© Hrytsak A.V., Bezpalyi K.V., Prysiazhnyi D.P., Lukanov M.V. Multi-level Secure Software Source Code Version Control System Based on Asymmetric Cryptographic Mechanisms // *Ukrainian Scientific Journal of Information Security, 2025, vol. 31, issue 3, pp. 175-181.*

DOI: 10.18372/2225-5036.31.21164

БАГАТОРІВНЕВА ЗАХИЩЕНА СИСТЕМА КОНТРОЛЮ ВЕРСІЙ ВИХІДНОГО КОДУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АСИМЕТРИЧНИХ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ

Анатолій Грицак, Кирило Безпалий, Дмитро Присяжний, Максим Луканов

Вінницький національний технічний університет, м. Вінниця, Україна

ГРИЦАК Анатолій Васильович, к.т.н.
Рік та місце народження: 1993 рік, м. Вінниця, Україна.
Освіта: Вінницький національний технічний університет.
Посада: доцент кафедри менеджменту та безпеки інформаційних систем Вінницького національного технічного університету.
Наукові інтереси: інформаційна безпека, криптографія, хмарні та блокчейн технології, захист інфраструктури Інтернету речей.
Публікації: близько 40 наукових публікацій.
E-mail: grytsak.a.v@gmail.com
ORCID: 0000-0002-0776-9889

БЕЗПАЛИЙ Кирило Валерійович
Рік та місце народження: 1991 рік, м. Вінниця, Україна.
Освіта: Вінницький національний технічний університет.
Посада: асистент кафедри менеджменту та безпеки інформаційних систем Вінницького національного технічного університету.
Наукові інтереси: безпека інформаційних систем, криптографія, хмарні та блокчейн технології.
Публікації: близько 20 наукових публікацій.
E-mail: kyrylo.bezpalyi@vntu.edu.ua
ORCID: 0009-0008-0331-9312

ПРИСЯЖНИЙ Дмитро Петрович
Рік та місце народження: 1988 рік, м. Вінниця, Україна.
Освіта: Вінницький національний технічний університет.
Посада: асистент кафедри менеджменту та безпеки інформаційних систем Вінницького національного технічного університету.
Наукові інтереси: безпека інформаційних систем, криптографія, хмарні та блокчейн технології.
Публікації: близько 25 наукових публікацій.
E-mail: dimpris@gmail.com
ORCID: 0009-0000-8327-3183

ЛУКАНОВ Максим Всеволодович
Рік та місце народження: 2001 рік, м. Донецьк, Україна.
Освіта: Вінницький національний технічний університет.
Посада: фахівець з інформаційної безпеки компанії KNESS Group.
Наукові інтереси: інформаційна безпека, криптографія, хмарні та блокчейн технології, безпека критичних інфраструктур.
Публікації: 3 наукових публікації.
E-mail: max.luk.2001@gmail.com
ORCID: 0009-0004-7506-9501

Анотація. *У даній статті описано багаторівневу захищену систему контролю версій вихідного коду програмного забезпечення, що базується на використанні асиметричних криптографічних механізмів. Основну увагу приділено розробці та обґрунтуванню методу, який передбачає застосування цифрового підпису для забезпечення контролю цілісності, автентичності та доступності програмного коду, а також для виявлення несанкціонованих модифікацій. Запропонований підхід орієнтований на інтеграцію в існуючі системи контролю версій і передбачає реалізацію механізмів, що забезпечують підвищення рівня безпеки на всіх етапах життєвого циклу програмного забезпечення. Зокрема, доступ до внесення змін у код контролюється через обов'язкову процедуру перевірки цифрового підпису, що дозволяє обмежити можливість несанкціонованого втручання. Підписання кожного файлу забезпечує можливість виявлення будь-яких змін у коді, навіть у випадках часткової модифікації або додавання нових компонентів.*

Додатковою перевагою методу є те, що використання ключа підпису не надає прямого доступу до репозиторію, що зменшує ризики компрометації системи. Перед виконанням підпису всі зміни проходять перевірку уповноваженою особою, що дозволяє поєднати криптографічні механізми із процесами контролю якості програмного забезпечення. Важливим аспектом є також забезпечення безпечного використання приватного ключа, який не передається іншим користувачам і застосовується лише у визначених середовищах.

