AMERICAN INTERNATIONAL UNIVERSITY – BANGLADESH

Faculty of Science & Technology

Programs (√): √ CSE □ EEE □ IPE □ CoE □ BBA □ BA

Course Name & ID: RESEARCH METHODOLOGY CSC4197

Semester: Spring 2024-2025 **Date of Submission:** …27/4/25……

SUBMITTED TO:

……………..*Dr. Md. Abdullah - Al - Jubair*…………………

SUBMITTED BY:

ID	NAME
22-49442-3	Ayesha Khatun
22-46868-1	Fardin-Al-Sezan
22-47243-1	Ananna Monjur
22-47278-1	Brishav Mondal
22-47249-1	Abu Nayem Md Arman

FOR FACULTY USE ONLY

Faculty comments: ………………………..,

Signature: ………………………. **Date: ……………………….**

Topic of Interest:

Blockchain in Smart Contracts and Cybersecurity

Blockchain technology, which first gained attention as the backbone of Bitcoin, has since grown into a powerful innovation with applications across many different industries. One of the most impactful uses of this technology is in smart contracts—self-executing contracts with the terms directly written into code. These contracts run on decentralized blockchain networks and carry out their terms automatically, without needing a middleman. Their biggest advantages include transparency, fast processing, lower costs, and independence from third parties. That said, as more industries begin to adopt smart contracts, cybersecurity challenges are becoming more pressing. Weaknesses in the code can be exploited, and history has already shown the risks—like the infamous DAO hack in 2016, where a small bug in the smart contract's code led to the loss of millions of dollars. This highlights the growing importance of cybersecurity in the blockchain ecosystem. To address these concerns, researchers and developers have been working on ways to make smart contracts safer. This includes creating secure programming languages, applying formal verification methods to check for errors, and enforcing thorough auditing processes. As smart contracts continue to shape the future of finance, supply chains, and even legal systems, making sure they're secure will be key to building trust and allowing the technology to grow.

Problem Statement:

Blockchain is becoming a go-to technology for automating digital agreements through smart contracts—self-running pieces of code that run on platforms like Ethereum and Hyperledger. These contracts offer several advantages, including decentralization, transparency, and speed. However, they also come with a serious downside: cybersecurity risks. In the past, hackers have taken advantage of weaknesses in smart contract code to steal or lock up massive amounts of digital assets. Well-known examples like the DAO hack in 2016 and the Parity wallet hacks in 2017 show just how devastating these flaws can be when security isn't properly handled.

This study specifically focuses on smart contract developers and organizations (who) working with blockchain technologies, particularly those using Ethereum and similar platforms (where). The goal is to identify the most common security vulnerabilities in smart contracts (what) and understand how and why these issues arise. Are they caused by simple coding mistakes, a lack of auditing, or the absence of formal verification tools (how and why)? These are the key questions we aim to answer.

These challenges are not hypothetical—they're happening now, especially as smart contracts become a standard part of industries like finance, supply chain management, and law (when). As more businesses begin to depend on blockchain for critical operations, it's becoming increasingly important to secure smart contracts against potential threats.

Articles that support this problem include:

- Mehar, M. I., et al. (2019). *Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack.*
- Atzei, N., Bartoletti, M., & Cimoli, T. (2017). *A survey of attacks on Ethereum smart contracts.*
- Luu, L., et al. (2016). *Making smart contracts smarter.*

This research aims to dig deeper into what makes smart contracts vulnerable and how those risks can be reduced through improved coding standards, thorough auditing, and the use of advanced security tools.

Primary Research Question:

What are the main cybersecurity vulnerabilities in smart contracts, and how can developers reduce these risks using secure coding practices, thorough auditing, and verification techniques?

This research question aims to uncover the most common and critical security flaws found in smart contracts built on blockchain platforms. It also looks at practical ways to prevent these issues from occurring in the first place. The study will explore how and why these vulnerabilities emerge, what tools are available to identify and address them, and how developers can apply best practices to improve the overall security and reliability of smart contract systems.

The ultimate goal is to offer actionable strategies that developers and organizations can use to build safer, more trustworthy smart contract applications.

Research Objectives:

This study will aim to:

1. **To identify and categorize the main cybersecurity vulnerabilities** in smart contracts, such as reentrancy attacks, integer overflows, and flaws in access control. This includes understanding the technical roots of these problems and how they are commonly exploited by attackers.

2. **To investigate existing tools and best practices** used in securing smart contracts, including auditing frameworks, formal verification methods, and secure programming languages like Vyper or frameworks like OpenZeppelin.
3. **To analyze how developers and organizations can adopt secure coding practices** to reduce the risk of smart contract vulnerabilities in real-world blockchain environments.

