Sistemas de Control de Acceso a Vehículos Electrónicos Tesla con Smart Cards y Especificaciones Formales

D. León[1], K. Haytara[2]
Universidad La Salle de Arequipa, Perú
[1]dleona@ulasalle.edu.pe, [2]khaytarar@ulasalle.edu.pe

***Resumen*—La seguridad en el acceso a vehículos ha evolucionado con la adopción de tecnologías avanzadas, como las tarjetas inteligentes (smart cards). Tesla implementa un sistema basado en MIFARE, que opera bajo la norma ISO 14443 Tipo A a 13.56 MHz, permitiendo la autenticación del usuario mediante proximidad. Además, Tesla complementa esta funcionalidad con llaves digitales en dispositivos móviles y Bluetooth. Para garantizar la fiabilidad del sistema, se emplean especificaciones formales, como Redes de Petri, Lógica Temporal y lenguajes de modelado como Z o B, alineándose con la norma ISO 26262 para la seguridad funcional en automoción. La aplicación de estos métodos permite la verificación rigurosa del sistema, reduciendo vulnerabilidades y mejorando la seguridad.**

***Palabras clave*—Control de acceso, Smart Cards, Tesla, VDM++, Especificación formal, Seguridad funcional, Redes de Petri, Lógica Temporal, Validación.**

I. Introducción

La seguridad y el control de acceso en vehículos han evolucionado con la adopción de tecnologías avanzadas, como las tarjetas inteligentes (smart cards). Tesla ha integrado esta tecnología en sus vehículos, permitiendo a los usuarios desbloquear y encender el automóvil mediante autenticación sin contacto. Este sistema mejora la seguridad y la comodidad, reduciendo la dependencia de llaves físicas tradicionales.

Tesla emplea tarjetas inteligentes basadas en MIFARE, que operan a una frecuencia de 13.56 MHz. Estas tarjetas, en combinación con otros métodos de acceso como llaves digitales en teléfonos móviles y conectividad Bluetooth, garantizan un acceso seguro y flexible. Sin embargo, para asegurar la fiabilidad de estos sistemas, es crucial aplicar enfoques de especificación formal, los cuales permiten modelar y verificar el comportamiento del sistema antes de su implementación.

En este contexto, el presente trabajo tiene como objetivo general analizar el sistema de control de acceso de Tesla basado en smart cards, destacando su tecnología, gestión de llaves y el uso de métodos formales para garantizar su seguridad y fiabilidad.

A. Objetivos Específicos

- Describir el funcionamiento de las tarjetas inteligentes MIFARE en los vehículos Tesla y su integración con otros métodos de acceso.
- Analizar la gestión de llaves y el control de accesos mediante la interfaz del vehículo.
- Explorar el uso de especificaciones formales como VDM++, Redes de Petri y Lógica Temporal para la validación del sistema de acceso.

II. Trabajos Relacionados

Los sistemas de control de acceso a vehículos han evolucionado con el uso de tecnologías inteligentes, permitiendo una autenticación segura y sin contacto. Diversos estudios han analizado las ventajas y desafíos de estas soluciones en términos de seguridad, usabilidad y eficiencia.

Hernández y Díaz [**?**] desarrollaron un sistema de control de acceso basado en Java Cards y hardware libre, empleando la plataforma Arduino para gestionar puertas y registrar el ingreso de personal. Este enfoque demuestra la viabilidad de integrar tarjetas inteligentes en entornos de bajo costo sin sacrificar la seguridad.

González López [**?**] diseñó un sistema de control de acceso con tarjetas RFID inteligentes en una biblioteca, permitiendo la identificación y manipulación de información de usuarios de forma automatizada. Su propuesta resalta la agilidad operativa y la disminución del uso de documentos físicos.

Desde la perspectiva industrial, Visionis Technologies [**?**] discuten cómo las tarjetas inteligentes podrían sustituir a las tarjetas de banda magnética en campus universitarios. Argumentan que estas nuevas credenciales permiten una autenticación más segura mediante chips programables y almacenamiento cifrado.

Finalmente, PYV Technology [**?**] ofrece una visión práctica sobre el funcionamiento de sistemas de control de acceso con tarjeta, detallando cómo la tecnología permite gestionar autorizaciones en tiempo real y prevenir accesos no autorizados en instalaciones corporativas.