Для підтвердження ефективності запропонованого методу було розроблено програмний продукт, який реалізує основні етапи процесу підпису та верифікації програмного коду. Проведене експериментальне дослідження дозволило підтвердити доцільність використання даного підходу, зокрема його здатність виявляти несанкціоновані зміни, забезпечувати контроль автентичності та інтегруватися в автоматизовані процеси розробки програмного забезпечення. Отримані результати свідчать про практичну цінність запропонованого методу та можливість його застосування у сучасних системах розробки програмного забезпечення.

Ключові слова: *система контролю версій, вихідний код, цифровий підпис, цілісність даних, криптографічні механізми, ECDSA, SHA-256.*

Вступ. Розробка програмних рішень та їх активне впровадження у різні сфери діяльності зумовлюють зростання інтересу до створення, вдосконалення та пошуку ефективніших підходів у галузі програмної інженерії. Водночас потреба як у подальшому розвитку програмного забезпечення, так і в запобіганні несанкціонованому доступу чи викраденню даних, забезпеченні достовірності коду та захисті авторських прав визначає необхідність його комплексного захисту на всіх етапах життєвого циклу.

На сьогодні існує значна кількість методів захисту програмного коду, зокрема підходи, що базуються на обфускації, застосуванні штучного інтелекту та інших технологіях [1]. Проте важливим є не лише забезпечення захисту кінцевого продукту, але й організація контролю цілісності та автентичності коду на всіх етапах роботи з ним.

Стрімкий розвиток інформаційних технологій зумовлює застосування систем контролю версій (СКВ) в якості базового інструменту розробки, який забезпечує відстеження та контроль змін, паралельну роботу команди та прозорий аудит здійснених модифікацій. Такі системи дозволяють структурувати процес написання коду, уникати конфліктів при злитті гілок, а також із мінімальним часовими затратами відновлювати попередні версії проєкту.

Функціонально СКВ вирішують чотири ключові завдання [2]:

– організацію доступу до ресурсів;

– ведення журналу змін;

– керування розгалуженням розробки;

– підтримку контролю версій продукту.

Виконання даних завдань гарантує цілісність архітектури та високу продуктивність командної роботи.

Процес автентифікації користувачів в СКВ має ряд критичних загроз, які можуть скомпрометувати цілісність вихідного коду та безпеку всієї інфраструктури програмного продукту. Одним із значущих ризиків є викрадення даних для автентифікації, наприклад, паролю, токену, криптографічного ключа. Серед найбільш розповсюджених типів атак є фішинг, підбір паролів, перехоплення токенів доступу та загалом усі методи соціальної інженерії. Своєчасне виявлення та ідентифікація даних атак є необхідною умовою для розробки стійких механізмів захисту та верифікації користувачів.

Результати сучасних досліджень підкреслюють, що в СКВ автентифікація є критичним елементом безпеки в межах методології DevSecOps. Відповідно до рекомендацій NIST (SP 800-204), забезпечення цілісності коду вимагає впровадження багаторівневих стратегій автентифікації та авторизації, що охоплюють контроль доступу користувачів та автоматизовані взаємодії в мікросервісних архітектурах. Також значна увага приділяється захисту від несанкціонованого доступу та витоку конфіденційних даних у зв’язку з компрометацією облікового запису розробника у CI/CD-конвеєрах [3].

Окрім згаданих методів автентифікації, існує альтернативний підхід – концепція «безпека як код». Даний метод означає, що верифікація користувача здійснюється безперервно на всіх етапах життєвого циклу розробки. Аналіз останніх публікацій вказує на ефективність інтеграції засобів автоматизованого тестування безпеки (SAST, DAST, IAST) та технологій самозахисту застосунків (RASP). Даний підхід дає можливість ідентифікувати вразливості, пов'язані з механізмами автентифікації, у реальному часі, що забезпечує цифрову верифікацію кожної зміни та мінімізує ризики підміни вихідного коду [4].

Для забезпечення цілісності даних, згідно з результатами досліджень, перспективним напрямом є використання технології блокчейн та алгоритму Proof-of-Work. В роботі [5] зазначено, що поєднання криптографічних методів хешування та електронних цифрових підписів (ЕЦП) у децентралізованих архітектурах дозволяє фахівцям ефективно запобігати несанкціонованим модифікаціям даних та забезпечувати захист системи від атак «грубої сили» та соціальної інженерії. В СКВ, де цілісність та достовірність коду є важливою задачею, вказаний метод верифікації дій є актуальним способом аудиту всієї історії розробки.