Hypothesis:

Smart contracts that are built using secure coding practices, formal verification methods, and thorough auditing processes are expected to have significantly fewer cybersecurity vulnerabilities compared to those developed without such precautions.

More specifically, development teams and blockchain platforms that utilize tools like static analyzers, formal testing frameworks, and established secure libraries—such as OpenZeppelin contract templates—are likely to be more effective at preventing common threats. These include reentrancy attacks, logic flaws, and unauthorized access vulnerabilities. This hypothesis will be tested by reviewing reported vulnerabilities, analyzing case studies of smart contract breaches, and evaluating the effectiveness of various security tools and coding standards.

Literature Review:

1. **Understanding Smart Contact Vulnerabilities**

   Atzei et al. (2017) provided one of the earliest and most comprehensive surveys on vulnerabilities in Ethereum smart contracts. They outlined common issues like reentrancy, unchecked external calls, and integer overflows/underflows. By analyzing several real-world hacks, the authors showed that many security flaws stem from weak programming practices and a lack of understanding of Ethereum's execution model. Their work offered a well-structured classification of known vulnerabilities, which has become a reference point for both developers and auditors. They also advocated for secure coding patterns and the use of automated vulnerability detection tools. However, given the rapid evolution of blockchain technologies—particularly in areas like DeFi and NFTs—the study is somewhat outdated and doesn't fully address newer threats.

2. **The DAO Attack and Its Lessons**

   Mehar et al. (2019) took an in-depth look at the infamous DAO attack, where a reentrancy bug allowed hackers to siphon off millions of dollars. The DAO, which operated as a decentralized investment platform, failed to catch this vulnerability due to insufficient code review, a lack of formal verification, and limited developer experience. This study laid out a detailed timeline of the incident and highlighted how blockchain's immutable nature worsened the situation. Beyond the technical flaws, the paper also explored governance issues and organizational gaps, arguing for stronger community oversight and the adoption of secure-by-design principles. It remains a crucial case study for understanding both the technical and structural risks of smart contract projects.

3. **Formal Verification for Smart Contracts**

Luu et al. (2016) introduced "OYENTE," one of the earliest tools for detecting smart contract vulnerabilities using static analysis. Their research showed that many bugs could be identified before contracts go live. The team emphasized the role of formal verification—mathematically proving that a contract behaves as intended—as a powerful way to improve reliability. However, they also acknowledged the downsides: formal methods can be complex and hard for most developers to adopt. Despite its challenges, this work was pioneering in bringing rigorous software verification techniques into the blockchain development space.

4. **Auditing Practices and Their Effectiveness**

In their 2018 study, Wüst and Gervais evaluated how smart contracts are audited in the Ethereum ecosystem. They found that while many projects claimed to undergo audits, the quality and depth of these reviews varied widely. Their findings showed that formal, third-party audits significantly reduce vulnerabilities, but aren't foolproof—some bugs still make it through, especially in complex or innovative systems. The study concluded that audits are essential, but not enough on their own. Continuous monitoring and the ability to upgrade contracts post-deployment are equally important for maintaining security.

5. **Secure Development Frameworks**

Delmolino et al. (2016) explored how smart contract security is addressed in educational settings and during early-stage development. They found that even computer science students—otherwise skilled programmers—struggled to write secure smart contracts. Many introduced serious bugs despite having a technical background. The study pointed to the importance of secure development frameworks and recommended the use of tools like OpenZeppelin, which provide reusable and pre-audited smart contract templates. This work highlights how good educational resources and developer tools can have a direct impact on the security of blockchain applications.

6. **Real-World Case Studies of Exploits**

Zhou et al. (2020) conducted a large-scale review of actual smart contract exploits, drawing on Ethereum blockchain data and developer activity on GitHub. Their analysis revealed that the same kinds of bugs—such as reentrancy and timestamp dependency—continue to surface in new contracts. One key takeaway from the study was that many developers fail to use available security tools, and because smart contracts are often immutable once deployed, it's difficult to patch issues later. The authors stressed the need for better developer education, standardized security practices, and tools that support smart contract upgrades and lifecycle management.

Selection of Design (Mixed):

This research adopts a **Mixed Methods** approach as the most suitable design.

A mixed methodology is ideal because it allows the study to combine the strengths of both qualitative and quantitative research. On the qualitative side, the research will explore the nature of cybersecurity vulnerabilities in smart contracts and how developers and organizations respond to them. This will involve examining detailed case studies—such as the DAO attack—reviewing smart contract code analysis reports, and conducting expert interviews to gather deeper insights.

On the quantitative side, the study will analyze data from blockchain platforms like Ethereum to identify how frequently specific vulnerabilities occur. It will also use audit reports and security statistics to measure the impact of known exploits and evaluate the effectiveness of various tools and frameworks.

Research Variables:

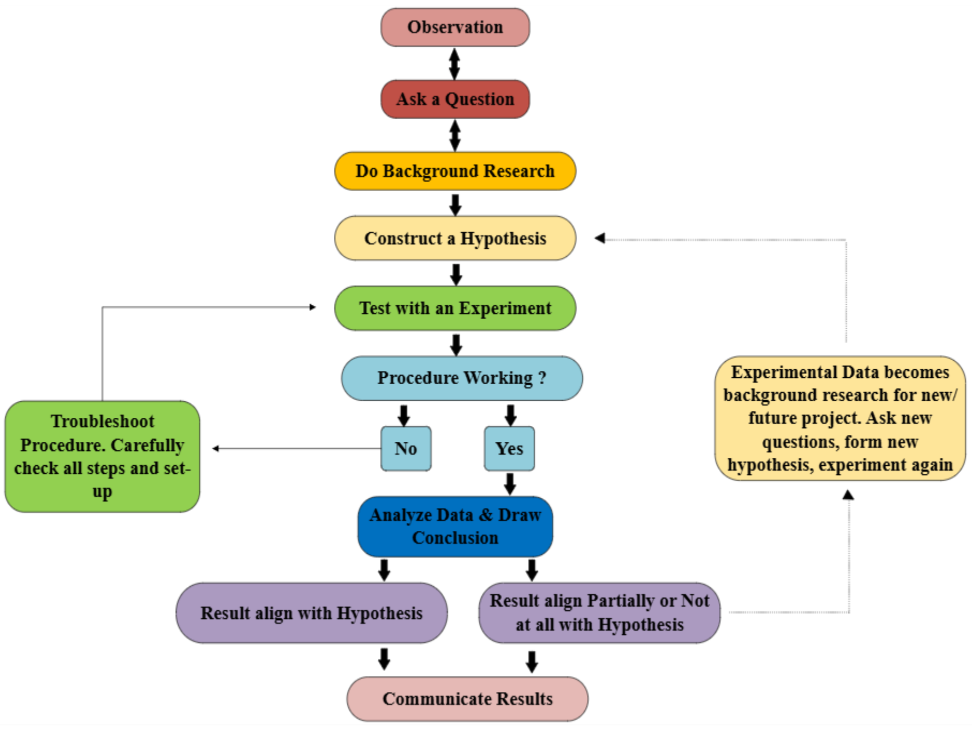
1. Independent Variables:

- **Development Practices:**
  - Use of formal verification tools (e.g., Oyente, Mythril)
  - Use of audited frameworks/libraries (e.g., OpenZeppelin)
  - Type of programming language (Solidity vs. Vyper)
  - Audit status (audited vs. non-audited contracts)
- **Security Tools:**
  - Static code analysis tools
  - Dynamic testing environments
  - Bug bounty programs

2. Dependent Variables:

- **Number of vulnerabilities detected** in smart contracts (e.g., reentrancy, integer overflow, logic errors)
- **Severity of exploits** (measured by funds lost or frozen)
- **Security audit scores** or ratings
- **Number of successful vs. failed attacks** after deployment

Scientific Method (Diagram):


Observation
Ask a Question
Do Background Research
Construct a Hypothesis
Test with an Experiment
Procedure Working ?
No
Yes
Troubleshoot Procedure. Carefully check all steps and set-up
Experimental Data becomes background research for new/ future project. Ask new questions, form new hypothesis, experiment again
Analyze Data & Draw Conclusion
Result align with Hypothesis
Result align Partially or Not at all with Hypothesis
Communicate Results

References (APA 6 Style):

Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK). *International Conference on Principles of Security and Trust*, 164–186. Springer.

Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2016). Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. *International Conference on Financial Cryptography and Data Security*, 79–94. Springer.

Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 254–269.

Mehar, M. I., Shier, C., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., Mclaren, M., & Laskowski, M. (2019). Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack. *Journal of Cases on Information Technology (JCIT)*, 21(1), 19–32.

Wüst, K., & Gervais, A. (2018). Do you really need a blockchain for everything? *IACR Cryptology ePrint Archive*, 2017, 375.

Zhou, Y., Kumar, D., Mason, J., Saxena, P., & Bailey, M. (2020). Erays: Reverse engineering Ethereum's opaque smart contracts. *USENIX Security Symposium*, 1371–1385.