Estos trabajos evidencian que las smart cards son una solución viable y segura para sistemas de control de acceso, y que su integración con tecnologías abiertas y prácticas modernas puede mejorar significativamente la gestión de accesos en diversos contextos.

A. Definición de Sistemas de Control de Acceso a Vehículos Electrónicos Tesla

Los sistemas de control de acceso son fundamentales en el ámbito de la seguridad vehicular. En el caso de los vehículos eléctricos como los de Tesla, estos sistemas garantizan que

solo personas autorizadas puedan acceder y operar el vehículo. El uso de tarjetas inteligentes (Smart Cards) en estos sistemas se ha popularizado debido a su seguridad avanzada y facilidad de uso. Estas tarjetas emplean tecnología de comunicación inalámbrica para autenticar al usuario de manera rápida y segura. Además, la integración con software de última generación mejora la experiencia del usuario, permitiendo la apertura y el arranque del vehículo sin necesidad de una llave física.

A pesar de sus ventajas, la seguridad de estos sistemas es crítica, ya que una vulnerabilidad en el proceso de autenticación podría permitir accesos no autorizados al vehículo, comprometiendo tanto la propiedad como la seguridad del conductor. Por lo tanto, es esencial observar rigurosamente los avances tecnológicos y las regulaciones en materia de protección de datos y privacidad [**?**].

El impacto de los sistemas de control de acceso en los vehículos es significativo no solo en términos de seguridad, sino también en la experiencia general del usuario. A través de estos sistemas, Tesla ha logrado integrar la automatización con la personalización del acceso, permitiendo que cada conductor o usuario autorizado tenga configuraciones específicas que se activan al momento de acceder al vehículo [**?**], [**?**].

Dentro de los aspectos más relevantes que se analizan en este trabajo se encuentran:

1) **Autenticación mediante Smart Cards:** Las tarjetas inteligentes actúan como el mecanismo principal de autenticación, garantizando que solo los usuarios preautorizados puedan acceder al vehículo. Estas tarjetas utilizan estándares de comunicación como NFC o Bluetooth para autenticar la identidad del usuario y permitir el acceso al sistema.
2) **Seguridad en la Comunicación:** Dado que la comunicación entre la tarjeta inteligente y el vehículo es inalámbrica, es fundamental garantizar la seguridad de los datos transmitidos. Esto se logra mediante el uso de encriptación de alto nivel y protocolos de autenticación robustos.
3) **Integración con Sistemas de Gestión de Vehículos:** Los sistemas de control de acceso no solo permiten el desbloqueo del vehículo, sino que también se integran con otras tecnologías del vehículo, como los sistemas de asistencia al conductor y la gestión de la batería, para optimizar la experiencia del usuario.
4) **Autenticación de Múltiples Niveles:** Para garantizar una mayor seguridad, se pueden implementar mecanismos de autenticación de múltiples factores que no solo incluyan la tarjeta inteligente, sino también la verificación biométrica o códigos de seguridad adicionales.

B. Estándares y Requisitos de Seguridad en los Sistemas de Control de Acceso

La protección de los datos de los usuarios y la integridad del sistema es crucial en los sistemas de control de acceso a vehículos. En este contexto, diversas organizaciones, como la International Organization for Standardization (ISO) y el National Institute of Standards and Technology (NIST), han establecido normas y directrices específicas para la implementación de estos sistemas. Los vehículos Tesla, al igual que otros automóviles de alta gama, deben cumplir con estos estándares para garantizar la seguridad y la privacidad de los datos de los usuarios.

Por ejemplo, uno de los principales estándares aplicados en el diseño de estos sistemas es la ISO/IEC 7816, que establece los requisitos de las tarjetas inteligentes para asegurar su compatibilidad y seguridad en el proceso de autenticación. Además, se implementan métodos de encriptación de datos, como AES (Advanced Encryption Standard), para proteger la transmisión de información durante el proceso de autenticación y evitar posibles ataques de interceptación de datos.