Незважаючи на різноманітність існуючих підходів, зокрема стандарти NIST та сучасні інноваційні рішення, кожен із них має певні недоліки та обмеження при практичному впровадженні в динамічних середовищах розробки. Зокрема, варто відзначити труднощі в управлінні криптографічними ключами, потребу у високих обчислювальних ресурсах, складність інтеграції засобів захисту в автоматизовані процеси. Вказані недоліки досить часто змушують фахівців здійснювати вибір між безпекою та продуктивністю системи.

Враховуючи наведені недоліки існуючих підходів, актуальною задачею залишається необхідність у пошуку оптимального балансу та вдосконаленні механізмів автентифікації, що

дозволить забезпечити безперервну верифікацію, контроль цілісності та достовірності даних без надмірного ускладнення робочих процесів.

Постановка проблеми

Проведений аналіз сучасних засобів автентифікації свідчить, що традиційні методи, зокрема такі як паролі та стандартна двофакторна автентифікація досить часто є недостатніми для захисту систем контролю версій від складних кібератак. Ключовим недоліком залишається залежність безпеки від людського фактору (поведінки користувача), що потенційно може створювати для зловмисника можливості для фішингових атак, перехоплення сесій та великої кількості атак із застосуванням методів соціальної інженерії. Попри використання додаткових факторів захисту, існуючі механізми не завжди здатні гарантувати цілісність та достовірність програмного коду під час етапу його передачі та модифікації, залишаючи можливості для несанкціонованих змін.

Враховуючи наведені фактори, доцільним є подальша розробка та впровадження на практиці вдосконалених методів верифікації користувачів, зокрема на основі цифрових підписів. Використання криптографічного підписування змін дозволяє забезпечити автентичність коду та унеможливити його підробку, що створює надійний механізм контролю.

Метою роботи є розробка комплексного рішення, яке б поєднувало багаторівневу автентифікацію користувачів із механізмами суворого підтвердження цілісності даних, що забезпечить високий рівень довіри до системи та стійкість до актуальних кіберзагроз.

Основна частина дослідження

Опишемо запропонований авторами даної статті алгоритм автентифікації вихідного коду програмного забезпечення, що базується на застосуванні цифрового підпису файлів у системах контролю версій. Основна мета такого підходу – це забезпечення цілісності та підтвердження автентичності всіх компонентів проєкту, а також засвідчення авторства. Окрім того, розробка на основі розробленого підходу слугує інструментом перевірки безпечності та надійності коду, яку здійснюють відповідні фахівці. У межах запропонованого підходу цифровий підпис надається виключно тому коду, який пройшов перевірку та був підтверджений відповідними користувачами. Такий код буде вважатись придатним до подальшого використання: розгортання на серверах, розповсюдження серед користувачів.

Початковим етапом алгоритму є формування уповноваженими користувачами криптографічної пари ключів – відкритого та закритого. Особлива увага приділяється організації безпечного зберігання закритого ключа з урахуванням необхідних рівнів захисту, що забезпечують конфіденційність, цілісність і доступність ключової інформації відповідно до внутрішніх політик організації, вимог національного законодавства та міжнародних стандартів кібербезпеки [6].

Далі відкритий ключ публікується у централізованому сховищі, що забезпечує його доступність для подальшої верифікації як з боку кінцевих користувачів, так і автоматизованих систем розгортання чи оновлення програмних застосунків. На цьому ж етапі реалізується механізм розмежування прав підпису файлів: відповідні повноваження надаються обмеженому колу осіб (зокрема керівникам підрозділів і команд, а також фахівцям з інформаційної безпеки) із дотриманням принципу найменших привілеїв (PoLP) [7].

На визначених етапах життєвого циклу програмного продукту, що встановлюються відповідно до потреб організації (зокрема під час розширення функціональності, усунення помилок і вразливостей або навіть після кожної зміни у вихідному коді), уповноважена особа здійснює перевірку розроблюваного програмного забезпечення. У разі схвалення внесених змін формується хеш для кожного файлу, що підлягає підпису, після чого генерується цифровий підпис із використанням особистого закритого ключа.

