Revista Científica Multidisciplinar G-ner@ndo ISSN: 2806-5905

Análisis de Técnicas de Protección de Código (Ofuscación o Cifrado)

Code Protection Techniques Analysis (Obfuscation or Encryption)

David Loor Navia & Miguel Rodríguez Véliz

DIMENSIÓN CIENTÍFICA

Enero - junio, V°7 - N°1; 2026

Recibido: 20-02-2026
Aceptado: 23-02-2026
Publicado: 27-02-2026

PAIS

- Ecuador, Portoviejo
- Ecuador, Portoviejo

INSTITUCION

- Universidad Técnica de Manabí
- Universidad Técnica de Manabí

CORREO:

✉ dloor1460@utm.edu.ec
✉ miguel.rodriguez@utm.edu.ec

ORCID:

https://orcid.org/0009-0008-5211-5471
https://orcid.org/0000-0003-4474-3853

FORMATO DE CITA APA.

Loor, D. & Rodríguez, M. (2026). Análisis de Técnicas de Protección de Código (Ofuscación o Cifrado). *Revista G-ner@ndo, V°7 (N°1). Pág. 2277 – 2290.*

Resumen

La protección del código fuente es crucial durante el proceso de desarrollo de software, sin embargo, suele ser un aspecto que, por falta de conocimientos o de capacitación por parte de los desarrolladores, no se le da la importancia que requiere, especialmente en entornos donde el código es distribuido al usuario final o expuesto a amenazas como la ingeniería inversa, resultando en pérdidas importantes para cualquier organización o equipo de desarrollo. Por tal motivo, se decidió hacer un análisis de las técnicas de ofuscación y cifrado, con el fin de establecer criterios que faciliten la elección de la técnica según el contexto del software. La investigación se desarrolló bajo un enfoque cualitativo, mediante el análisis documental de artículos científicos, obteniendo como resultado, una matriz comparativa que permitió identificar las fortalezas y debilidades de cada técnica y, un marco de decisión como herramienta de apoyo a los desarrolladores, que facilita la elección de la técnica de protección de código en base a criterios como el nivel de protección, la resistencia a la ingeniería inversa o el impacto en el rendimiento del software.

Palabras clave: Protección de Código, Cifrado, Ofuscación, Ingeniería Inversa, Seguridad del Software.

Abstract

Source code protection is crucial during the software development process; however, it is often an aspect that, due to a lack of knowledge or training among developers, is not given the importance it requires, especially when the code is distributed to end users or exposed to threats such as reverse engineering, resulting in significant losses for any organization or development team, for this reason, an analysis of obfuscation and encryption techniques was conducted in order to establish criteria that facilitate the selection of the appropriate technique according to the software context. The research was carried out under a qualitative approach through documentary analysis of scientific articles, as a result, a comparative matrix was developed, allowing the identification of the strengths and weaknesses of each technique, along with a decision framework as a support tool for developers, this framework facilitates the selection of code protection techniques based on criteria such as the level of protection, resistance to reverse engineering, and impact on software performance.

Keywords: Code Protection, Encryption, Obfuscation, Reverse Engineering, Software Security.

Introducción

Los avances tecnológicos son cada vez más grandes y frecuentes, siendo beneficiosos para la humanidad en general, sin embargo, el uso inmoral e inadecuado de la tecnología, en muchas ocasiones son la causa principal de vulnerabilidades y amenazas que pueden afectar a cualquier tipo de Software, representando una problemática que nos incumbe a todos. Debido a esto, tanto las grandes compañías como los pequeños desarrolladores, se encuentran en una búsqueda constante por proteger su código frente a las nuevas amenazas. El problema real reside cuando se hace un uso inapropiado de técnicas de protección de código o directamente no se aplican estas técnicas. En su estudio, Braz y Bacchelli (2022) revelan que la falta de capacitación y conocimientos son los principales desafíos a los que se enfrentan los desarrolladores al detectar problemas de seguridad, poniendo en riesgo no solo al software como tal, sino también a todos los usuarios vinculados al mismo. Aquella falta de capacitación, ya sea por desconocimiento o dejadez, puede resultar en la filtración del código fuente de las aplicaciones de software, lo que traería consigo consecuencias irreparables para cualquier organización, tal y como lo mencionan Association of Banks in Singapore (ABS, 2021) la posesión no autorizada del código fuente genera un gran impacto sobre las entidades financieras, dejándolas expuestas frente a riesgos como la pérdida de propiedad intelectual, explotación de vulnerabilidades o mucho peor, resultando en el robo total del software.