El monitoreo de la autenticación de accesos y las transacciones del vehículo es igualmente importante. Los sistemas avanzados permiten registrar todas las actividades realizadas, facilitando la detección de accesos no autorizados o actividades sospechosas. Además, las tarjetas inteligentes utilizadas en estos sistemas están diseñadas para resistir ataques de clonación y falsificación, garantizando que cada tarjeta sea única y no pueda ser replicada fácilmente [**?**], [**?**].

Por último, el sistema de control de acceso debe cumplir con la legislación local y global sobre privacidad de datos y protección de la información personal. Esto incluye la implementación de políticas de manejo de datos que permitan a los usuarios controlar qué información personal se almacena y cómo se utiliza dentro del sistema.

III. Especificaciones Formales y VDM++

A. VDM++

La Vienna Development Method (VDM) es uno de los métodos formales más antiguos y consolidados orientados a modelos para el desarrollo de software y sistemas computacionales. Este enfoque emplea lenguajes con una base matemática rigurosa para describir y analizar modelos durante las etapas iniciales del diseño. Esto posibilita la evaluación anticipada de propiedades críticas del sistema antes de comprometerse con su implementación. A través del desarrollo y estudio de estos modelos, se pueden detectar aspectos ambiguos o incompletos en las especificaciones informales, lo que refuerza la confianza en que la implementación final cumplirá con propiedades esenciales como la seguridad y la fiabilidad.

B. Definición de Clases

En VDM++, las clases representan componentes clave que agrupan tanto datos como comportamientos. Se definen especificando atributos (también conocidos como variables de instancia) y métodos (que pueden ser funciones u operaciones). Además, cada clase puede contener invariantes, que son condiciones que deben cumplirse durante toda la existencia de la instancia de clase.

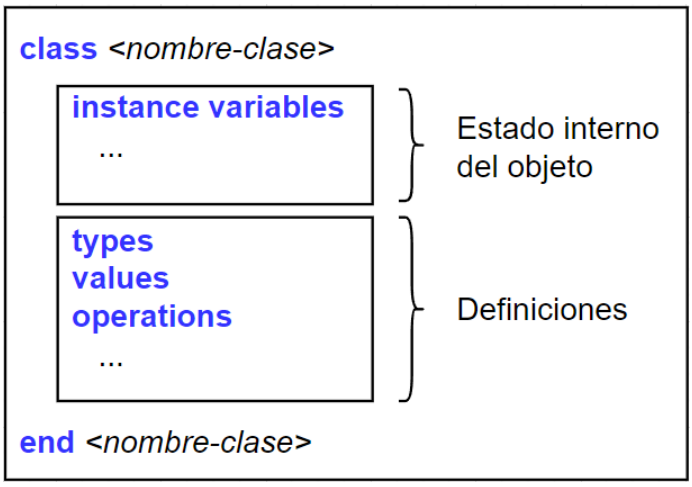

Fig. 1: Estructura básica de formalización de clases en VDM++

C. Tipos

VDM++ ofrece soporte para una extensa gama de tipos, incluyendo tanto tipos primitivos como compuestos. Entre los tipos básicos se encuentran los enteros, reales, booleanos y caracteres. Por otro lado, los tipos estructurados incluyen secuencias, conjuntos y tuplas. Asimismo, el lenguaje permite la definición de tipos personalizados, brindando mayor flexibilidad y capacidad de expresión para modelar sistemas complejos.

Operador	Semántica	Tipo
hd l	Primer elemento de la lista	seq1 of A → A
tl l	Cola de lista	seq1 of A → seq of A
len l	Largo de lista	seq of A → nat
inds l	Índices de lista	seq of A → set of A
l(i)	Elemento de la i-ésima posición	seq of A * nat1 → A
l1 = l2	Compara si dos listas son iguales	seq1 of A * seq of A → bool

Fig. 2: Operaciones Formales de tipo secuencia en VDM++

D. Expresiones

Las expresiones en VDM++ están formuladas de manera formal mediante operadores lógicos y matemáticos. Pueden clasificarse como aritméticas, booleanas o relacionales, y se utilizan para establecer condiciones del sistema, tales como precondiciones y postcondiciones de funciones y operaciones, así como para definir invariantes dentro de las clases.