Отримані підписи, відповідні хеші та супровідні метадані зберігаються у централізованому захищеному сховищі з обмеженими правами на редагування. Оновлення даних у такому сховищі відбувається виключно в процесі підписання файлів. До складу метаданих входить інформація про автора змін (розробника), особу, яка здійснила підпис, значення хешу, а також дата підписання. Це забезпечує можливість простеження всіх етапів перевірки та затвердження змін.

Ініціювання перевірки автентичності може здійснюватися як за запитом, так і автоматично. Зокрема, перевірка проводиться у випадках підозри на несанкціоновані зміни, виникнення інцидентів, пов'язаних із програмним кодом або його функціонуванням. Крім того, вона є невід'ємною складовою автоматизованих процесів у межах практик безперервної інтеграції, безперервної доставки та безперервного розгортання програмного забезпечення.

Процес верифікації реалізується послідовно:

Крок 1. Обчислення хешів поточних версій файлів, що підлягають перевірці.

Крок 2. Зіставлення отриманих хешів із відповідними цифровими підписами з використанням відкритих ключів, що зберігаються у централізованому сховищі.

Крок 3. Перевірка цілісності метаданих шляхом їх порівняння з даними, зафіксованими на етапі підписання.

Крок 4. Формування результату перевірки: у разі коректної верифікації підпису та незмінності метаданих підтверджується відсутність несанкціонованих змін у коді.

Загальна схема застосування цифрового підписування в системах контролю вихідного коду на основі запропонованого методу наведена на рис. 1.

