbol{AG}\phi \ iff, \forall \lambda = (s_i, s_{i+1}, \ldots)$,and $\forall j \geq$
$i, M, s_j \vDash \phi.$

$M, s_i \vDash \boldsymbol{EG}\phi \ iff \ \exists \lambda = (s_i, s_{i+1}, \ldots)$, and $\forall j \geq$
$i, M, s_j \vDash \phi$.

$M, s_i \vDash \boldsymbol{A}[\phi_1 \boldsymbol{U} \phi_2] \ iff \ \forall \lambda = (s_i, s_{i+1}, \ldots),$
$\exists \ j \geq i$ such that $M, s_j \vDash \phi_2$and,$\forall k, i \leq k <$
$j, M, s_k \vDash \phi_1.$

$M, s_i \vDash \boldsymbol{E}[\phi_1 \boldsymbol{U} \phi_2] \quad iff \; \exists \lambda = (s_i, s_{i+1}, \ldots)$such that $\exists j \geq i, M, s_j \vDash \phi_2$and,$\forall k, i \leq k < j, M, s_k \vDash \phi_1$.

3.3. LTL Syntax

Linear-Time Temporal Logic (LTL) is used to describe and reason about the behavior of systems over time [18], [19]. A system's properties can be specified at different points in time. This makes it ideal for verifying software and hardware systems. In contrast to expressing just individual states, it allows us to express properties that hold over sequences of states or events. A temporal relationship can be specified between events in LTL [19]. The proposed system's correctness conditions and properties are encoded using CTL and LTL in this study. This aims to determine their differences in the context of such systems.

As in CTL, LTL consists of a set of propositions symbols $p_i, i = 0,1,2, \ldots$, booleans operators $\neg, \vee, \wedge, \rightarrow, \top, \perp$, and temporal operators $\boldsymbol{X}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{U}$. Formulas in LTL are those generated by:

$\phi ::= p_i \mid \neg\varphi \mid \varphi 1 \vee \varphi 2 \mid \varphi 1 \wedge \varphi 2 \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi 1 U \varphi 2 \mid$.

3.4. LTL Semantics

Similarly, as in CTL, if ϕ is a path formula, $M, s_i \vDash p_k$, where the path is such that $\lambda = si, si{+}1, \ldots$. This means that ϕ holds along path λ in the model structure M. The relation is defined inductively as follows (see [19]) :

$M, s_i \vDash \neg \phi \quad iff \quad M, s_i \nvDash \phi$

$M, s_i \vDash \phi \wedge \psi \; iff \; M, s_i \vDash \phi \; and \; M, s_i \vDash \psi$

$M, \vDash \phi \vee \psi \; iff \; M, s_i \vDash \phi \; or \; M, s_i \vDash \psi$

$M, s_i \vDash \phi \Rightarrow \psi \; iff \; if \; M, s_i \vDash \phi$ then $M, s_i \vDash \psi$

$M, s_i \vDash \boldsymbol{X}\Phi \; iff \quad M, s_{i+1} \vDash \Phi.$

$M, s_i \vDash \boldsymbol{F}\phi \; iff \; \exists \, j \geq i, \; M, s_j \vDash \Phi$
$M, s_i \vDash \boldsymbol{G}\phi \; iff \; \forall j \geq i, M, s_j \vDash \phi$

$M, s_i \vDash \boldsymbol{A}[\phi_1 \boldsymbol{U} \phi_2] \; iff \; \exists \, j \geq i$ such that $M, s_j \vDash \phi_2$ and, $\forall k, i \leq k < j, M, s_k \vDash \phi_1$.

4. SPECIFICATIONS OF THE PROPOSED CAR SECURITY SYSTEM

Specifications is a fundamental concept in real systems correctness. Specifications refer to a set of requirements that a system must meet. In other words, specifications define what a system should do [20], [21]. Therefore, in this section, we shall introduce a set of correctness conditions for the proposed model that should be satisfied. We consider the proposed model is correct if and only if all the following conditions and properties are satisfied:

1. At any time, if the driver has a key and enters a correct password, then the car can be started. This condition is encoded into CTL and LTL, respectively, as follows:

$$\alpha_1 = AG\ (pw \wedge tk \Rightarrow car\_started)$$
$$\beta_1 = G\ (pw \wedge tk \Rightarrow car\_started)$$

