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

**Alumnos:** D. León, K. Haytara

**Universidad:** La Salle de Arequipa

# Índice


- Introducción
- Resumen
- Especificaciones formales y VDM++
- Diagrama de Clases
- Validación
- Máquina de Estados
- Conclusiones
- Referencias

# Resumen

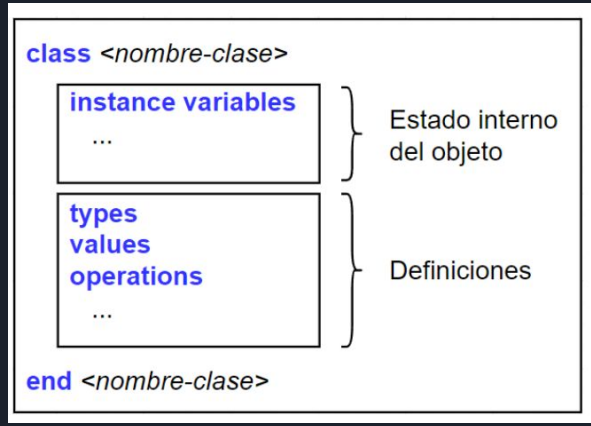
Resumen—La seguridad en el acceso a vehículos ha evolucionado con la adopción de tecnologías avanzadas, como las tarjetas inteligentes (smart cards). Tesla implementa un sistema basado en MIFARE, que opera bajo la norma ISO 14443 Tipo A a 13.56 MHz, permitiendo la autenticación del usuario mediante proximidad. 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.

# INTRODUCCIÓN

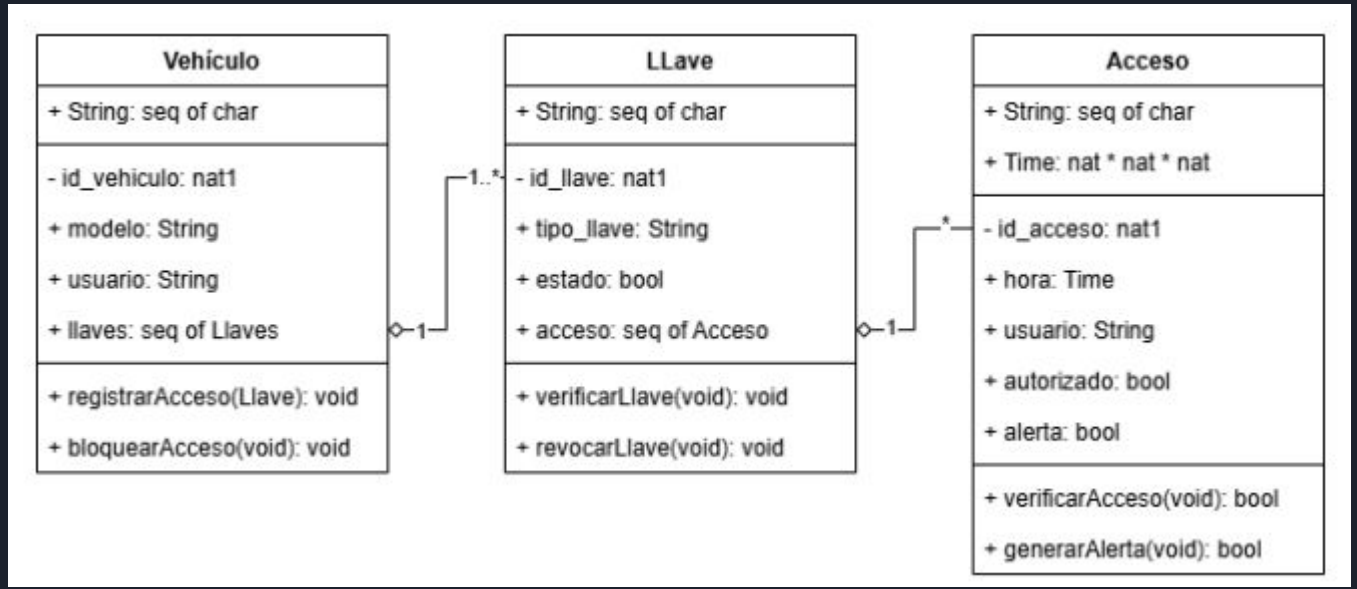
La seguridad y el control de acceso en vehículos han evolucionado con la adopción de tecnologías avanzadas, como las tarjetas inteligentes (smart cards). Tesla ha integrado esta tecnología en sus vehículos, permitiendo a los usuarios desbloquear y encender el automóvil mediante autenticación sin contacto.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