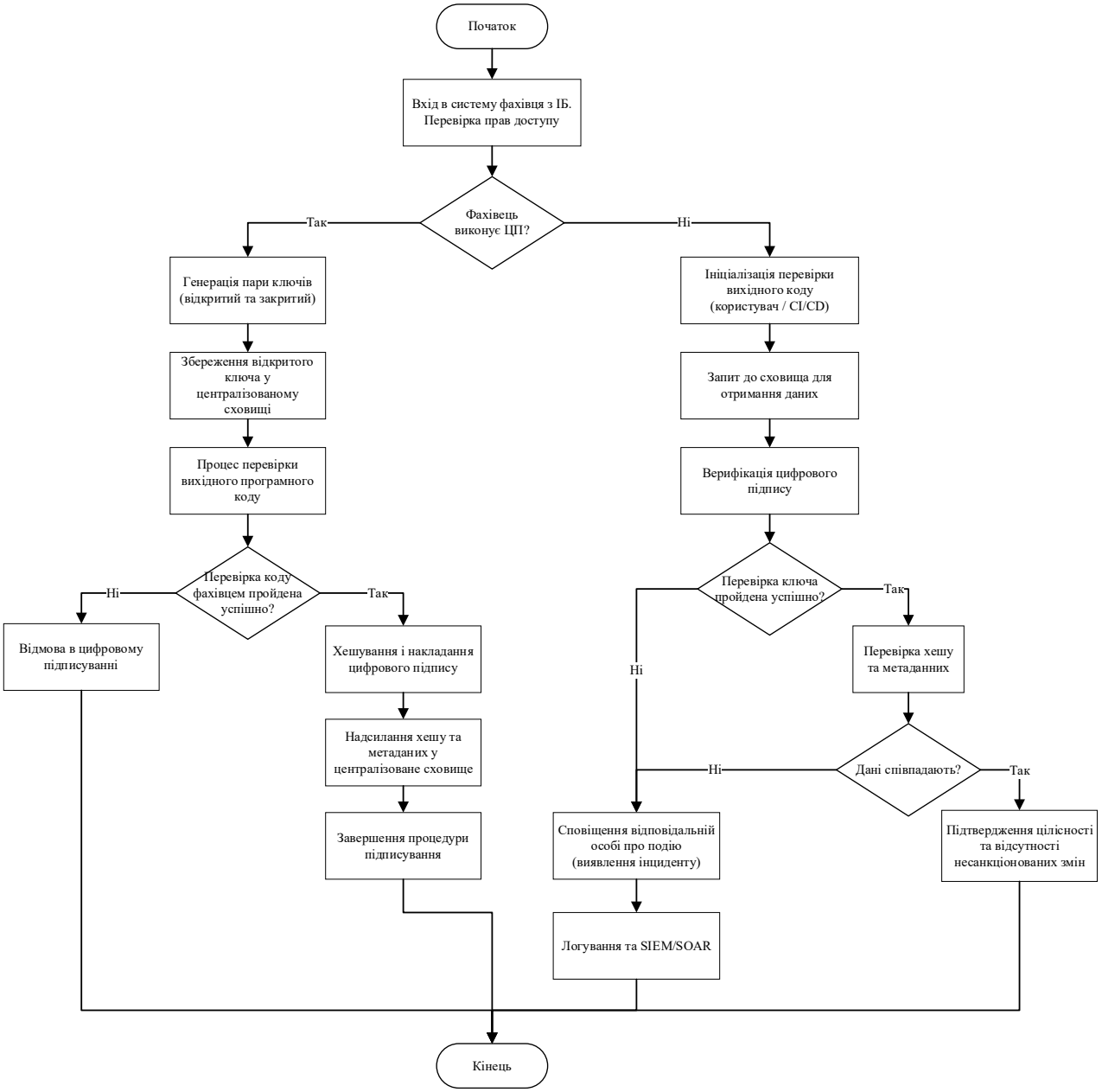

Рис. 1. Узагальнена схема запропонованого методу контролю версій вихідного коду

У випадку використання автоматизованих систем безперервної інтеграції, доставки та розгортання програмного забезпечення, успішне проходження перевірки є підставою для подальшого виконання відповідних процесів. Якщо ж цифровий підпис визнається недійсним або виявлено зміни метаданих, ініціатор перевірки отримує відповідне повідомлення, також здійснюється інформування уповноважених осіб або регулюючих органів. В автоматизованих сценаріях у такому разі виконання процесу розгортання або оновлення призупиняється.

Вибір криптографічних алгоритмів для системи контролю версій базується на вимогах до високої стійкості, швидкодії та відповідності міжнародним стандартам. Оптимальним рішенням для хешування є алгоритм SHA-256, який завдяки генерації 256-бітного хешу забезпечує надійний захист від колізій. Він поєднує широку підтримку у криптографічних бібліотеках із високою швидкістю обробки великих масивів даних, що мінімізує затримки при підготовці файлів до підпису.

Для створення цифрових підписів обрано алгоритм ECDSA на базі еліптичних кривих. Його ключова перевага полягає у забезпеченні високого рівня безпеки при значно меншій довжині ключів, ніж у традиційних методах, що прискорює обчислення та знижує навантаження на систему. Використання зв'язки SHA-256 та ECDSA відповідає стандартам NIST і гарантує цілісність коду без втрати продуктивності, що є критичним для автоматизованих процесів розробки.

Розглянемо особливості практичної реалізації програмного застосунку, призначеного для забезпечення автентифікації вихідного коду із використанням цифрового підпису в системах контролю версій. На відміну від загального

алгоритму, наведений підхід орієнтований на прикладну реалізацію відповідних процесів.

Крок 1. Ініціалізація та генерація ключів.

На початковому етапі реалізується механізм генерації криптографічної пари ключів у вигляді окремого програмного скрипту. Під час виконання передбачається введення ідентифікаційних даних користувача та вибір місця збереження закритого ключа. Відкритий ключ передається адміністратору для подальшого внесення до централізованого сховища.

Крок 2. Підпис файлів засобами застосунку.

Після перевірки програмного коду уповноважена особа ініціює процес підписання, використовуючи відповідний програмний інструмент. На цьому етапі задається шлях до закритого ключа та до файлів проєкту, після чого автоматично обчислюються хеш-значення файлів і формується їх цифровий підпис.

Крок 3. Передача та фіксація результатів підпису.

Згенеровані підписи, хеші та супровідні метадані передаються до централізованого сховища. Перед збереженням виконується автоматична перевірка відповідності підпису відкритому ключу, що дозволяє забезпечити коректність даних без додаткового втручання адміністратора.

Крок 4. Ініціалізація процесу верифікації.

Перевірка автентичності може запускатися як у ручному режимі, так і в межах автоматизованих процесів безперервної інтеграції, доставки та розгортання. При цьому застосунок ініціює запит до централізованого сховища для отримання необхідних даних.

Крок 5. Виконання перевірки та обробка результатів.

У процесі верифікації обчислюються хеші поточних файлів і здійснюється перевірка цифрових підписів із використанням відкритих ключів. У разі успішної перевірки підтверджується цілісність коду, що дозволяє продовжити виконання відповідних процесів. У протилежному випадку формується повідомлення про інцидент, а автоматизовані процеси, за наявності, зупиняються.

Далі проведемо аналіз рівня захищеності систем контролю версій при використанні вдосконаленого методу автентифікації та його відповідність міжнародним стандартам.

Впровадження методу автентифікації на основі цифрового підпису в системи контролю версій суттєво підвищує загальний рівень безпеки. Це дозволяє створити надійний механізм верифікації (узгодження) змін та посилити захист від типових атак.

Для детального порівняння запропонованого підходу з чинними механізмами автентифікації, у таблиці 1 наведено порівняльний аналіз стійкості до загроз.

Таким чином, проведений аналіз свідчить, що розроблений метод суттєво підвищує рівень захищеності системи завдяки забезпеченню незалежної перевірки цілісності файлів через цифровий підпис та унеможливленню несанкціонованого внесення змін у код без верифікації відповідальною особою.

Таблиця 1

Порівняльна характеристика стійкості методів до кіберзагроз

Загроза	*Існуючі методи*	*Обмеження існуючих методів*	*Запропонований метод (цифровий підпис)*
Несанкціонований доступ через викрадені облікові дані	Паролі, 2FA, SSH-ключі, токени доступу	Паролі можуть бути скомпрометовані; 2FA піддається фішингу; SSH-ключі та токени залежать від безпечного зберігання	Доступ до змін контролюється через обов'язкову перевірку цифрового підпису
Підробка змін у коді	Відсутній ефективний захист	Жоден із методів не забезпечує контроль цілісності змін у коді	Підпис кожного файлу дозволяє виявити будь-які несанкціоновані зміни
Компрометація ключів/токенів	SSH-ключі, токени доступу	Уразливість у разі викрадення або компрометації	Ключ підпису не надає доступу до репозиторію та виступає додатковим фактором контролю
Недосконалості або вразливості коду	Відсутній контроль	Немає механізму перевірки якості змін перед інтеграцією	Перед підписом зміни проходять перевірку уповноваженою особою
Соціальна інженерія	Паролі, 2FA, токени, SSH-ключі	Можливе отримання доступу через фішинг або компрометацію користувача	Приватний ключ не передається та використовується лише у визначених середовищах

Перевагами запропонованого рішення є автоматизація процесів контролю через інтеграцію з CI/CD-пайплайнами, повна прозорість аудиту завдяки збереженню метаданих, а також можливість взаємодії з системами моніторингу SIEM та SOAR. Крім того, впровадження цього методу сприяє виконанню вимог міжнародних стандартів з кібербезпеки, що підтверджує його практичну цінність для побудови захищених архітектур.

Для оцінки нормативно-технічної бази запропонованого рішення було проведено аналіз його відповідності ключовим галузевим регламентам. У таблиці 2 представлено порівняльний огляд вдосконаленого методу автентифікації на відповідність вимогам стандарту

ISO/IEC 27001, а також актуальним рекомендаціям NIST (National Institute of Standards and Technology) та переліку критичних заходів безпеки CIS Controls. Це дозволяє визначити ступінь інтеграції методу в сучасні системи управління інформаційною безпекою та його відповідність міжнародним практикам захисту даних.

Таблиця 2

Відповідність запропонованого методу міжнародним стандартам

Напрям забезпечення безпеки	*Реалізація у запропонованому методі*	*Відповідність стандартам*
Цілісність даних	Використання цифрових підписів для контролю незмінності файлів	ISO/IEC 27001 (A.10.1.1), NIST SP 800-57, CIS Control 5
Автентичність даних	Підтвердження джерела змін через асиметричну криптографію	ISO/IEC 27001 (A.10.1.2), NIST SP 800-63, CIS Control 13
Контроль доступу	Обмеження доступу до ключів і процесу підпису на основі ролей	ISO/IEC 27001 (A.9.4.1), NIST SP 800-53 (AC-3), CIS Control 4
Реагування на інциденти	Виявлення змін та формування повідомлень про інциденти	ISO/IEC 27001 (A.16.1.1), NIST SP 800-61, CIS Control 17
Автоматизація контролю	Інтеграція перевірок у CI/CD процеси	ISO/IEC 27001 (A.12.6.1), NIST SP 800-137, CIS Control 18
Захист передачі даних	Використання криптографічних механізмів при передачі	ISO/IEC 27001 (A.10.1.4), NIST SP 800-52/77, CIS Control 13

Отже, впровадження методу в СКВ на основі цифрового підпису суттєво посилює захищеність таких, мінімізуючи ризики несанкціонованого доступу, підробки коду та атак соціальної інженерії. Завдяки прозорості процесів і можливості автоматизованого аудиту підхід підвищує довіру до цілісності даних та пришвидшує реагування на інциденти. Відповідність рішення стандартам ISO 27001, NIST та CIS Controls підтверджує його релевантність сучасним безпековим вимогам, що дозволяє ефективно інтегрувати метод у процеси автоматизованої розробки (DevSecOps).