2. At any time, the driver cannot start the car using the key without entering the correct password. This can be encoded by CTL and LTL, respectively, as follows:

$$\alpha_2 = AG\ (tk \Rightarrow pw\ )$$
$$\beta_2 = G\ (tk \Rightarrow pw\ )$$

3. At any time, the siren will not eventually be sounding if the driver knows the correct password. The CTL and LTL specifications of this case, respectively, as follows:

$$\alpha_3 = AG(pw \Rightarrow AF \neg Siren\_on\ )$$
$$\beta_3 = G(pw \Rightarrow F \neg Siren\_on\ )$$

4. At any point, the siren will not be sounding if the driver locks the car doors by remote control unit. This can be encoded in CTL and LTL such that:

$$\alpha_4 = AG(Siren\_on \Rightarrow \neg rp\_on)$$
$$\beta_4 = G(Siren\_on \Rightarrow \neg rp\_on)$$

To ensure the safety and reliability of our proposed model, we shall add the following properties that the model should meet:

5. There is no eventually a chance to start the car with the wrong password. This is encoded as follows:

$$\alpha_5 = \neg EF\ ((\neg pw\ \wedge car\_started\ ))$$
$$\beta_5 = \neg F\ ((\neg pw\ \wedge car\_started\ ))$$

6. There is no eventually a chance to start the car without having a key. This is encoded as follows:

$$\alpha_6 = \neg EF\ ((\neg tk\ \wedge car\_started\ ))$$
$$\beta_6 = \neg F\ ((\neg tk\ \wedge car\_started\ ))$$

Now, suppose we have a model M representing the proposed car security model and a formula Φ representing the correctness conditions in CTL and/or LTL such that:

$$\Phi = \bigwedge_{1 \le i \le 6} (\alpha_i \vee \beta_i)\,.$$

Then, for all $s_j \in S$ (set of states).

5. NUSMV MODEL CORRESPONDING TO CAR SECURITY MODEL

The purpose of this section is to describe the proposed model in the NuSMV script. By doing so, the model is ensured to meet the correctness conditions and properties mentioned above. The NuSMV script is a specialized language for describing Finite State Machines (FSM) (see [22] and [23]). Compared to traditional propositional logic, temporal logic specifications offer several advantages. The ability to reason about time and change is one of its most important advantages. Many real-world systems are dynamic in nature, and traditional propositional logic cannot capture this. On the other hand, temporal logic allows complex temporal properties and correctness conditions to be expressed. As a result, it is ideal for modeling real-time systems [24]. Many researchers use various techniques to prove such systems in order to find the best proof that can be used to modify the system as required [2], [16]. The proposed technique aims to prove the proposed correctness conditions of the system through model checker. After that, the model checker will generate the state-space of all possible states for a system and interpret temporal logic formulae over it. Two possible results are generated, true and false. True means that the correctness condition is satisfied (proved). False means that the correctness condition is not satisfied (disprove). This will generate a counterexample to show the designer of the system the defect in the system and help to modify it as required. This is depicted in Figure 5 below.

5.1. State Variables of the Corresponding Model

In this subsection, the state variables, correctness conditions, and properties will be introduced in the corresponding NuSMV model. FSM in Figure 1, which represents the proposed model, is described in NuSMV as follows:

State s1 represents door_locked state.
State s2 represents door_unlocked state.
State s4 represents waiting state.
State s5 represents siren_on state.
State s6 represents ready state.
State s7 represents car_started state.

Now the correctness conditions and properties are described in NuSMV as follows:

At any time, if the driver has a key and enters a correct password, then the car can be started. This condition is encoded into CTL and LTL as follows:

$$\alpha_1 = AG\ (pw \wedge tk \Rightarrow car\_started)$$
$$\beta_1 = G\ (pw \wedge tk \Rightarrow car\_started).$$

This condition can be written in NuSMV script such that:

$$\boldsymbol{SPEC\ AG}(\boldsymbol{pw}\ \&\ \boldsymbol{tk} \text{ -> } \boldsymbol{state = s7})$$
$$\boldsymbol{LTLSPEC\ G}(pw\ \&\ tk \text{ -> } state = s7).$$

SPEC and *LTLSPEC* (in NuSMV) denotes to CLT and LTL Specifications language, and the operators *AG* and *G* have the same meaning as they defined in Section 3. The other properties and correctness conditions can be encoded in the same way, See Figure 2.

```
MODULE main
VAR
state : {s1, s2, s4, s5, s6, s7};
DEFINE
pw := state in {s6, s7};
tk := state = s7;
rp_on := state = s1;
rp_off := {s2, s4, s5, s6, s7};

-- s1: door_locked
-- s2: door_unlocked
```

Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
-- s4: waiting
-- s5: siren_on
-- s6: ready
-- s7: car_started

ASSIGN
init(state ) := {s1};
next(state) :=
case
state = s1: {s1, s2};
state = s2: {s1, s2, s4};
state = s4: {s5,s6};
state = s5: {s1, s4};
state = s6: {s1, s7};
state = s7: {s1, s7};
TRUE: state;
esac;
--LTL specifications:
 LTLSPEC G(pw & tk -> state=s7)
 LTLSPEC G( tk -> pw)
 LTLSPEC G( pw -> F state!=s5)
 LTLSPEC G(state=s5 -> !rp_on)
 LTLSPEC !F(!pw & state=s7 )
 LTLSPEC !F(!tk & state=s7 )

--CTL specifications:
SPEC AG(pw & tk -> state=s7)
SPEC AG( tk -> pw)
SPEC AG( pw -> AF state!=s5)
SPEC AG(state=s5 -> !rp_on)
SPEC !EF(!pw & state=s7 )
SPEC !EF(!tk & state=s7 )
```

Figure 2: NuSMV script for the proposed system

6. RESULTS AND DISCUSSION

We briefly describe the runs of the model in NuSMV model checker in this section. NuSMV model checker evaluates the CTL and LTL specifications in section 4 to determine their truth or falsity in the finite state machine model in Figure 1. A counterexample of the FSM that falsifies a specification is constructed and printed by NuSMV when the specification is discovered to be false. NuSMV returns true if the specification is true otherwise. Based on the run in Figure 3, all properties and conditions introduced in section 4 hold in all situations, see Figure 3. Also, to demonstrate NuSMV's ability to detect false conditions, the following assumptions are made:

You may have entered the wrong password on the keypad even though the car is ready to start.

This condition can be encoded into CTL and LTL as follows:

$$\alpha_7 = EF\ (ready\ \wedge \neg pw\ )$$
$$\beta_7 = F\ (ready\ \wedge \neg pw\ )$$

This condition is falsified by NuSMV, and a counterexample is generated in both LTL and CTL, as shown in Figure 4.

```
*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification AG ((pw & tk) -> state = s7)  is true
-- specification AG (tk -> pw)  is true
-- specification AG (pw -> AF state != s5)  is true
-- specification AG (state = s5 -> !rp_on)  is true
-- specification !(EF (!pw & state = s7))  is true
-- specification !(EF (!tk & state = s7))  is true
-- specification  G ((pw & tk) -> state = s7)  is true
-- specification  G (tk -> pw)  is true
-- specification  G (pw ->  F state != s5)  is true
-- specification  G (state = s5 -> !rp_on)  is true
-- specification !( F (!pw & state = s7))  is true
-- specification !( F (!tk & state = s7))  is true
```

Figure 3: NuSMV run