# Especificaciones formales y VDM++




| 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 |

# Diagrama de Clases



# Validación

```
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)
    ];
```

# Validación

```
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
```

# Validación

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

```
>> 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
```

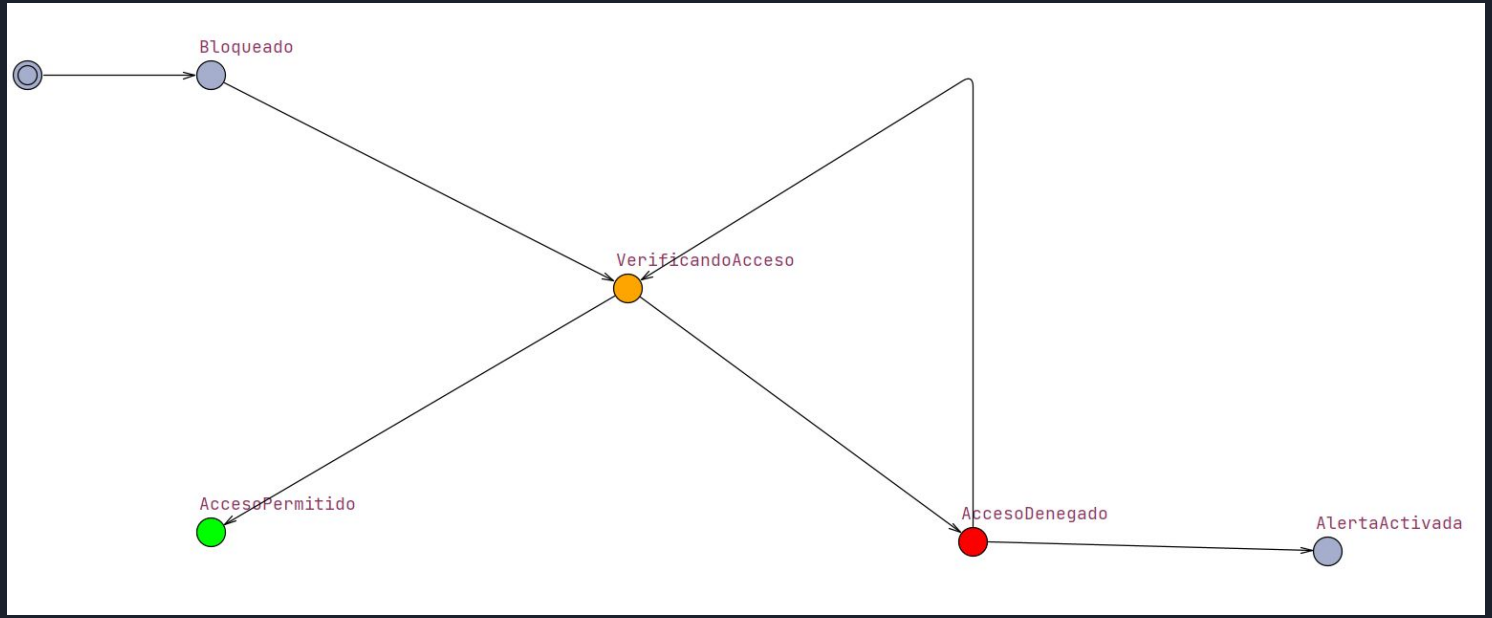
Fig. 5: Realizando Analisis de Coverage, Parte 1.