Para contrarrestar esta problemática, es necesario aplicar el uso de técnicas de protección de código, y si bien existe una gran diversidad de ellas, como objeto de estudio se ha decidido analizar únicamente las técnicas de ofuscación y de cifrado, debido a que sus enfoques son complementarios en el campo de la seguridad en el desarrollo de software. El análisis se basa en el contexto en que ambas técnicas son aplicadas, mas no en su enfoque técnico, debido a que son técnicas fundamentalmente distintas y no pueden

ser comparadas técnicamente, sin embargo, sí es posible analizar contextos como el nivel de seguridad o el impacto en el rendimiento del software en cuestión, evidenciando de esta forma, los puntos fuertes y debilidades de cada una de ellas.

El cifrado hace posible el intercambio de mensajes de forma segura, siendo leídos únicamente por personas autorizadas, garantizando así, la confidencialidad del mensaje (Medina, 2017), hacer uso de ésta como técnica de protección de código, garantiza un alto nivel de seguridad, ya que implica el uso de algoritmos y claves de cifrado y descifrado, la cuales serán entregadas a las personas autorizadas. La criptografía se puede dividir en tres tipos:

Criptografía simétrica: usa una única clave que deben conocer el emisor y el receptor para cifrar y descifrar el mensaje, lo que es considerado un punto débil, ya que la clave puede ser interceptada si se envía mediante canales no seguros (Becerra, 2025).

Criptografía asimétrica: usa una clave pública que se pueden difundir sin problemas y una privada que es generada a partir de una contraseña, el inconveniente de este sistema es que toma un mayor tiempo de cifrado y descifrado de la información (Becerra, 2025).

Criptografía híbrida: combina las fortalezas de los dos sistemas anteriores, la clave de sesión es cifrada asimétricamente mientras que el mensaje es cifrado simétricamente, de esta forma contamos con mayor seguridad y eficiencia en el cifrado de la información (Becerra, 2025).

Por otro lado, la ofuscación es una técnica que transforma el código en otro menos entendible, dificultando el entendimiento del mismo, pero manteniendo la funcionalidad, aunque el código original haya sido modificado (Rodríguez et al., 2024), de este modo, es

mucho más complejo para un atacante lograr analizar un código ofuscado. Entre los diversos enfoques de ofuscación de código, Raitsis et al. (2025) destacan los siguientes:

Ofuscación léxica: se encarga de renombrar variables, clases y funciones para ocultar su significado, por ejemplo, una variable llamada "UserID" podría renombrarse como "Bar20zA10lo01", aumentando de esta forma la complejidad del análisis del código.

Ofuscación del flujo de control: introduce construcciones de flujo de control engañosas o reorganiza las rutas de ejecución, lo cual implica añadir código muerto para ocultar el verdadero flujo del sistema.

Ofuscación de datos: implica cifrar información confidencial dentro del código para hacerlos ilegibles, implicando que si un atacante accede al código no pueda interpretar fácilmente los datos protegidos.

Es necesario tener en consideración que ambas técnicas cuentan con sus propias limitaciones. En cuanto al cifrado, AlRoubiei et al. (2020) presentan en su trabajo un análisis de algoritmos simétricos y asimétricos donde evidencian el coste computacional en la aplicación de algoritmos de cifrado, siendo esta su principal desventaja. Por otro lado, Nuñez et al. (2015) mencionan que la ofuscación no trata de esconder el código, sino de hacerlo incompresible para los humanos durante un periodo de tiempo “razonable”, es decir que, un atacante con esfuerzo, determinación y mucho tiempo libre, lograría entender la estructura de una aplicación ofuscada. Aplicar esta técnica podría no ofrecer un nivel de seguridad tan alto, pero lo compensa con un bajo impacto en el rendimiento de la aplicación.

En la mayoría de estudios se destaca la utilización de la ofuscación y el cifrado como técnicas de protección, cada una aplicada en distintos contextos, por ejemplo, Rodríguez et al. (2024) en su investigación, presentan un modelo para ofuscar el grafo de llamadas

como método de protección, obteniendo resultados positivos en su implementación, mientras que Bernal et al. (2023) mencionan dos métodos de cifrado para la protección de bases de datos, el cifrado simétrico y el cifrado asimétrico, también conocido como “llave pública” y “llave privada” respectivamente. Al igual que en los casos previamente mencionados, en la mayoría de estudios tanto el cifrado como la ofuscación son analizados y aplicados de manera aislada, sin proporcionar criterios que orienten a los desarrolladores a escoger una técnica u otra. Por tal motivo, realizar un análisis de ambas técnicas permite abordar varios campos de la protección del código desde dos perspectivas opuestas, pero que se complementan entre sí, de esta forma y a través de este estudio se busca ofrecer una ayuda en la toma de decisiones a aquellos desarrolladores o equipos de desarrolladores que lanzan sus aplicaciones sin protección previa al código, permitiendo reconocer a través de la lectura de la presente investigación, cuál de las dos técnicas es conveniente de acuerdo al contexto de su software.

El presente estudio está organizado de tal forma que se describe la metodología utilizada en la sección de materiales y métodos, para posteriormente, presentar los criterios de selección de la técnica de protección de código y analizar los resultados de la comparación entre la ofuscación y el cifrado.

Métodos y Materiales

El enfoque de la investigación es cualitativo, ya que “se trata de comprender el conjunto de cualidades interrelacionadas que caracterizan a un determinado fenómeno para construir un conocimiento” (Mejía Navarrete, 2014, p.2). De tal modo, esta investigación se centra en el análisis y en la comparación de la ofuscación y el cifrado como técnicas de protección de código, sin llevar a cabo experimentos que requieran análisis numérico.

"La investigación exploratoria es aquella que se efectúa sobre un tema u objeto desconocido o poco estudiado, por lo que sus resultados constituyen una visión aproximada de dicho objeto, es decir, un nivel superficial de conocimientos" (Arias-Odón, 2006, p.23). Si bien existen estudios sobre ofuscación y cifrado, se analizan de forma aislada la una de la otra, por lo que no se cuenta con criterios claros que permitan compararlas, siendo esta investigación una forma de facilitar la elección de la técnica de protección de código a través de la comprensión del contexto del software.

Métodos de investigación

- Método Analítico: "es aquel método de investigación que consiste en la desmembración de un todo, descomponiendo sus partes o elementos para observar las causas, la naturaleza y los efectos" (Ruiz, 2006a), mediante su utilización se permitirá descomponer el objeto de estudio —la protección del código— en elementos principales, como la comparación entre la ofuscación y el cifrado, identificando sus causas, características y efectos de cada técnica de manera individual.

- Método Sintético: trabaja en conjunto con el método analítico, "el método sintético es un proceso de razonamiento que tiende a reconstruir un todo, a partir de los elementos distinguidos por el análisis" (Ruiz, 2006b). Permite integrar los resultados obtenidos en el análisis de cada técnica, reconstruyendo una visión global en base a los criterios analizados y facilita establecer conclusiones generales.

Como técnica de investigación se empleó un análisis documental de artículos científicos y documentos académicos relacionados con las técnicas de ofuscación y cifrado, con el objetivo de obtener fundamentos teóricos para la comparación de las técnicas.

Análisis de resultados

En base al análisis de ambas técnicas, se definen los criterios de elección que permiten comparar las técnicas de ofuscación y cifrado para dirimir entre una u otra dependiendo del contexto en que se apliquen.

Tabla 1. *Definición de criterios de selección de la técnica de protección de código*

Criterio	Definición
Nivel de protección de código	Evalúa el grado de dificultad que la técnica impone a terceros para comprender, leer o modificar el código.
Resistencia a la ingeniería inversa	La ingeniería inversa es un proceso que da a entender cómo está construido un programa, en manos equivocadas suele ser usada para encontrar vulnerabilidades en aplicaciones de software (Votipka et al., 2019), este criterio evalúa la resistencia de la técnica ante este tipo de amenazas.
Dificultad de implementación	Se refiere al esfuerzo necesario para implementar la técnica de protección, considerando el conocimiento avanzado y la integración en el proceso de desarrollo de software.
Impacto en el rendimiento del software	Evalúa el efecto de aplicar la técnica de protección sobre el desempeño de un determinado software, considerando el coste computacional requerido.
Compatibilidad con tipos de software	Describe en qué contextos de desarrollo de software se aplican ambas técnicas.

Nota. Los criterios fueron seleccionados teniendo en consideración las fortalezas y debilidades de ambos enfoques

Una vez definidos los criterios, es momento de proceder con las respectivas comparaciones entre la ofuscación y el cifrado, facilitando la discusión posterior y la elección de la técnica, para ello se ha elaborado una matriz comparativa evaluada bajo una escala cualitativa, donde se analizan ambas técnicas en base a fundamentos como las características intrínsecas de cada técnica, su comportamiento en ejecución y el contexto en que son aplicadas.

Tabla 2. *Matriz comparativa.*

Criterio	Justificación	
Nivel de protección de código	**Ofuscación: Medio**	**Cifrado: Alto**
	La ofuscación logra dificultar la compresión del código para los atacantes, pero su protección no es absoluta ya que, con el tiempo es posible llegar a comprenderlo. Incluso, Schrittwieser y Katzenbeisser (2011) en su estudio mencionan que la ofuscación es mucho más vulnerable mientras el software está en ejecución, para revertir esto, proponen el concepto de diversificación del flujo de control. En contraparte, el cifrado sí ofrece un alto nivel de protección, pero hay que tener en consideración, que se debe cifrar únicamente las partes críticas o sensibles del código y descifrarlas en memoria durante la ejecución, Morel et al. (2023) en su estudio, descifra instrucciones dentro del CPU, lo que complica a un atacante extraer información sensible.	
Resistencia a la ingeniería inversa	**Ofuscación: Media**	**Cifrado: Alta**
	Una ofuscación medianamente bien implementada debe estar acompañada de otras técnicas como anti-debugging, que en caso de detectar síntomas de ingeniería inversa sobre el software, termina el programa o ejecuta alguna otra instrucción como el cambio del flujo o la inserción de código muerto, también, se recomienda el uso de anti-tampering para detectar si el código ha sido modificado (Illuru y Anthoniraj, 2025), de esta forma, la ofuscación puede hacer frente a ataques de ingeniería inversa, pero es importante aclarar, que solo la dificulta, mas no la imposibilita. En cuanto al cifrado, la ingeniería inversa no busca descifrarlo porque es prácticamente imposible si se hace uso de algoritmos como Advanced Encryption Standard (AES), en su lugar, la ingeniería inversa intenta encontrar problemas en la implementación como la clave de cifrado sin protección dentro del código, la forma en que se genera o si es posible interceptarla, por lo tanto, la resistencia es alta si se hace uso de algoritmos correctos de cifrado y si la implementación es adecuada.	
Dificultad de implementación	**Ofuscación: Media**	**Cifrado: Alta**
	Para Blake (2025) el cifrado presenta un mayor grado de dificultad ya que requiere de conocimientos avanzados de llaves, algoritmos y sobre todo un entorno de ejecución	

	seguro para descifrar el mensaje, mientras que la ofuscación, suele ser implementada mediante el uso de herramientas automatizadas, sin necesidad de hardware adicional.	
Impacto en el rendimiento del software	**Ofuscación: Bajo**	**Cifrado: Alto**
	Es un hecho que la implementación de técnicas de ofuscación tienen un impacto mucho menor en el rendimiento del software. La ofuscación es un proceso que no introduce cálculos complejos durante la ejecución del software, lo que implica un impacto bajo en el rendimiento del mismo (Collberg et al., 1997), mientras que el uso de técnicas de cifrado demandan un uso de recursos mucho mayor en comparación a la técnica de ofuscación. En su estudio, Singh (2022) demuestra que el uso de técnicas de cifrado incrementan el uso del CPU, afectando directamente en el rendimiento de las aplicaciones que implementan estas técnicas.	
Compatibilidad con tipos de software	**Ofuscación: Alta**	**Cifrado: Media**
	La ofuscación es utilizada en aplicaciones de escritorio, aplicaciones móviles, o aplicaciones web donde el código es distribuido al usuario final y podría ser expuesto. Por su parte, el cifrado, es aplicado sobre todo en sistemas que manejan información sensible, como aplicaciones financieras, donde la prioridad es proteger datos o componentes críticos en lugar de todo el código.	

Fuente. Elaboración propia

Realizado el análisis a ambas técnicas, es posible establecer un conjunto de reglas de decisión que ayuden en la elección de la técnica de protección de código de acuerdo al contexto del software en cuestión.

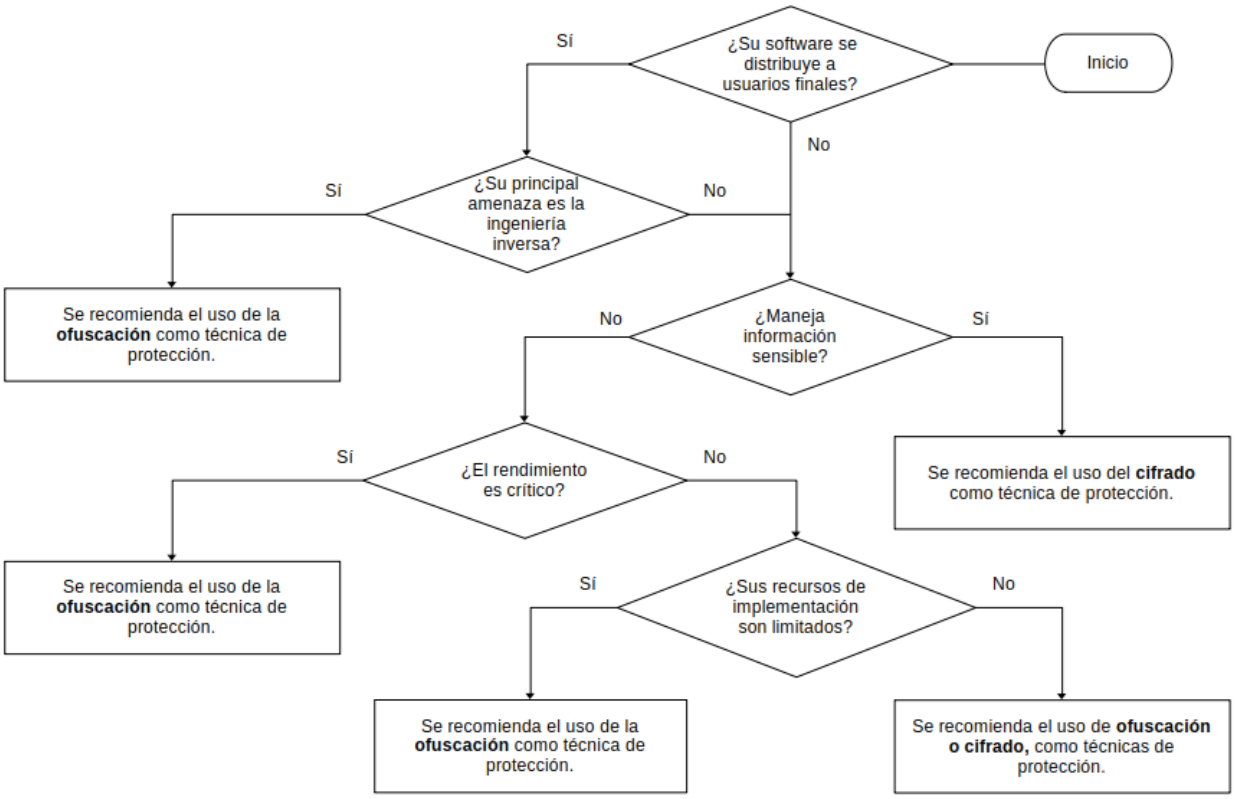
Tabla 3. *Reglas de decisión.*

Reglas	Descripción
El software se distribuye a usuarios finales	Cuando el software es distribuido a usuarios finales, el código es propenso a sufrir ataques de ingeniería inversa.
La principal amenaza del software es la ingeniería inversa	Se recomienda el uso de técnicas de ofuscación por su capacidad de dificultad la comprensión del software, de no ser el caso, es necesario evaluar otras posibilidades.
El software maneja información sensible	En caso de ser cierto, se recomienda el uso de técnicas de cifrado por sobre otras opciones, debido a su alto nivel de confidencialidad.
El rendimiento es un factor crítico en el software	Si el rendimiento es crítico se recomienda priorizar la ofuscación debido a que su impacto en ejecución es mucho menor en comparación con el cifrado.
Recursos de implementación limitados.	Si no se cuenta con los conocimientos ni los recursos necesarios para implementar una técnica de cifrado, se recomienda el uso de técnicas de ofuscación, caso contrario, es posible implementar ambos enfoques.

Fuente. Elaboración propia

La Figura 1 denota el flujo de decisión recomendado que sintetiza las reglas propuestas, se aconseja seguir este flujo para llegar a una decisión final.

Figura 1. *Flujo de decisión para la selección de técnicas de protección de código*



Nota. El flujo es de carácter ilustrativo con el fin de complementar las reglas definidas.

Conclusiones

Se logró cumplir el objetivo de analizar las técnicas de cifrado y ofuscación como técnicas de protección de código, concluyendo que el cifrado garantiza la confidencialidad del código y la información sensible, a costo de altos niveles de implementación y recursos computacionales, mientras que la ofuscación se encarga de dificultar la comprensión del código fuente, reduciendo el riesgo de ingeniería inversa a bajo costo computacional y de recursos. Es importante mencionar que no existe una técnica superior a la otra, ya que dependen del contexto en el que se apliquen, como el tipo de aplicación, los recursos disponibles, el nivel de seguridad, entre otros. Gracias a la investigación se logró desarrollar una matriz comparativa donde se definieron criterios que permiten la elección entre cifrado y ofuscación, en base a dicha matriz se construyó un marco de selección como herramienta

de apoyo para desarrolladores y equipos de software que no cuentan con conocimientos avanzados en seguridad informática, a los cuales se les facilitar dirimir entre una técnica u otra. Se reconoce como limitación del estudio la falta de validaciones a través de la práctica o estudios de caso, sin embargo, los resultados obtenidos cumplen con lo establecido y el estudio es sustentado por la evidencia científica.

Se recomienda a los desarrolladores de software utilizar el marco de decisión propuesto como guía para seleccionar técnicas de protección de código, considerando el contexto de su software, considerando que, en contextos donde el software maneje información sensible o crítica, se sugiere priorizar el uso de técnicas de cifrado, mientras que, en aplicaciones distribuidas al usuario final donde el código fuente está expuesto a ataques de ingeniería inversa, se recomienda priorizar técnicas de ofuscación.

Finalmente, se aconseja evaluar la posibilidad de combinar las técnicas de cifrado y ofuscación como estrategia de seguridad extra, especialmente en aplicaciones comerciales y para futuras investigaciones, realizar estudios que validen el marco de decisión mediante pruebas prácticas, que validen los resultados obtenidos en este trabajo.

Referencias bibliográficas

AlRoubiei, M., AlYarubi, T., & Kumar, B. (2020). Critical Analysis of Cryptographic Algorithms. 2020 8th International Symposium on Digital Forensics and Security (ISDFS), 1-7. https://doi.org/10.1109/ISDFS49300.2020.9116213

Arias-Odón, F. (2006). Fidias G. Arias—El Proyecto de la Investigación.