```
-- specification EF (state = s6 & !pw)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    state = s1
    rp_off = ((s2 union s4) union s5) union s6, s7
    rp_on = TRUE
    tk = FALSE
    pw = FALSE
-- specification  F (state = s6 & !pw)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 2.1 <-
    state = s1
    rp_off = ((s2 union s4) union s5) union s6, s7
    rp_on = TRUE
    tk = FALSE
    pw = FALSE
  -> State: 2.2 <-
    state = s2
    rp_on = FALSE
  -> State: 2.3 <-
    state = s4
  -> State: 2.4 <-
    state = s6
    pw = TRUE
  -> State: 2.5 <-
    state = s1
    rp_on = TRUE
    pw = FALSE
```

Figure 4: NuSMV Counterexample.

7. CONCLUSION

In this research, the NuSMV model checker and temporal logic are used to prove the proposed car security system. The purpose of this paper is to answer some questions about the capability of CTL and/or LTL to specify proposed systems. Furthermore, NuSMV model checker can enhance and verify the proposed system. It is possible in this study to specify such a system using temporal logic, and the NuSMV model checker can verify it. NuSMV models return true when the correctness conditions are met and false when counterexamples are provided. A system like this can identify errors or bugs before they are applied in the real world. Moreover, using NuSMV to verify the system would operate correctly under a wide range of correctness conditions to ensure it is free of error. This study shows that the CTL correctness conditions representing by $\alpha_1, \alpha_2, \ldots, \alpha_6$ and the LTL correctness conditions representing by $\beta_1, \beta_2, \ldots, \beta_6$ are satisfied by the proposed system, which means that these correctness conditions are true in all of the system states cases. On the other hand, the correctness conditions α_7 and β_7 are not satisfied by the system, and counterexamples are generated to show the states where these unsatisfied correctness conditions are violated. It is known that LTL is used to verify liveness properties of reactive systems and CTL is used to verify safety properties of concurrent systems. But, in this research, both CTL and LTL can be used to specify the proposed car security system. Also, this could be the first step to specify and verify more complicated similar systems. Finally, this research provides a specification of the proposed system in CTL and LTL suitable for proving correctness conditions and properties of the proposed system. Similar specifications can be used for any type of system that generates complicated system behavior. As a result of this research, we were able to introduce and use CTL and LTL in the context of car security protocols that use various types of tokens for security.

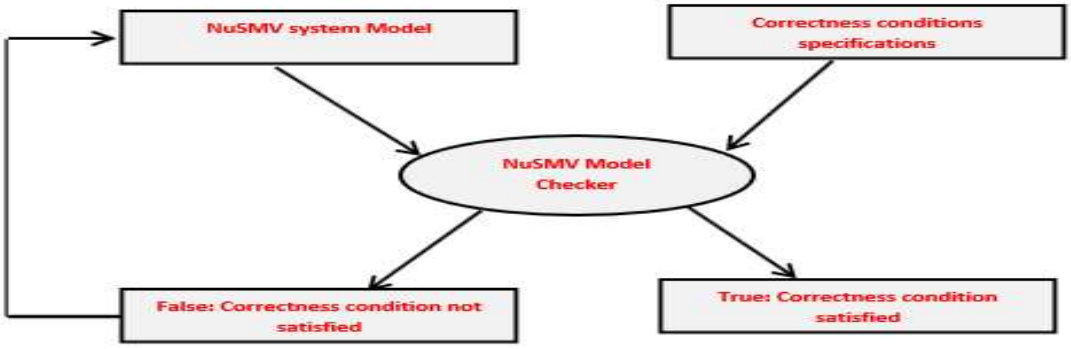

Figure 5: Model Checker

REFERENCES

[1]. The National Insurance Crime Bureau (NICB). https://www.nicb.org/ last visit 30/6/2023.

[2]. A. Alomari, R. Alshorman, "Proving the Correctness Conditions of the Three-Way Handshake Protocol Using Computatinal Tree Logic", *journal of Theoretical and Applied Information Technology*, ,Vol. 99 , No. 15, 2021, pp. 3725-3735.

[3]. Hassan, Z., Bradley, A. R., & Somenzi, F.. "Incremental, inductive CTL model checking". *International Conference on Computer Aided Verification*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 532-547.

[4]. Classen, A., Cordy, M., Schobbens, P. Y., Heymans, P., Legay, A., & Raskin, J. F.. "Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking". *IEEE Transactions on Software Engineering*, Vol. 39, No. 8, 2012, pp. 1069-1089 .

[5]. Wang, H., Gu, M., Wu, S., & Wang, C., "A driver's car-following behavior prediction model based on multi-sensors data". *EURASIP Journal on Wireless Communications and Networking*, Vol. 2020, No. 1, 2020, pp. 1-12,

[6]. Popov, G., & Angelov, P. "A Car Security System Model Based on Timed Petri

Nets". *Advanced Aspects of Theoretical Electrical Engineering Sozopol,* 2009.