IV. Propuesta

A. Requerimientos Funcionales del Sistema

Para la determinación de los requerimientos en el sistema de control de acceso a vehículos Tesla con Smart Cards, se deben considerar los siguientes:

- R1. Acceso Seguro: El sistema debe permitir la autenticación de los usuarios mediante tarjetas inteligentes MIFARE o llaves digitales, garantizando un acceso seguro al vehículo.
- R2. Gestión de Llaves: Debe ofrecer la posibilidad de gestionar hasta 19 llaves, de las cuales un máximo de 4 pueden ser tarjetas inteligentes.
- R3. Acceso Adicional: El sistema debe permitir la opción de acceso a través de la aplicación móvil y llaves Bluetooth en dispositivos compatibles.
- R4. Autenticación Inmediata: El sistema debe ser capaz de realizar la autenticación de usuarios de manera rápida, con un rango de lectura de hasta 10 cm para las tarjetas inteligentes.
- R5. Seguridad y Fiabilidad: Debe incluir mecanismos de seguridad avanzados, como el uso de especificaciones formales para garantizar su fiabilidad y evitar vulnerabilidades.

B. Diagrama de Clases

El diagrama de clases se mostrará a continuación.

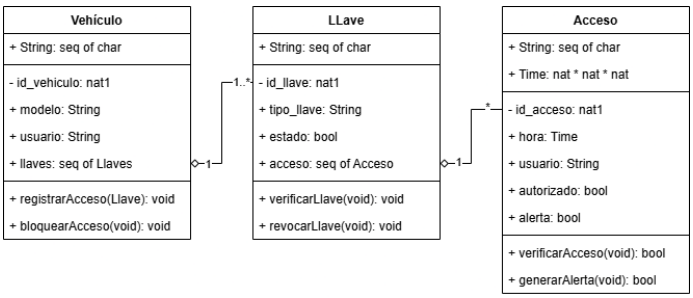

Fig. 3: Diagrama de Clases para el Sistema de Control de Acceso a Vehículos Tesla basado en Smart Cards.

C. Sistema de Control

El sistema de control de acceso de Tesla basado en Smart Cards se formaliza a través de tres entidades principales: `Vehiculo`, `Llave`, y `Acceso`. A continuación, se describe cada uno de estos elementos.

La clase `Vehiculo` representa los vehículos que utilizan el sistema de control de acceso. Los atributos de la clase son:

- **id_vehiculo**: Identificador único para cada vehículo.
- **modelo**: Modelo descriptivo del vehículo (por ejemplo, Model S, Model 3).
- **usuario**: Usuario propietario o autorizado para acceder al vehículo.
- **llaves**: Las llaves correspondientes al vehículo.

La clase `Llave` captura la información relacionada con las llaves utilizadas para el acceso al vehículo:

- **id_llave**: Identificador de la llave.
- **tipo_llave**: Tipo de llave (tarjeta inteligente, llave digital, Bluetooth).
- **estado**: Estado de la llave (autorizada, revocada, etc.).
- **acceso**: Los accesos relacionados con la tarjeta.

La clase `Acceso` registra los intentos de acceso y las autenticaciones realizadas:

- **id_acceso**: Identificador único del intento de acceso.
- **hora**: Hora exacta del intento de acceso.
- **usuario**: Usuario que intenta acceder al vehículo.
- **autenticado**: Indica si el acceso fue autorizado o no.
- **alerta**: Indica si el acceso fue denegado debido a un fallo en la autenticación o un intento de acceso no autorizado.

El sistema emitirá una alerta si se detecta un intento de acceso no autorizado o si la llave utilizada no está registrada o ha sido revocada. La función `verificar_acceso()` comprobará si la llave o el método de acceso es válido, y generará una señal de alerta en caso de fallo.

V. Especificación Formal

A. Clase Vehiculo

La clase **Vehiculo** representa un vehículo de un usuario, el cual tiene diferentes llaves de acceso, y cada llave, contiene diferentes registros de acceso.

- `id`: Un identificador único para el vehículo.
- `modelo`: El modelo del vehículo, representado como una secuencia de caracteres.
- `usuario`: El dueño del vehículo, representado como una secuencia de caracteres.positivo.
- `llaves`: Una secuencia de instancias de la clase llave. Debido a que, un usuario puede tener varias llaves de acceso.

Además, la clase contiene dos invariantes:

- La longitud del nombre debe ser mayor que cero.
- El atributo llaves, debe tener, por lo menos, una instancia de la clase Llave.

El código en VDM++ de la clase es el siguiente:

```
class Vehiculo

types
public String = seq of char

instance variables
private id_vehiculo: nat1;
public modelo: String;
private usuario: String;
public llaves: seq of Llave;
public intentos: nat;

operations
public Vehiculo: nat1 * String * String * seq
    of Llave ==> Vehiculo
Vehiculo(id, model, user, keys) == (
    id_vehiculo := id;
    modelo := model;
    usuario := user;
    llaves := keys;
);

public integrarLlave: Llave ==> ()
integrarLlave (key) == (
    llaves := llaves ^ [key];
);

public bloquearAcceso: nat1 * bool ==> ()
bloquearAcceso(idx, e) == (
  llaves(idx).setEstado(false);
);

public restringirLlave: () ==> ()
restringirLlave() == (
        intentos := 0;
    for i = 1 to len llaves do (
        for j = 1 to len llaves(i).accesos do
            (
            if llaves(i).verificarAcceso(j) =
                true then (
                intentos := intentos + 1;
            );

            if intentos = 3 then (
                bloquearAcceso(i, false);
                intentos := 0;
            );
        );
    );
);

end Vehiculo
```

B. Clase Llave

La clase **Llave** modela un aspecto de acceso al vehículo de un usuario, el cual puede tener muchas llaves de acceso.

- `id`: Un identificador único de la llave.
- `tipo`: Descripción adicional sobre la llave de acceso.
- `estado`: Determina si llave tiene permisos de acceso al vehículo.
- `accesos`: Una secuencia de accesos que determinar un seguimiento de accesos de la llave.

Los invariantes de la clase aseguran que:

- El atributo tipo, necesita tener un valor no nulo, por ello, su logitud debe ser mayor a 0.

El código en VDM++ de la clase es el siguiente:

```
class Llave

types
public String = seq of char

instance variables
private id_llave: nat1;
private tipo_llave: String;
public estado: bool;
public accesos: seq of Acceso;

operations
public Llave: nat1 * String * bool * seq of
    Acceso ==> Llave
Llave (id, type_key, flag, access) == (
    id_llave := id;
    tipo_llave := type_key;
    estado := flag;
    accesos := access;
);

public setEstado: bool ==> ()
setEstado(flag) == (
    estado := flag;
);

public verificarAcceso: nat1 ==> bool
verificarAcceso(idx) == (
    return accesos(idx).alerta;
);

public registrarAcceso: Acceso ==> ()
registrarAcceso (acc) == (
        accesos := accesos ^ [acc];
);

end Llave
```

C. *Clase Acceso*

La clase **Acceso** almacena la información sobre múltiples accesos de la llave, para determinar la cantidad de intentos que se han realizado de manera insegura, para controlar o bloquear las llaves.

- `id`: Un identificador único para los accesos.
- `hora`: Establece a qué hora se ha intentado realizar un acceso al vehículo y con qué llave.
- `usuario`: Una secuencia de caracteres para representar al usuario.
- `autorizado`: Un valor booleano que indica si el acceso ha sido autorizado.
- `alerta`: Retorna un booleano para determinar si el intento de acceso es fallido.

Los invariantes de la clase aseguran que:

- La hora en la que se realizó el acceso con una llave no sea nula.
- El usuario que realizó el acceso, no sea nulo.

El código en VDM++ de la clase es el siguiente:

```
class Acceso

types
public String = seq of char;
public Time = nat * nat * nat;

instance variables
private id_acceso: nat1;
public hora: Time;
public usuario: String;
public autorizado: bool;
public alerta: bool;

operations
public Acceso: nat1 * Time * String * bool *
    bool ==> Acceso
Acceso (id, time, user, auth, alert) == (
    id_acceso := id;
    hora := time;
    usuario := user;
    autorizado := auth;
    alerta := alert;
);

end Acceso
```

VI. Validación del Modelo

Para la realización de la validación se implementó la clase Validacion, donde se crearon los objetos necesarios de las clases desarrolladas. Además, se implementó los métodos para su creación y adecuado funcionamiento.

```
class Validacion

types
public String = seq of Char;

instance variables
public vehiculo: Vehiculo;
public llaves: seq of Llave;
public accesos1: seq of Acceso;
public accesos2: seq of Acceso;
public intentos: nat;

operations
public Validacion: () ==> ()
Validacion() == ();

public comprobacionPrueba1: () ==> ()
comprobacionPrueba1() == (

    intentos := 0;

    accesos1 := [
        new Acceso(1, mk_(23, 12, 50), "
            Usuario 1", true, false),
        new Acceso(2, mk_(22, 11, 20), "
            Usuario 1", true, false),
        new Acceso(3, mk_(10, 53, 30), "
            Usuario 1", true, false),
        new Acceso(4, mk_(9, 10, 33), "Usuario
            1", true, false),
        new Acceso(5, mk_(7, 24, 23), "Usuario
            1", true, false)
    ];
```

```
    accesos2 := [
        new Acceso(1, mk_(23, 12, 50), "
            Usuario_1", true, false),
        new Acceso(2, mk_(22, 11, 20), "
            Usuario_1", true, false),
        new Acceso(3, mk_(10, 53, 30), "
            Usuario_1", true, false),
        new Acceso(4, mk_(9, 10, 33), "Usuario
            _1", true, false),
        new Acceso(5, mk_(7, 24, 23), "Usuario
            _1", true, false)
    ];

        llaves := [
                new Llave(1, "tipo1", true,
                    []),
                new Llave(2, "tipo2", true,
                    [])
        ];

        for i = 1 to len accesos1 do (
                llaves(1).registrarAcceso(
                    accesos1(i));
        );

        for i = 1 to len accesos2 do (
                llaves(2).registrarAcceso(
                    accesos2(i));
        );

    vehiculo := new Vehiculo(1, "Modelo_1", "
        Usuario_1", []);

        for i = 1 to len llaves do (
                vehiculo.integrarLlave(llaves(
                    i));
        );

);

public comprobacionPrueba2: () ==> ()
comprobacionPrueba2() == (
    accesos1 := [
        new Acceso(1, mk_(23, 12, 50), "
            Usuario_1", true, true),
        new Acceso(2, mk_(22, 11, 20), "
            Usuario_1", true, true),
        new Acceso(3, mk_(10, 53, 30), "
            Usuario_1", true, false),
        new Acceso(4, mk_(9, 10, 33), "Usuario
            _1", true, true),
        new Acceso(5, mk_(7, 24, 23), "Usuario
            _1", true, true)
    ];

    accesos2 := [
        new Acceso(1, mk_(23, 12, 50), "
            Usuario_1", true, true),
        new Acceso(2, mk_(22, 11, 20), "
            Usuario_1", true, false),
        new Acceso(3, mk_(10, 53, 30), "
            Usuario_1", true, true),
        new Acceso(4, mk_(9, 10, 33), "Usuario
            _1", true, false),
        new Acceso(5, mk_(7, 24, 23), "Usuario
            _1", true, true)
    ];


        llaves := [
                new Llave(1, "tipo1", true,
                    []),
                new Llave(2, "tipo2", true,
                    [])
        ];

        for i = 1 to len accesos1 do (
                llaves(1).registrarAcceso(
                    accesos1(i));
        );

        for i = 1 to len accesos2 do (
                llaves(2).registrarAcceso(
                    accesos2(i));
        );

    vehiculo := new Vehiculo(1, "Modelo_1", "
        Usuario_1", []);

        for i = 1 to len llaves do (
                vehiculo.integrarLlave(llaves(
                    i));
        );
);

public verificacion_restriccion_llave: () ==>
    ()
verificacion_restriccion_llave() == (
        vehiculo.restringirLlave();
);

end Validacion
```

A continuación, se presenta el uso de la clase Validacion, para la ejecución de las pruebas establecidas de funcionamiento del Sistema de Control de Acceso.

```
>> create analisis := new Validacion()
>> print { analisis.comprobacionPrueba1() }
{ nil }
>> print { analisis.comprobacionPrueba2() }
{ nil }
>> print { analisis.verificacion_restriccion_llave() }
{ nil }
```

Fig. 4: Realizando Pruebas con la clase Analisis.

Por consiguiente, se presenta la construcción del Coverage de las clases de nuestro Sistema, la cual consta de un 100%. Para ello, se muestra la primera parte de los resultados del Coverage.

```
>> tcov write test.tc
>> rtinfo test.tc
  100%      4  Llave`Llave
  100%      2  Llave`setEstado
  100%     20  Llave`registrarAcceso
  100%     10  Llave`verificarAcceso
  100%  Llave
  100%    150  Acceso`Acceso
```

Fig. 5: Realizando Analisis de Coverage, Parte 1.

Finalmente, se muestra la segunda parte de los resultados del Coverage del Sistema.

```
  100%  Acceso
  100%      2  Vehiculo`Vehiculo
  100%      4  Vehiculo`integrarLlave
  100%      2  Vehiculo`bloquearAcceso
  100%      1  Vehiculo`restringirLlave
  100%  Vehiculo
  100%      4  Validacion`Validacion
  100%      2  Validacion`comprobacionPrueba1
  100%      2  Validacion`comprobacionPrueba2
  100%      2  Validacion`verificacion_restriccion_llave
  100%  Validacion

Total Coverage: 100%
```

Fig. 6: Realizando Analisis de Coverage, Parte 2.

VII. Model Checking

Para nuestro sistema de control de acceso, se define un modelo que representa el comportamiento de un mecanismo que permite o deniega el acceso en función de intentos, autenticación y condiciones de bloqueo.

A.

Variables del Modelo

- **Estado**: Puede tomar uno de los siguientes valores:
  - `Bloqueado`
  - `VerificandoAcceso`
  - `AccesoPermitido`
  - `AccesoDenegado`
  - `AlertaActivada`
- **intentos**: número de intentos realizados.
- **llave_valida**: booleano que indica si la llave es válida.
- **autenticacion_exitosa**: booleano que representa una autenticación exitosa.
- **bloquear**: booleano que indica si se debe bloquear el sistema.

B.

Transiciones del Modelo

- Desde `Bloqueado`:
  - Si `llave_valida`, transita a `VerificandoAcceso`.

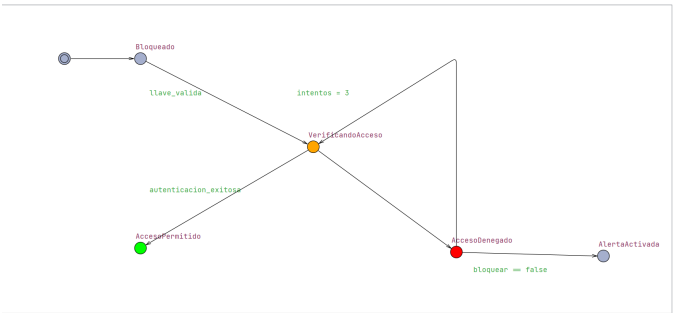

Fig. 7: Primer Diagrama de Transiciones

- Desde `VerificandoAcceso`:
  - Si `autenticacion_exitosa`, va a `AccesoPermitido`.
  - Si `intentos = 3`, va a `AccesoDenegado`.
- Desde `AccesoDenegado`:
  - Si `bloquear == false`, va a `AlertaActivada`.

C.

Especificaciones del Modelo

- **CTL:**
  - $AG(\texttt{intentos} \leq 3)$: *Siempreelnúmerodeintentosnodebesuperar*3.$EF(\texttt{Estado} = \texttt{AccesoPermitido})$: *Eventualmentesepuedellegarapermitirelacceso.*
  - $AG(\texttt{Estado} = \texttt{Bloqueado} \rightarrow AX(\texttt{llave\_valida} \rightarrow \texttt{Estado} = \texttt{VerificandoAcceso}))$: *Sielsistemaestábloqueadoylallaveesválida, puedeiraverificació*
- **LTL:**
  - $G(\texttt{Estado} = \texttt{VerificandoAcceso} \wedge \texttt{intentos} = 3 \rightarrow X(\texttt{Estado} = \texttt{AccesoDenegado}))$
  - $F(\texttt{Estado} = \texttt{AlertaActivada})$: En algún momento, puede activarse la alerta si el acceso es denegado y no se bloquea.

D.

Descripción del Modelo

El modelo representa un sistema de autenticación que verifica accesos en base a una llave válida, controla el número de intentos y define comportamientos como acceso permitido, acceso denegado y alerta activada. Asegura que no se realicen más de 3 intentos y que los estados de alerta se activen bajo condiciones específicas.

VIII. Conclusión

Tesla utiliza tarjetas inteligentes MIFARE (ISO 14443 Tipo A) y llaves digitales vía Bluetooth o aplicaciones móviles para autenticar usuarios sin contacto. Esta combinación proporciona un acceso más seguro, rápido y flexible, eliminando la necesidad de llaves físicas tradicionales.

El uso de métodos formales como VDM++ y Redes de Petri garantiza fiabilidad. Para asegurar que el sistema sea confiable y libre de errores críticos, se emplean especificaciones formales. Estas permiten modelar, validar y verificar rigurosamente el comportamiento del sistema antes de su implementación, alineándose con normas como ISO 26262 para la seguridad funcional en automoción.

REFERENCES

[1] S. P. Kaluvuri and G. Sindre, "A survey of practical formal methods for security," *International Journal of Computer Applications*, vol. 183, no. 23, pp. 1–11, 2021, accedido el 20 de junio de 2025. [Online]. Available: https://www.researchgate.net/publication/354379432

[2] GitHub, "Tesla key card protocol reverse engineering," https://gist.github.com/underhood/5a981f7098db17c66d4b99b6da3f63dc, 2020, accedido el 20 de junio de 2025.

[3] Not a Tesla App, "How tesla key cards work and how to pair them," https://www.notateslaapp.com/software-updates/item/how-tesla-key-cards-work-and-how-to-pair-them, 2023, accedido el 20 de junio de 2025.

[4] NXP Semiconductors, "Mifare classic ev1 1k - product data sheet," https://www.nxp.com/docs/en/data-sheet/MF1S50YYX$_V$1.*pdf*, 2019, $accedidoel20dejuniode2025$.

[5] Wikipedia contributors, "Mifare," https://en.wikipedia.org/wiki/MIFARE, 2024, accedido el 20 de junio de 2025.

[6] ——, "Mifare4mobile," https://en.wikipedia.org/wiki/MIFARE4Mobile, 2024, accedido el 20 de junio de 2025.

[7] eBay, "Nfc smart ring for tesla model 3/y," https://www.ebay.com/itm/264691129906, 2023, accedido el 20 de junio de 2025.

[8] Nedap Identification Systems, "Vehicle access control solutions," https://www.nedapidentification.com/solutions/vehicle-access-control/, 2021, accedido el 20 de junio de 2025.

[9] ASSA ABLOY, "Rfid smart card mifare classic 1k," https://www.assaabloy.com/products/smart-cards/rfid/mifare-classic, 2022, accedido el 20 de junio de 2025.

[10] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 2000.

[11] S. P. Kaluvuri and G. Sindre, "A survey of practical formal methods for security," in *10th International Workshop on Security and Trust Management (STM)*, 2014. [Online]. Available: https://dl.acm.org/doi/10.1007/978-3-319-17040-4$_8$

[12] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[13] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, pp. 213–254, 2007.

[14] C. Gómez and A. Rubio, "Dynamic access control through petri net workflows," in *Annual Computer Security Applications Conference (ACSAC)*, 2013, pp. 101–110.

[15] M. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. World Scientific, 1999.

[16] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: Probabilistic symbolic model checker," in *12th International Conference on Computer Performance Evaluation*, 2002, pp. 200–204.

[17] G. Boström and H. Sandberg, "Formal methods in automotive systems: A review," KTH Royal Institute of Technology, Tech. Rep., 2015. [Online]. Available: https://www.kth.se

[18] E. Clarke, T. Henzinger, H. Veith, and R. Bloem, Eds., *Handbook of Model Checking*. Springer, 2018.

[19] International Organization for Standardization, "Iso/iec 7816-4:2011 - integrated circuit cards – organization, security and commands for interchange," https://www.iso.org/standard/54550.html, 2011, accedido el 20 de junio de 2025.

[20] National Institute of Standards and Technology, "Fips pub 197: Advanced encryption standard (aes)," https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, U.S. Department of Commerce, Tech. Rep., 2001, accedido el 20 de junio de 2025.