Association of Banks in Singapore (ABS). (2021). Study on Risk and Impact of Source Code Leakage from Software Vendors [White Paper]. Association of Banks in Singapore. https://abs.org.sg/docs/library/study-on-risk-and-impact-of-source-code-leakage-from-software-vendors_final.pdf

Becerra, L. (2025, junio). La criptografía como pilar de la seguridad informática. ResearchGate. https://www.researchgate.net/publication/392715089_LA_CRIPTOGRAFIA_COMO_PILAR_DE_LA_SEGURIDAD_INFORMATICA

Bernal, J., Zorrilla, F., Palacios, M., Rosales, N., & Farías, V. (2023, diciembre 31). Implementación de técnicas de encriptación en la seguridad de las bases de datos. IPSUMTEC, Vol. 6 Núm 7 (2023). https://revistas.milpaalta.tecnm.mx/index.php/IPSUMTEC/article/view/247

Blake, H. (2025). Comparative Study of Obfuscation vs. Encryption in Firmware Protection. https://www.researchgate.net/publication/393622775_Comparative_Study_of_Obfuscation_vs_Encryption_in_Firmware_Protection

Braz, L., & Bacchelli, A. (2022). Software security during modern code review: The developer's perspective. Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '22, 810-821. https://doi.org/10.1145/3540250.3549135

Collberg, C., Thomborson, C., & Low, D. (1997). A Taxonomy of Obfuscating Transformations. http://www.cs.auckland.ac.nz/staff-cgi-bin/mjd/csTRcgi.pl?serial.

Illuru, S., & Anthoniraj, S. (2025). A Study on Anti-Debugging and Anti-Tampering Techniques in Applications. International Journal of Innovative Research in Technology, 11(11), 1783-1788.

Medina, L. (2017). Criptografía y mecanismos de seguridad. Areandina. https://digitk.areandina.edu.co/entities/publication/06567e93-11e1-4657-99d9-bc880f0b05a3

Mejía Navarrete, J. (2014). Sobre la investigación cualitativa. Nuevos conceptos y campos de desarrollo. Investigaciones Sociales, 8(13), 277-299. https://doi.org/10.15381/is.v8i13.6928

Morel, L., Couroussé, D., & Hiscock, T. (2023). Code Polymorphism Meets Code Encryption: Confidentiality and Side-Channel Protection of Software

Components. Digital Threats: Research and Practice, 4(2), 1-27. https://doi.org/10.1145/3487058

Nuñez, Y., Bolívar, M., Díaz, H., & Sepúlveda, R. (2015, febrero 15). Arquitectura extensible para la protección automatizada de software: Un caso de estudio. Revista Cubana de Ciencias Informáticas, Vol. 9. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992015000500001&lang=es

Raitsis, T., Elgazari, Y., & Lurie, Y. (2025, enero). Code Obfuscation: A Comprehensive Approach to Detection, Classification, and Ethical Challenges. ResearchGate. https://www.researchgate.net/publication/388271314_Code_Obfuscation_A_Comprehensive_Approach_to_Detection_Classification_and_Ethical_Challenges

Rodríguez, M., Hernández, A., Supúlveda, R., & Núñez, Y. (2024, agosto). Validación de un modelo de protección basado en la ofuscación de código. SciELO. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59362024000200091&lang=es

Ruiz, R. (2006a). Historia de la ciencia y el método científico. Eumed. www.eumed.net/libros/2007b/283/

Ruiz, R. (2006b). Historia y Evolución del Pensamiento Científico. Eumed. https://www.eumed.net/libros-gratis/2007a/257/7.2.htm

Schrittwieser, S., & Katzenbeisser, S. (2011). Code Obfuscation against Static and Dynamic Reverse Engineering. En T. Filler, T. Pevný, S. Craver, & A. Ker (Eds.), Information Hiding (Vol. 6958, pp. 270-284). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24178-9_19

Singh, A. (2022). Testing the Impact of Encryption on Network Performance. INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, 06, 1-7. https://doi.org/10.55041/IJSREM16483

Votipka, D., Rabin, S., Micinski, K., & Foster, J. (2019, abril). An Observational Investigation of Reverse Engineers' Process and Mental Models. ResearchGate. https://www.researchgate.net/publication/332773835_An_Observational_Investigation_of_Reverse_Engineers'_Process_and_Mental_Models