[7]. Alshorman R. and Hussak W.," A CTL Specification of Serializability for Transactions Accessing Uniform Data*", International Journal of Computer and Information Engineering*, Vol. 3. , No. 3, 2009, pp. 780-786.

[8]. Casoni M., Grazia C. A., Klapez M., and Patriciello N."Implementation and validation of TCP options and congestion control algorithms for ns-3". *In Proceedings of the 2015 Workshop on ns-3 (WNS3 '15). ACM*, New York, NY, USA, 2015, pp.112-119.

[9]. Bhushan, Ram C., and Dharmendra K. Yadav. "Formal Specification and Verification of Data Separation for Muen Separation Kernel." *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)* .Vol. 15, No. 2, 2022, pp. 274-283, 2022.

[10]. Debiao, H., Jianhua, C., & Jin, H. " An ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security". *Information Fusion*, Vol 13. No 3, 2012, pp. 223-230.

[11]. Yin L, Fang B, Guo Y, Sun Z, Tian Z. "Hierarchically defining Internet of Things security: From CIA to CACA*", International Journal of Distributed Sensor Networks*. Vol. 16, No. 1, 2020.

[12]. Cai, Mingyu, et al. "Learning minimally-violating continuous control for infeasible linear temporal logic specifications." *2023 American Control Conference (ACC).* IEEE, 2023, pp. 1446-1452.

[13]. Walker M, Khazra P, Nanah Ji A, Wang H, van Breugel F., "jpf-logic: a Framework for Checking Temporal Logic Properties of Java Code", *ACM SIGSOFT Software Engineering Notes*.,Vol. 48, No. 1, 2023, pp. 32-36.

[14]. R. Alshorman and W. Hussak, "Multi-step transactions specification and verification in a mobile database community," *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, Damascus, Syria, 2008, pp. 1-6.

[15]. Abdelrahman, R. B., Alshorman, R., Hussak, W., & Trehan, A, "Specification of synchronous network flooding in temporal logic", *International Arab Journal of Information Technology*, Vol. 17, No. 6, , 2020, pp. 867-874.

[16]. R. Alshorman, "Toward Proving the Correctness of TCP Protocol Using CTL", *International Arab Journal of Information Technology (IAJIT)* ,Vol. 16, No. 03, 2019, pp. 85 - 92.

[17]. Gao, H., Zhou, L., Kim, J. Y., Li, Y., & Huang, W., "Applying probabilistic model checking to the behavior guidance and abnormality detection for A-MCI patients under wireless sensor network". *ACM Transactions on Sensor Networks*, Vol. 19, No. 3, 2023, pp. 1-24.

[18]. Gadducci, Fabio, Andrea Laretto, and Davide Trotta. "Specification and verification of a linear-time temporal logic for graph transformation." *International Conference on Graph Transformation. Cham: Springer Nature,* Switzerland, 2023, pp. 22-42.

[19]. R. Alshorman, and W. Hussak, "Specifying a timestamp-based protocol for multi-step transactions using LTL", *International Journal of Computer and Information Engineering*, Vol. 4, No 11, 2010, pp.1716-1723.

[20]. G. Amram, D. Ma'ayan, S. Maoz, O. Pistiner and J. O. Ringert, "Triggers for Reactive Synthesis Specifications," *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, Melbourne, Australia, 2023, pp. 729-741.

[21]. A. Badithela, J. B. Graebener, W. Ubellacker, E. V. Mazumdar, A. D. Ames and R. M. Murray, "Synthesizing Reactive Test Environments for Autonomous Systems: Testing Reach-Avoid Specifications with Multi-Commodity Flows," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, 2023, pp. 12430-12436.

[22]. Zhong, Deming, Rui Sun, Haoyuan Gong, and Tianhuai Wang.. "System-Theoretic Process Analysis Based on SysML/MARTE and NuSMV", *Applied Sciences,* Vol 12, No. 3: 1671, 2022.

[23]. Ma, Junda, Guoxin Wang, Jinzhi Lu, Shaofan Zhu, Jingjing Chen, and Yan Yan. "Semantic Modeling Approach Supporting Process Modeling and Analysis in Aircraft Development" *Applied Sciences* 12, no. 6: 3067, 2022.

[24]. Huo, Xu, et al. "A dynamic soft sensor of industrial fuzzy time series with propositional linear temporal logic." *Expert Systems with Applications*, Vol 201, 2022, pp. 117176.