Borneo International Journal eISSN 2636-9826; Vol. 9 (2); 2026; 9-15
Published by *Majmuah Enterprise*
www.majmuah.com

Zero-Knowledge Proofs in Penetration Testing: Verifying Vulnerabilities Without Revealing Sensitive Information

CUI YUCHONG and MOHAMAD FADLI BIN ZOLKIPLI

School of Computing, College of Art and Science, Universiti Utara Malaysia (UUM), 06010 Sintok, Kedah, MALAYSIA
Email:Cuiyuchong00@gmail.com, m.fadli.zolkipli@uum.edu.my

Received: April 15, 2026
Accepted: May 07, 2026
Online Published: June 04, 2026

Abstract

In penetration testing practice, vulnerability verification usually relies on the reproduction of the exploitation process or the description of the attack path. Although this method can provide direct evidence of the authenticity of the vulnerability, it may also introduce additional risks if the vulnerability is not repaired or involves sensitive systems, such as the abuse of the attack method or the internal system. The information was overexposed. Therefore, how to minimise information disclosure while ensuring the validity of verification has gradually become a practical problem that needs to be faced in the process of security assessment. Focussing on this contradiction, zero-knowledge proof provides a technical path different from the traditional way of thinking. Its basic idea is not to hide the information itself, but to transform the question of "whether the loophole can be triggered" into a formal proposition by changing the verification method, and verify it through the proof mechanism, so as not to disclose the specific On the premise of inputting, executing the path or using the details, the verifier can confirm the existence of the vulnerability. Based on this idea, this article analyses the application of zero-knowledge proof in vulnerability verification in combination with the typical process in penetration testing. It focusses on the formal modelling method of vulnerability exploitation process, the construction method of constraint system, and the proof generation and verification mechanism, and further combines existing research to disclose vulnerabilities, Sort out the application possibilities in scenarios such as test result verification and software supply chain security. The analysis results show that this method has certain advantages in reducing the risk of sensitive information leakage and enhancing the independence of the verification process. It is especially suitable for environments with weak multi-party participation or trust. However, at the same time, it still has certain limitations in terms of computing overhead, model expression ability and engineering realisation complexity. Relevant questions The question needs to be weighed according to the specific scenario in practical application, and needs to be further improved by follow-up research.
Keywords: zero-knowledge proof(ZKP); penetration test; vulnerability verification; information disclosure control; software supply chain security

1. Introduction

In recent years, the development of software systems has changed significantly. The transformation from a single application to a microservice architecture, as well as the extensive use of open source components and third-party dependence, have made the system more functionally flexible, but at the same time, its potential attack surface is also expanding. (Gokkaya, B., Aniello, L., & Halak, B. 2023)Studies have generally pointed out that security problems in modern software often do not come from a single module, but are hidden in complex dependencies and interaction logic. (Sa'ad et al., 2026)Against this background, it is difficult to meet the actual needs by relying solely on passive Defence mechanisms, and the penetration test of actively discovering problems has gradually become an important means in security practice. (Kapur, S., & Saurabh, P. 2025) However, in practice, vulnerability "discovery" and "verification" are not issues at the same level. The discovery of vulnerabilities can be done through scanning, fuzzy testing or code analysis, while verifying vulnerabilities usually requires further progress - testers need to construct specific inputs, reproduce trigger paths, and sometimes even provide complete utilisation code. (Li et al., 2025) In other words, verification itself often means a certain degree of "exposure" to the internal behaviour of the system. In most cases, this exposure is controllable, but the process becomes delicate when the vulnerability has not been fixed or the system itself is highly sensitive. Some practises in reality, such as phased disclosure or limiting the scope of information dissemination, do mitigate risks to a certain extent, but their effects largely depend on the trust base between the participants. Once this relationship of trust is not established or the number of participants increases (such as in the crowd test of vulnerabilities or supply chain security assessment), the problem becomes more complicated: the verifier needs to be sure that the vulnerability does exist, while it is difficult for the provider to complete the proof without revealing key details. This contradiction of "both proving and hiding" has not been well resolved under the existing technical framework. (Cuéllar et al., 2025)

From this perspective, it may be more meaningful to remodel the verification process itself than simply restricting the flow of information. Zero knowledge proof is a kind of technology that has attracted attention under this idea. Its core idea is not to hide the information itself, but to change the way of "proof" - the certifier can make the verifier believe that a conclusion is valid without disclosing the specific content. Introducing this mechanism into vulnerability verification can transform the process of relying on using details into a formal proof of the fact that "loopholes can be triggered", so as to theoretically avoid unnecessary information leakage. (Scaramuzza et al., 2025) Of course, this way of thinking is not without a price. On the one hand, the exploit process often involves complex program execution status, which is abstracted into a computing model that can be used for zero-knowledge proof, which requires additional modelling work; on the other hand, the existing zero-knowledge proof technology still has certain limitations in computing efficiency, especially in the face of large-scale software systems, its actual The feasibility is still to be verified. Some existing exploratory studies show that the method has some operability in specific types of vulnerabilities (such as memory errors), but its applicability in more complex scenarios is still unclear. (Sakwa et al., 2025)

Based on the above background, this article tries to sort out this problem from the two levels of method mechanism and application scenario. Instead of treating zero-knowledge proof as an alternative to the existing penetration testing process, it is better to regard it as a supplementary means - providing a new verification path in a specific environment where information disclosure needs to be controlled. Specifically, (Cuéllar et al., 2025) this article will analyse the formal representation of the vulnerability verification process, the construction method of zero-knowledge proof and its application in actual security scenarios, and discuss its advantages and limitations in combination with existing research. It should be noted that this article focusses more on the feasibility and application boundaries of the method itself, rather than engineering the specific realisation. Through the systematic collation of relevant issues, it is hoped that it can provide a relatively clear analytical framework for subsequent research, and also provide reference for the further application of zero-knowledge proof in the field of software security.

2. Literature review

2.1 Development background of zero-knowledge proof technology

On the whole, zero-knowledge proof was originally mainly used as a theoretical tool in cryptography to solve the problem of completing identity verification or knowledge proof without leaking information. Early research focussed on interactive proof protocols, and its scope of application is relatively limited. With the introduction of Fiat-Shamir transformation and other methods, non-interactive zero-knowledge proof has gradually become the focus of research, making the proof process more operational in engineering. (Williams, A. 2024; Cuéllar et al., 2025) In recent years, the development of zk-SNARK, zk-STARK and other schemes has further promoted the application in this field. Such methods have achieved an ideal balance between proving efficiency and verification expenses, making the verifiability of complex computing processes possible. Studies have generally agreed that the maturity of this type of technology provides a basis for the inclusion of procedural execution behaviour in the proof framework. (El-Hajj, M., & Oude Roelink, B. 2024) However, it should be noted that most of these methods are for general-purpose computing, and their application in specific security scenarios still needs to be adjusted according to the characteristics of the problem.

2.2 Research on vulnerability verification and information control

In the field of vulnerability verification, how to control information disclosure while ensuring the credibility of the results has always been a relatively practical but not easy to solve thoroughly. Early work usually relies on a controlled environment, such as a sandbox mechanism or a trusted execution environment to limit the leakage of sensitive information. Such methods reduce the risk to a certain extent, but do not change the basic pattern of the verification process relying on the use of details. Some studies also try to abstractly represent the vulnerability trigger path through program analysis, such as describing the relationship between input and execution path with symbolic execution techniques, so as to reduce dependence on the complete utilisation process. This method theoretically helps to reduce the degree of information exposure, but in the actual verification process, it is often necessary to share some key data, so its effect is limited in complex scenarios. (TU, H. 2025) In general, the research at this stage is more about improving under the existing verification framework than fundamentally changing the idea of "proving through display".

2.3 Application exploration of zero-knowledge proof in vulnerability verification

With the gradual maturity of zero-knowledge proof technology, some studies began to try to introduce it into the vulnerability verification scenario. Some work transforms the vulnerability exploitation process into verifiable calculations, such as formalising the question of "whether a specific input can trigger abnormal behaviour" by constructing an arithmetic circuit or constraint system, and using the proof mechanism for verification.(Yang et al., 2022)In memory security vulnerabilities (such as buffer overflow, cross-border access, etc.), such methods have made

some progress. The relevant results show that it is feasible to verify the existence of vulnerabilities without disclosing specific inputs or exploiting paths. (Cuéllar et al., 2025).Judging from the existing research, this idea still faces many challenges in practical application. First of all, different types of vulnerabilities differ greatly in the form of expression, especially those involving complex business logic, and their behaviour is difficult to describe through a unified model; (Metin et al., 2025) secondly, converting the real program execution process into a constraint system suitable for zero-knowledge proof has a high complexity itself. In large-scale software systems It is easy to bring significant calculation overhead. In addition, some studies have also pointed out that the incompleteness of the constraint modelling process may affect the accuracy of the erification results, which increases the uncertainty of practical application to a certain extent. existing research has verified the potential value of zero-knowledge proof in vulnerability verification from different angles, especially in scenarios where information disclosure needs to be controlled, providing a new technical path. However, at the same time, there is still room for further improvement in terms of expression ability, efficiency and engineering realisation, and relevant issues still need to be explored continuously. (Wang et al., 2026)

3. Methodology

When introducing zero-knowledge proof into penetration testing scenarios, it is necessary to build a systematic method framework that can support vulnerability verification. Unlike the traditional verification method of relying on the reproduction of vulnerabilities, this method takes "verifiable calculation" as the core and realises the indirect verification of the existence of vulnerabilities through formal modelling and mathematical proof. This section elaborates on the methodology from three aspects: overall process, modelling strategy and verification mechanism. (Afshar, A., & Goyal, R. 2025)

3.1 Overall Method Framework

On the whole, the vulnerability verification process based on zero-knowledge proof can be divided into four main stages: vulnerability identification, formal modelling, constraint system construction, and proof generation and verification. This process reflects the shift from empirical analysis to formal verification.

Figure 1: Zero-knowledge Proof Vulnerability Verification Flow Chart

In this framework, there is a clear dependency between the stages. The vulnerability identification stage provides the basis for subsequent modelling. The formal modelling stage abstracts the vulnerability behaviour into a mapping between the input and the system state, and the constraint system construction further transforms the model into a verifiable mathematical expression. Finally, the verification of the existence of vulnerabilities is realised through the proof generation and verification stages. (Sheybani et al. 2025)

3.2 Vulnerability Modelling and Constraint Expression

In the whole method framework, modelling and constraint expression are the most critical links. Its core task is to transform the vulnerability exploitation process into a formalised model and further express it as a constraint system that can be proved to be handled by zero knowledge. For vulnerabilities with a relatively simple structure, such as memory cross-border access, the trigger conditions can be more directly expressed as constraints; while for vulnerabilities involving permission control or complex business logic, the process of changing the state of the system needs to be abstracted. In this process, there is a clear trade-off between the accuracy and complexity of modelling: high-precision models help to improve the accuracy of verification, but also increase the calculation cost of proof generation. (Sheybani et al. 2025) Therefore, in practical applications, the "critical path modelling" strategy is usually adopted, that is, only the core execution path related to vulnerability triggers is retained. While reducing the complexity, this method may also introduce the problem of incomplete expression, which needs to be adjusted according to specific scenarios. (Liang et al., 2025)

3.3 Certification generation and verification mechanism

After completing the modelling and constraint system construction, the verification process is realised through the zero-knowledge proof mechanism. The certifier generates the certificate based on private input (i.e. the vulnerability trigger data), while the verifier only verifies the certificate through public parameters without obtaining the specific input or execution path. (Zhao et al., 2024) This mechanism realises the separation of verification and information disclosure, so that vulnerabilities can be identified without exposing the details of exploitation. (Wei et al., 2025) However, it also means that the verification results are highly dependent on the accuracy of the modelling and constraint system. Once there is a deviation in the model, even if it is proved to be valid, it may not be able to truly reflect the vulnerability status. Therefore, in practical applications, it is necessary to strictly control the model design and verification process. (Liang et al., 2025)

4. Discussion

On the basis of the above methodological framework, we can further analyse the performance of zero-knowledge proof in practical applications, including system architecture, application scenarios, method advantages and limitations.

4.1 System architecture and information flow reconstruction

From the perspective of system realisation, the zero-knowledge proof method usually involves two core roles: the proof party and the verification party. The former is responsible for vulnerability modelling and proof generation, and the latter is responsible for verifying the validity of the proof. Compared with the traditional method, the biggest change in this structure is the redesign of the information flow.

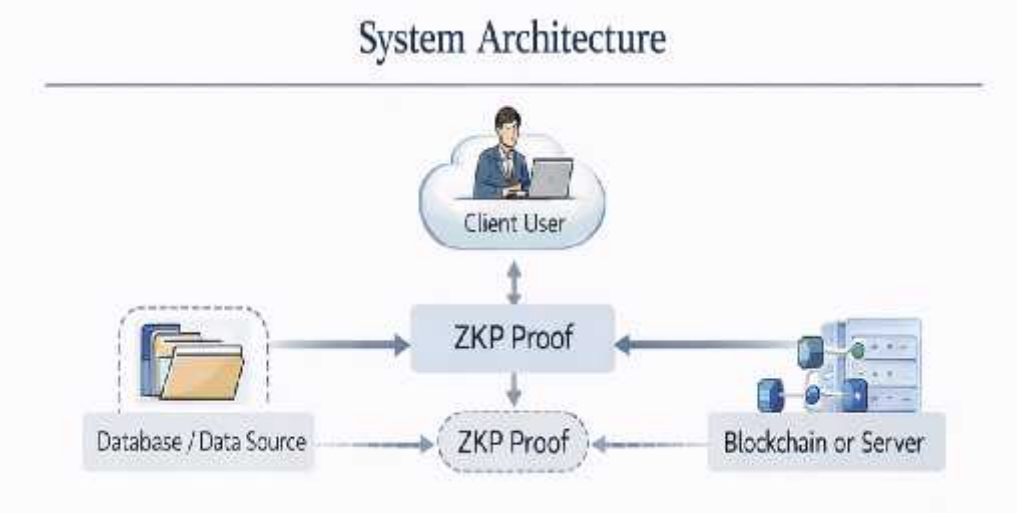

Figure 2 System architecture diagram (Prover–Verifier model)

In this architecture, the verification process no longer relies on the details of vulnerability exploitation, but on the proof results themselves. This mechanism reduces dependence on trust and is especially suitable for cross-organisational verification scenarios. However, its effectiveness depends on the accuracy of the modelling and constraint system, which has also become a key issue in system design. (Kiesel, J., & Heiss, J. 2025)

4.2 Application scenario and applicability analysis

In practical applications, the zero-knowledge proof method is mainly applicable to scenarios with high requirements for information disclosure, such as vulnerability disclosure, vulnerability reward platform and software supply chain security verification. In these scenarios, this method can reduce the exposure of sensitive information while ensuring the validity of verification. (Tang et al., 2024) However, the applicability of this method depends to some extent on the formalisation of the vulnerability. For vulnerabilities with a clear structure, the modelling and verification process is relatively direct; for complex business logic vulnerabilities, the modelling cost is high, which may limit the practical application.

4.3 Method Comparison and Complementarity

Compared with traditional penetration testing methods, there is a significant difference in the verification logic of zero-knowledge proof. The traditional method emphasises the visibility of the exploit process, while the zero-knowledge proof emphasises the verifiability of the verification results.

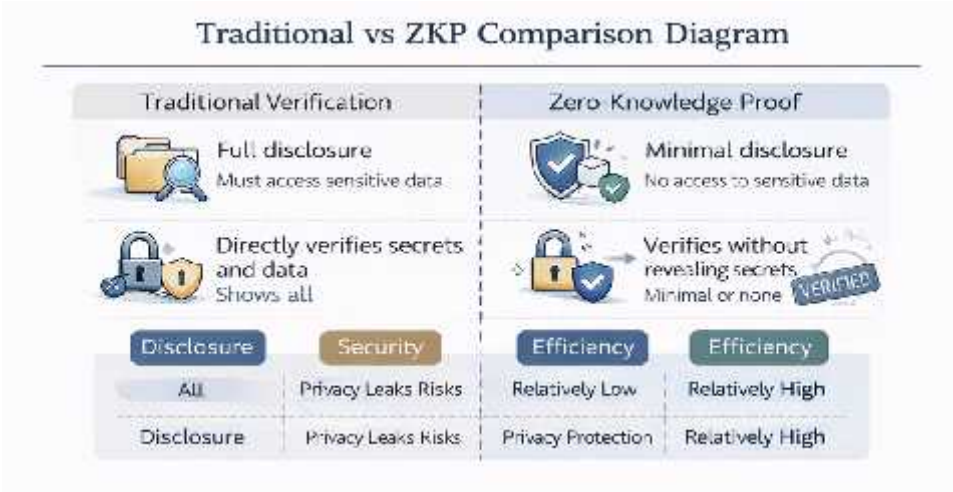

Figure 3:Comparison between the traditional verification process and the ZKP verification process

Table 1 Comparison between traditional penetration test and ZKP method

Contrast dimension	Traditional penetration test	ZKP method
Verification method	Take advantage of reproduction	Mathematical proof
Information Disclosure	High	Low
Verifiability	Trust-dependent	Independently verifiable
Computational Overhead	Low	High
Applicable Scenarios	General-purpose	High-security requirements

4.4 Limitations and future development direction

Although zero-knowledge proof provides a new methodological path for vulnerability verification, it still faces many challenges in practical application. First of all, the modelling complexity is high, and there is a lack of a unified expression between different types of vulnerabilities; secondly, the calculation overhead of the proof generation process is large, which may affect efficiency in complex systems; in addition, the current lack of a mature tool chain and standardised framework limits its popular application. Future research can be carried out from the following directions: first, improve the automation of modelling and proof generation to reduce the threshold of use; second, optimise the design of the constraint system to control the complexity of calculation while ensuring the expression ability; third, promote the integration with the existing security tool chain so that it can be embedded in actual penetration tests. Process; Fourth, explore more complex application models such as cross-organisational verification and multi-vulnerability joint proof.

5. Conclusion

Putting zero-knowledge proof into the specific process of penetration testing, it mainly changes the implementation of vulnerability verification, not the vulnerability discovery itself. It can be seen from the previous methodological framework that this process actually transforms the verification path that originally relied on the reproduction of vulnerability exploitation into a computational problem consisting of modelling, constraint expression, and proof

generation and verification. In other words, verification no longer relies on "showing the attack", but on "proving that the attack is valid", which makes it possible to identify the vulnerability without exposing the exploit details. In this transformation process, modelling is still the most critical and easily underestimated link. For loopholes with a clear structure, their behaviour can be more directly expressed as constraints, so as to smoothly enter the proof stage; however, when the problem involves complex business logic or multi-stage state changes, the difficulty is often not the proof itself, but how to accurately portray the vulnerability trigger process.(Pailoor et al., 2023) Once the modelling deviates from the actual execution semantics, even if the verification results are valid, they may lose their practical meaning. At the same time, it is proved that the computational overhead generated is still high in complex scenarios, which also limits the scope of application of the method to a certain extent. From the perspective of application, this method is more suitable for scenarios with high requirements for information disclosure, such as vulnerability disclosure or cross-organisational verification. In these cases, the focus is often on confirming the existence of vulnerabilities, rather than disclosing their specific utilisation methods. (Kiesel, J., & Heiss, J. 2025) Zero-knowledge proof provides a compromise path here, so that verification can be completed under the premise of controlling information exposure. Therefore, rather than treating it as an alternative to traditional penetration tests, it should be understood as a supplement that is more suitable under specific conditions. With the gradual improvement of modelling methods, proof efficiency and related tools, there is still room for further expansion of this idea in the future. (Sheybani et al., 2025)

Acknowledgment

The author would like to express sincere appreciation to Universiti Utara Malaysia for providing the academic environment, resources, and opportunities that supported the completion of this study. Special gratitude is extended to the course lecturer for Hacking & Penetration Tests for valuable guidance, constructive feedback, and continuous encouragement throughout the research process.
The author is also grateful to classmates and peers whose discussions and shared perspectives helped refine the analytical insights presented in this paper. Finally, heartfelt appreciation is conveyed to the author's family for their patience, support, and constant motivation, which remained an important source of strength during the preparation of this work.

References

Gokkaya, B., Aniello, L., & Halak, B. (2023). Software supply chain: review of attacks, risk assessment strategies and security controls. *arXiv preprint arXiv:2305.14157*. https://doi.org/10.48550/arXiv.2305.14157

Sa'ad, U., Na, W., Dao, N. N., & Cho, S. (2026). Dynamic dependency-aware vulnerability and patch management for critical interconnected systems. *Journal of Network and Computer Applications*, 104436. https://doi.org/10.1016/j.jnca.2026.104436

Li, S., Li, Y., Chen, Z., Dong, C., Wang, Y., Li, H., ... & Zhu, H. (2025, April). Transferfuzz: Fuzzing with historical trace for verifying propagated vulnerability code. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)* (pp. 268-280). IEEE. https://doi.org/10.1109/ICSE55347.2025.00061

Cuéllar Gempeler, S., Harris, B., Parker, J., Pernsteiner, S., Sweet, I., & Tromer, E. (2025). Cheesecloth: Zero-Knowledge Proofs of Real-World Vulnerabilities. *ACM Transactions on Privacy and Security*, *28*(4), 1-35. https://doi.org/10.1145/3747589

Scaramuzza, F., Ferreira, R. C., Suller, T. M., Quattrocchi, G., Tamburri, D. A., & Heuvel, W. J. V. D. (2025). " Show Me You Comply... Without Showing Me Anything": Zero-Knowledge Software Auditing for AI-Enabled Systems. *arXiv preprint arXiv:2510.26576*. https://doi.org/10.48550/arXiv.2510.26576

Sakwa, C. O., Anyembe, A. O., & Li, F. (2025). A survey of folding-based zero-knowledge proofs. *Information Sciences*, 122698. https://doi.org/10.1016/j.ins.2025.122698

Williams, A. (2024). Zero-knowledge proofs and their role within the blockchain. *Communications of the ACM*, *67*(7), 6-7. https://doi.org/10.1145/3653478

El-Hajj, M., & Oude Roelink, B. (2024). Evaluating the efficiency of zk-snark, zk-stark, and bulletproof in real-world scenarios: A benchmark study. *Information*, *15*(8), 463. https://doi.org/10.3390/info15080463

TU, H. (2025). Boosting symbolic execution for vulnerability detection. https://ink.library.smu.edu.sg/etd_coll/704/#:~:text=https%3A//ink.library.smu.edu.sg/etd_coll/704

Yang, Y., Han, S., Xie, P., Zhu, Y., Ding, Z., Hou, S., ... & Zheng, H. (2022). Implementation and optimization of zero-knowledge proof circuit based on hash function sm3. *Sensors*, *22*(16), 5951. https://doi.org/10.3390/s22165951

Metin, B., Wynn, M., Tunalı, A., & Kepir, Y. (2025). Business Logic Vulnerabilities in the Digital Era: A Detection Framework Using Artificial Intelligence. *Information*, *16*(7), 585. https://doi.org/10.3390/info16070585

Wang, Z., Zhu, X., Zhang, X., Deng, Y., & Song, X. (2026). Sending zero-knowledge proofs to the future. *Cybersecurity*, *9*(1), 110. https://doi.org/10.1186/s42400-025-00453-7

Afshar, A., & Goyal, R. (2025). Verifiable streaming computation and step-by-step zero-knowledge. *Cryptology ePrint Archive*. https://ia.cr/2025/251

Sheybani, N., Ahmed, A., Kinsy, M., & Koushanfar, F. (2025). Zero-knowledge proof frameworks: A systematic survey. *arXiv preprint arXiv:2502.07063*. https://doi.org/10.48550/arXiv.2502.07063

Liang, J., Hu, D., Wu, P., Yang, Y., Shen, Q., & Wu, Z. (2025). {SoK}: Understanding {zk-SNARKs}: The Gap Between Research and Practice. In *34th USENIX Security Symposium (USENIX Security 25)* (pp. 2085-2104). https://eprint.iacr.org/2025/172

Zhao, X., Xia, F., Xia, H., Mao, Y., & Chen, S. (2024). A zero-knowledge-proof-based anonymous and revocable scheme for cross-domain authentication. *Electronics*, *13*(14), 2730. https://doi.org/10.3390/electronics13142730

Wei, J., Chen, Y., Yang, X., Luo, Y., & Pei, X. (2025). A verifiable scheme for differential privacy based on zero-knowledge proofs. *Journal of King Saud University Computer and Information Sciences*, *37*(3), 14. https://doi.org/10.1007/s44443-025-00028-z

Tang, X., Shi, L., Wang, X., Charbonnet, K., Tang, S., & Sun, S. (2024). Zero-knowledge proof vulnerability analysis and security auditing. *Cryptology ePrint Archive*. https://eprint.iacr.org/2024/514?utm_source=chatgpt.com

Pailoor, S., Chen, Y., Wang, F., Rodríguez, C., & Van Geffen, J. (2023). *Automated Detection of Under-Constrained Circuits in Zero-Knowledge Proofs*. Proceedings of the ACM on Programming Languages (PACMPL).

Kiesel, J., & Heiss, J. (2025, September). Confidentiality-Preserving Verifiable Business Processes Through Zero-Knowledge Proofs. In *International Conference on Enterprise Design, Operations, and Computing* (pp. 119-136). Cham: Springer Nature Switzerland. https://doi.org/10.48550/arXiv.2509.20300