Для перевірки працездатності запропонованого методу було здійснено його практичну реалізацію у вигляді програмного застосунку, що дозволило провести експериментальну оцінку ефективності підходу в умовах, наближених до реальних сценаріїв використання систем контролю версій. Реалізація охоплювала всі ключові етапи методу, зокрема генерацію криптографічних ключів, підпис файлів та подальшу верифікацію їх цілісності.

У ході тестування підтверджено коректність базового сценарію роботи методу: за відсутності змін у підписаних файлах процес верифікації завершується успішно, що свідчить про збереження цілісності та автентичності даних. Водночас при внесенні несанкціонованих змін до файлів або додаванні нових, непідписаних елементів, система однозначно ідентифікує невідповідності, що призводить до негативного результату перевірки. Це підтверджує здатність методу ефективно виявляти порушення цілісності програмного коду.

Додатково було змодельовано ситуацію компрометації підпису шляхом зміни відповідних даних у сховищі за результатами якої можна зробити висновок, що у разі невідповідності цифрового підпису відкритому ключу верифікація також завершується з помилкою, а отже дозволяє виявити підробку або модифікацію підпису. Таким чином, метод забезпечує контроль не лише цілісності файлів, але й достовірності механізму підпису.

Окремо підтверджено можливість інтеграції перевірки в автоматизовані процеси безперервної інтеграції та розгортання, де невдале проходження верифікації призводить до зупинки подальших етапів, що, відповідно, дозволяє використовувати запропонований підхід як механізм превентивного контролю якості та безпеки програмного забезпечення.

Крім того, результати тестування засвідчили, що реалізація методу не потребує значних обчислювальних ресурсів і може бути впроваджена без істотного впливу на продуктивність системи. Виявлені інциденти фіксуються та зберігаються для подальшого аналізу, що створює передумови для інтеграції з системами моніторингу інформаційної безпеки.

Таблиця 3

Результати експериментальної перевірки методу

Сценарій тестування	*Очікуваний результат*	*Фактичний результат*
Перевірка незмінених файлів	Успішна верифікація	Успішна верифікація
Несанкціонована зміна файлу	Виявлення змін	Зміни виявлено
Додавання нового файлу	Виявлення невідповідності	Виявлено новий файл
Підробка цифрового підпису	Помилка верифікації	Підпис не підтверджено

Узагальнюючи отримані результати, можна стверджувати, що запропонований метод забезпечує ефективне виявлення несанкціонованих змін, контроль автентичності джерела змін та можливість інтеграції в сучасні процеси розробки програмного забезпечення, що підтверджує його практичну придатність.

Висновки. У статті авторами запропоновано багаторівневу захищену систему контролю версій вихідного коду програмного забезпечення на основі асиметричних криптографічних механізмів. Програмна розробка на основі запропонованого методу дозволяє забезпечити цілісність та автентичність даних, контроль доступу, оперативне реагування на інциденти, автоматизувати контроль та захищену передачу даних.

Зокрема, у роботі представлено алгоритм перевірки цілісності вихідного коду на основі цифрового підпису, що дозволяє підтвердити справжність вихідного коду та відсутність

несанкціонованих змін. Такий підхід суттєво підвищує надійність систем контролю версій.

Для захисту обрано два ключові інструменти асиметричної криптографії:

– SHA-256 – алгоритм, що забезпечує створення унікальних «відбитків» даних, які неможливо підробити, що гарантує цілісність коду;

– ECDSA – алгоритм на базі еліптичних кривих, який працює швидко і не потребує значних обчислювальних ресурсів, що важливо для великих проєктів.

Поєднання цих технологій забезпечує високу безпеку без втрати швидкодії. Метод без значних зусиль інтегрується в автоматизовані процеси розробки та відповідає сучасним стандартам безпеки.

Результатом роботи є гнучке та надійне рішення, яке дозволяє ефективно контролювати автентичність коду та захищати програмний продукт від стороннього втручання.

Література

[1] Golovko I. Intelligent Approaches to Source Code Protection. Measuring and computing devices in technological processes. 2025. No. 3. P. 120–125. URL: https://doi.org/10.31891/2219-9365-2025-83-16 (date of access: 10.03.2026).

[2] T. T. Grybyk. Features of Using a Version Control System in the Development of Educational Projects in Software Engineering / T. G. Grybyk // Innovative Pedagogy. – 2024. – No. 70, Vol. 1. – Pp. 183–188.

[3] SP 800-204, Security Strategies for Microservices-based Application Systems | CSRC. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/pubs/sp/800/204/final (date of access: 10.03.2026).

[4] Duda O., Orlov M., Pavliv I. Integration of Source Code Analysis Tools Into the Innovative DevSecOps Methodology. Vìsnik Nacìonal'nogo unìversitetu "L'vìvs'ka polìtehnìka". Serìâ Ìnformacìjnì sistemi ta merežì. 2025. Vol. 18, no. 1. P. 209–228. URL: https://doi.org/10.23939/sisn2025.18.1.209 (date of access: 10.03.2026).

[5] Improvement of database protection system from unauthorized modification based on blockchain technology and Proof-of-Work consensus algorithm / O. Saliieva et al. Herald of Khmelnytskyi National University. Technical sciences. 2024. Vol. 331, no. 1. P. 312–318. URL: https://doi.org/10.31891/2307-5732-2024-331-47 (date of access: 10.03.2026).

[6] ProfStandart. Qualification Center for Information Technology and Cybersecurity at the State Research Institute of Cybersecurity Technologies. URL: https://qc.csi.cip.gov.ua/en/pages/professional-qualifications (date of access: 10.03.2026).

[7] Analysis of modern approaches to access management in a cloud environment. Modern Information Security. 2025. Vol. 61, no. 1. URL: https://doi.org/10.31673/2409-7292.2025.017186 (date of access: 10.03.2026).

Hrytsak A.V., Bezpalyi K.V., Prysiazhnyi D.P., Lukanov M.V. Multi-level Secure Software Source Code Version Control System Based on Asymmetric Cryptographic Mechanisms

Abstract. *This article describes a multi-level secure software source code version control system based on the use of asymmetric cryptographic mechanisms. The main focus is on the development and justification of an authentication method that involves the application of digital signatures to ensure the control of software code integrity, authenticity, and availability, as well as to detect unauthorized modifications. The proposed approach is designed for integration into existing version control systems and involves the implementation of mechanisms that enhance the security level at all stages of the software development life cycle. Specifically, access to making changes to the code is controlled through a mandatory digital signature verification procedure, which limits the possibility of unauthorized interference. Signing each file ensures the detection of any changes in the code, even in cases of partial modification or the addition of new components. An additional advantage of the method is that the use of a signing key does not grant direct access to the repository, which reduces the risks of system compromise. Before signing, all changes undergo a review by an authorized person, allowing the combination of cryptographic mechanisms with software quality control processes. A crucial aspect is also the secure use of the private key, which is not shared with other users and is applied only within defined environments. To confirm the effectiveness of the proposed method, a software product was developed that implements the main stages of the software code signing and verification process. The conducted experimental study confirmed the feasibility of this approach, specifically its ability to detect unauthorized changes, ensure authenticity control, and integrate into automated software development processes. The results obtained indicate the practical value of the proposed method and the possibility of its application in modern software development systems.*

Keywords: *version control system, source code, digital signature, data integrity, cryptographic mechanisms, ECDSA, SHA-256.*

Грицак Анатолій Васильович*, к.т.н., доцент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет.*
Hrytsak Anatolii, *Ph.D., Associate professor of the Department of Management and Security of Information Systems, Vinnytsia National Technical University*

Безпалий Кирило Валерійович*, асистент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет.*
Bezpalyi Kyrylo, *Assistant Professor of the Department of Management and Security of Information Systems, Vinnytsia National Technical University*

Присяжний Дмитро Петрович, *асистент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет.*
Prysiazhnyi Dmytro*, Assistant Professor of the Department of Management and Security of Information Systems, Vinnytsia National Technical University*

Луканов Максим Всеволодович*, фахівець з інформаційної безпеки, KNESS Group.*
Lukanov Maksym*, information security specialist, KNESS Group.*