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%
```

# Máquina de Estados



# Conclusiones

- **Tesla mejora la seguridad del acceso con tecnología avanzada de smart cards**
  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.

- **El sistema responde a intentos de acceso no autorizados con mecanismos automáticos de restricción**
  El modelo propuesto incluye clases como Vehiculo, Llave y Acceso que controlan los intentos de autenticación. Si se detectan accesos fallidos repetidos, el sistema bloquea automáticamente la llave sospechosa, mejorando la protección frente a ataques o intentos de uso indebido.

# Referencias

1. Kaluvuri, S. P., & Sindre, G. (2021). A survey of practical formal methods for security. International Journal of Computer Applications, 183(23), 1–11.

https://doi.org/10.5120/ijca2021921530

2. GitHub. (2020). Tesla Key Card protocol reverse engineering. Recuperado de

https://gist.github.com/underhood/5a981f7098db17c66d4b99b6da3f63dc

3. Not a Tesla App. (2023). How Tesla key cards work and how to pair them. Recuperado de

https://www.notateslaapp.com/software-updates/item/how-tesla-key-cards-work-and-how-to- pair-them

4. NXP Semiconductors. (2019). MIFARE Classic EV1 1K - Product data sheet. Recuperado de

https://www.nxp.com/docs/en/data-sheet/MF1S50YYX_V1.pdf

5. Wikipedia. (2024). MIFARE. Recuperado de https://en.wikipedia.org/wiki/MIFARE

Advanced Encryption Standard (AES). Recuperado de

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

# Referencias


6. Wikipedia. (2024). MIFARE4Mobile. Recuperado de https://en.wikipedia.org/wiki/MIFARE4Mobile

7. eBay. (2023). NFC Smart Ring for Tesla Model 3/Y. Recuperado de https://www.ebay.com/itm/264691129906

8. Nedap Identification Systems. (2021). Vehicle access control solutions. Recuperado de https://www.nedapidentification.com/solutions/vehicle-access-control/

9. ASSA ABLOY. (2022). RFID Smart Card MIFARE Classic 1K. Recuperado de https://www.assaabloy.com/products/smart-cards/rfid/mifare-classic

10. Clarke, E. M., Grumberg, O., & Peled, D. A. (2000). Model checking. MIT Press.

# Referencias

11. Kaluvuri, S. P., & Sindre, G. (2014). A survey of practical formal methods for security. Proceedings of the 10th International Workshop on Security and Trust Management (STM). Recuperado de https://dl.acm.org/doi/10.1007/978-3-319-17040-4_8

12. De Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. Tools and Algorithms for the Construction and Analysis of Systems, 337–340.

https://doi.org/10.1007/978-3-540-78800-3_24

13. Jensen, K., Kristensen, L. M., & Wells, L. (2007). Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer, 9(3), 213–254. https://doi.org/10.1007/s10009-007-0038-x14. Gómez, C., & Rubio, A. (2013). Dynamic access control through Petri net workflows. In ACSAC '13: Annual Computer Security Applications Conference, 101–110.

https://doi.org/10.1145/2523649.2523665

15. Zhou, M. C., & Venkatesh, K. (1999). Modeling, simulation, and control of flexible manufacturing systems: A Petri net approach. World Scientific.

# Referencias

16. Kwiatkowska, M., Norman, G., & Parker, D. (2002). PRISM: Probabilistic symbolic model checker. Proceedings of the 12th International Conference on Computer Performance Evaluation, 200–204. https://doi.org/10.1007/3-540-46029-2_13

17. Boström, G., & Sandberg, H. (2015). Formal methods in automotive systems: A review. Technical report, KTH Royal Institute of Technology. Recuperado de https://www.kth.se

18. Clarke, E., Henzinger, T., Veith, H., & Bloem, R. (Eds.). (2018). Handbook of model checking. Springer. https://doi.org/10.1007/978-3-319-10575-8

19. ISO. (2011). ISO/IEC 7816-4:2011. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. International Organization for Standardization. https://www.iso.org/standard/54550.html

20. National Institute of Standards and Technology (NIST). (2001). FIPS PUB 197: