Automation, Control and Intelligent Systems
2024; 12(4): 108-113
http://www.sciencepublishinggroup.com/j/acis
doi: 10.11648/j.acis.20241204.12
ISSN: 2328-5583 (Print); ISSN: 2328-5591 (Online)

How to Improve the Security of Password Checkers

Tri Minh Ngo

The University of Danang - University of Science and Technology, Danang, Vietnam

Email address:
nmtri@dut.udn.com

To cite this article:
Tri Minh Ngo. (2024). How to Improve the Security of Password Checkers. *Automation, Control and Intelligent Systems, 12*(4), 108-113. https://doi.org/10.11648/j.acis.20241204.12

Received: 18 October 2024; **Accepted:** 2 December 2024; **Published:** 18 December, 2024

Abstract: After years of the attempt to replace password with other alternatives such as biometrics and smart cards, password is still the most pervasive user authentication mechanism. The password checking authentication is widely used for financial services, online social networks, and many other applications. This paper aims to analyze the security of a *password checker* qualitatively and quantitatively, and show how to improve it. *Qualitative* security analysis, in which it does not allow any information flow from secret date to public data, considers that the password checker is *not* a secure process. Therefore, an alternative analysis for the password checker is to analyze *quantitatively*, i.e., quantifying its information flow and determining how much secret information has been leaked. This method can be used to decide whether we can tolerate small leakages. A quantitative security analysis can be seen as a generalization of a qualitative one. To improve the security of the password checker, we propose a *noisy-output* policy, i.e., a situation where a system operator is able to add noise to the output: instead of always producing the exact outcomes, the system sometimes reports noisy outcomes. The noisy outcomes reduce the correlation between the output and the input, and thus reduce the leakage.

Keywords: Password Checker, Noisy-Output Policy, Quantitative Security Analysis

1. Introduction

The password checking authentication is widely used for login systems, banking systems, and many other real-world applications. In this paper, we analyze the security of a *password checker* qualitatively and quantitatively, and propose a method to improve it. Qualitative security analysis, such as *noninterference* [8] and *observational determinism* [9, 18] properties, is proposed for strict applications, in which they prohibit any information flow from a high security level to a low security level. For simplicity, throughout this paper, we consider a simple two-point security lattice, where the data is divided into two disjoint subsets, of private (high) and public (low) security levels, respectively. For many real-world applications, the leakage is unavoidable. A typical example is a password checker where an attacker (user) tries a string to guess the password [3, 7, 16]. Even when the attacker makes a wrong guess, secret information has been leaked, i.e., it reveals information about what the real password is *not*. Thus, despite the correct functioning, the password checker is rejected by qualitative security standard.

Therefore, an alternative approach for such applications is to consider quantitative security analysis, i.e., quantifying the information flow and determining how much secret information has been leaked [4]. This method can be used to decide whether we can tolerate *minor* leakages. A quantitative security theory can be seen as a generalization of a qualitative one. Quantifying information flow also provides a way to judge whether an application leaks more information than another, in case both are insecure.

1.1. Quantitative Security Analysis

Quantitative theory sees a program as a channel in the information-theoretic sense, where the secret S is the input and the observable final outcomes O are the output [3]. An attacker, by observing O, might be able to derive information about S.

The quantitative analysis of information flow then analyzes the amount of private data that an attacker might learn. The analysis is based on the notion of *entropy*. The entropy of a random private variable expresses the *uncertainty* of an attacker about its value, i.e., how *difficult* it is for an attacker to discover its value.

The leakage of a program is typically defined as the difference between the secret's initial uncertainty, i.e., the uncertainty of the attacker about the private data before observing the program's public outcomes, and the secret's remaining uncertainty, i.e., the uncertainty of the attacker after observing the outcomes, i.e.,

Information leakage = Initial uncertainty - Remaining uncertainty.

The literature argues that observable outcomes would reduce the initial uncertainty of the attacker on the secret; and thus, cause the leakage of the system.

This paper presents an idea that is to enhance the security of the password checkers. The system operator adds noise to the response of the system, i.e., instead of always producing the exact outcomes, the program might sometimes report a noisy one. The noisy-output policy makes the outcomes of program more random, and thus, it reduces the correlation between the output and the input. As a consequence, it increases the remaining uncertainty, and the value of leakage is reduced.

1.2. Contributions

We discuss how to enhance the security of a system by adding noise to its output. We apply this idea to the situation of a password checker, which is popular in many real-world applications. We show quantitatively that this will reduce the value of leakage.

1.3. Organization of the Paper

Section 2 presents the preliminaries, then, Section 3 discusses the classical quantitative security analysis for a password checker. Section 4 discusses the idea of adding noise to the ouput, while Section 5 proves that the noisy output reduce the leakage of the system quantitatively. Section 6 discusses how noisy-output policy can be applied to real-world applications. Section 7 discusses related work, and finally, Section 8 concludes the paper.

2. Preliminaries

2.1. Probabilistic Distribution

Let X be a discrete random variable with the carrier $\mathbf{X} = \{x_1, \ldots, x_n\}$. A *probability distribution* π over a set $\mathbf{X}$ is a function $\pi : \mathbf{X} \to [0, 1]$, such that the sum of the probabilities of all elements over $\mathbf{X}$ is 1, i.e., $\sum_{x_i \in \mathbf{X}} \pi(x_i) = 1$. If $\mathbf{X}$ is uncountable, then $\sum_{x_i \in \mathbf{X}} \pi(x_i) = 1$ implies that $\pi(x_i) > 0$ for countably many $x_i \in \mathbf{X}$. The probabilistic behavior of X is then simply given by probabilities $p(X = x_i) = \pi(x_i)$.

When X is clear from the context, we use the notation $\pi = \{p(x_1), p(x_2), \ldots, p(x_n)\}$ to denote the probabilities of elements in $\mathbf{X}$, i.e., $p(X = x_i)$.

2.2. Shannon Entropy

Definition 2.1. The Shannon entropy of a random variable X is defined as [3],

$$\mathcal{H}(X) = -\sum_{x \in \mathbf{X}} p(x) \log p(x),$$

where the logarithm is to the base 2.

Definition 2.2. The conditional Shannon entropy of a random variable X given Y is [3],

$$\mathcal{H}(X|Y) = \sum_{y \in \mathbf{Y}} p(y) \mathcal{H}(X|Y = y),$$

where $\mathcal{H}(X|Y = y) = -\sum_{x \in \mathbf{X}} p(x|y) \log p(x|y)$.

It is possible to prove that $0 \leq \mathcal{H}(X|Y) \leq \mathcal{H}(X)$. The minimum value of $\mathcal{H}(X|Y)$ is 0, if X is completely determined by Y. The maximum value of $\mathcal{H}(X|Y)$ is $\mathcal{H}(X)$, when X and Y are independent.

2.3. Information-theoretic Channel

The quantitative security analysis in the information-theoretic sense models the system as a channel with the secret as the input and the observables as the output. Formally, an information-theoretic channel is a triple $(\mathbf{X}, \mathbf{Y}, M)$, where $\mathbf{X}$ represents a finite set of secret inputs, $\mathbf{Y}$ represents a finite set of observable outputs, and M is a $|\mathbf{X}| \times |\mathbf{Y}|$ channel matrix which contains the conditional probabilities $p(y|x)$ for each $x \in \mathbf{X}$ and $y \in \mathbf{Y}$. Thus, each entry of M is a real number between 0 and 1, and each row sums to 1.

2.4. Basic Settings for the Analysis

First, we denote the high security input as S and the low security input as L, i.e., the string to guess the password. Since the high security output is irrelevant, programs only give a low security outcome O. Our goal is to quantify how much information about S is deduced by observing O. We also assume that the sets of possible values of data are finite, as in the traditional approaches.

Secondly, for generalization, we assume that there is an *uniform* probability distribution on the high values.

Notice that these restrictions aim to demonstrate our core idea. However, the analysis might be adapted to more complex situations easily after some trivial modifications.

3. Quantitative Security Analysis for a Password Checker

First, we introduce the model of quantitative security analysis, and how to apply it to a typical example of password checker.

3.1. *The Model of Quantitative Security Analysis*

Classical works [5, 6, 12–14, 19] use information theory to analyze information flow quantitatively. A program is seen as a standard channel with S as the *private* input (high security) and O as the *public* output (low security). Let $\mathcal{H}(S)$ denote the uncertainty of the attacker on the secret before executing the program, and $\mathcal{H}(S|O)$ the uncertainty after the program has been executed and public outcomes are observed, where $\mathcal{H}$ is the Shannon entropy. The leakage of the program is defined as $\mathcal{L}(P) = \mathcal{H}(S) - \mathcal{H}(S|O)$, where $\mathcal{L}(P)$ denotes the leakage of P.

3.2. *A Case Study of Password Checker*

Consider the following password checker. Let S denote the private password, L the string entered by the attacker, and O the public answer,

$$\texttt{if } (S = L) \quad \texttt{then } O := 1 \texttt{ else } O := 0.$$

Assume that S might be A_1, A_2, A_3, or A_4, with $\pi = \{p(A_1) = p(A_2) = p(A_3) = p(A_4) = 0.25\}$. Thus, $\mathcal{H}(S) = -4 \log 0.25 = 2$.

Without loss of generality, we assume that $L = A_1$. When $L = A_1$, the password checker is modeled by the following channel M,

M	$O = 1$	$O = 0$
$S = A_1$	1	0
$S = A_2$	0	1
$S = A_3$	0	1
$S = A_4$	0	1

The channel M and the distribution π determine the joint probability matrix J, where $J[s, o] = \pi(s) \cdot M[s, o]$.

J	$O = 1$	$O = 0$
$S = A_1$	0.25	0
$S = A_2$	0	0.25
$S = A_3$	0	0.25
$S = A_4$	0	0.25

The joint probability matrix J determines a marginal distribution of O, i.e., $p(o) = \sum_{\forall s} J[s, o]$. Thus, $p(O = 1) = 0.25$ and $p(O = 0) = 0.75$. Since $p(S = s|O = o) = \frac{J[s,o]}{p(o)}$, then $p(S = A_1|O = 1) = 1$, $p(S = A_2|O = 1) = p(S = A_3|O = 1) = p(S = A_4|O = 1) = 0$, and $p(S = A_1|O = 0) = 0$, $p(S = A_2|O = 0) = p(S = A_3|O = 0) = p(S = A_4|O = 0) = 0.333$. Thus,

$$\mathcal{H}(S|O) = -0.25(1 \log 1) - 0.75(3 \cdot 0.333 \log 0.333) = 1.189.$$

Hence, $\mathcal{L}(P, \pi) = 2 - 1.189 = 0.811$.

4. Adding Noise to the Output

To improve the security of the password checker, we propose a situation in which the system operator adds noise in a controlled way to the output, i.e., to mask the contribution of the data while still preserving the overall accuracy of the system. The noisy outcomes might mislead the attacker's belief about the secret, i.e., they increase the final uncertainty. As a consequence, the value of leakage is decreased. This idea is illustrated in details below.

Password checker with noisy outcomes: Consider the previous password checker. We assume that the system operator *secretly* changed its behavior, i.e., the real password checker is a probabilistic password checker where the system operator introduced some perturbation mechanism to the output (We assume that the attacker does not know about the security policy applied to the system.),

$$\texttt{if } (S = L) \texttt{ then } \{O := 1 \;{}_{0.9}\| \; O := 0\}$$

$$\texttt{else } \{O := 0 \;{}_{0.9}\| \; O := 1\}.$$

In this version, the exact answers are reported with probability 0.9, e.g., when $S = L$, $O = 1$ is reported with probability 0.9, and $O = 0$ with probability 0.1. Therefore, the real channel M' is given as follows,

M'	$O = 1$	$O = 0$
$S = A_1$	0.9	0.1
$S = A_2$	0.1	0.9
$S = A_3$	0.1	0.9
$S = A_4$	0.1	0.9

Notice that the attacker still thinks that the system is M, but in fact, the real system is M'. Similarly, based on π and M', we determine the joint probability matrix J' as follows:

J'	$O = 1$	$O = 0$
$S = A_1$	0.225	0.025
$S = A_2$	0.025	0.225
$S = A_3$	0.025	0.225
$S = A_4$	0.025	0.225

Thus, the computation gives the real distribution $p(O = 1) = 0.3$ and $p(O = 0) = 0.7$. The real posteriori probabilities are $p(S = A_1|O = 1) = 0.75$, $p(S = A_2|O = 1) = p(S = A_3|O = 1) = p(S = A_4|O = 1) = 0.083$, and $p(S = A_1|O = 0) = 0.37$, $p(S = A_2|O = 0) = p(S = A_3|O = 0) = p(S = A_4|O = 0) = 0.321$. Therefore, $\mathcal{H}(S|O) = 1.84$. Hence, $\mathcal{L}'(P, \pi) = 2 - 1.84 = 0.16$. This result shows that the leakage has been decreased.

5. Password Checker's Leakage Without/With Noisy Output

5.1. *Without Noisy Output*

In general, we assume that S might be A_1, A_2, A_3,..., or A_n, with $\pi = \{p(A_1) = p(A_2) = p(A_3) = \cdots = p(A_n) = \frac{1}{n}\}$. Similarly, without loss of generality, let assume that the attacker chooses $L = A_1$. When $L = A_1$, the password checker is modeled by the following channel M,

M	$O = 1$	$O = 0$
$S = A_1$	1	0
$S = A_2$	0	1
$S = A_3$	0	1
$\vdots$	$\vdots$	$\vdots$
$S = A_n$	0	1

Therefore,

$$\mathcal{L}(P, \pi) = \log(n) - \left(\frac{n-1}{n}\right) \cdot \log(n-1).$$

5.2. With Noisy Output

The noisy outcomes change the behavior of the system, i.e., they change the channel M that models the system (the public channel that the attacker knows) to M' (the real channel in secret). The noisy outcomes should be added in such a way that they change the original channel, but still preserve a certain level of *reliability*, e.g., the above password checker works properly in 90% of the time. Here, we define the reliability of a system as the probability that a system will perform its function correctly during a specified period of observation time. Totally random outcomes might achieve the best confidentiality, but with these outcomes, the system is practically useless.

With noisy output, the password checker is modeled by the following channel M', where $\mathcal{R}$ is the reliability of the systems.

M'	$O = 1$	$O = 0$
$S = A_1$	$\mathcal{R}$	$1 - \mathcal{R}$
$S = A_2$	$1 - \mathcal{R}$	$\mathcal{R}$
$S = A_3$	$1 - \mathcal{R}$	$\mathcal{R}$
$\vdots$	$\vdots$	$\vdots$
$S = A_n$	$1 - \mathcal{R}$	$\mathcal{R}$

Therefore, $\mathcal{L}'(P, \pi) = \log(n) - \mathcal{H}(S|O)$, where $\mathcal{H}(S|O)$ is given by:

$$\begin{aligned}\mathcal{H}(\mathcal{S}|\mathcal{O}) = \; & -\left(\tfrac{n-1}{n} + \tfrac{2-n}{n}\mathcal{R}\right) \cdot \left(\tfrac{\mathcal{R}}{(n-1)+(2-n)\mathcal{R}} \log \tfrac{\mathcal{R}}{(n-1)+(2-n)\mathcal{R}} + (n-1) \cdot \tfrac{1-\mathcal{R}}{(n-1)+(2-n)\mathcal{R}} \log \tfrac{1-\mathcal{R}}{(n-1)+(2-n)\mathcal{R}}\right) \\ & -\left(\tfrac{1}{n} + \tfrac{n-2}{n}\mathcal{R}\right) \cdot \left(\tfrac{1-\mathcal{R}}{1+(n-2)\mathcal{R}} \log \tfrac{1-\mathcal{R}}{1+(n-2)\mathcal{R}} + (n-1) \cdot \tfrac{\mathcal{R}}{1+(n-2)\mathcal{R}} \log \tfrac{\mathcal{R}}{1+(n-2)\mathcal{R}}\right).\end{aligned}$$

Now, we need to show that, with noisy output, the leakage will be reduced. The common idea is that the observation of the program's public outcomes would enhance the attacker's knowledge about the private data. With noisy output, the attacker's knowledge is misled, and thus, the leakage will be decreased.

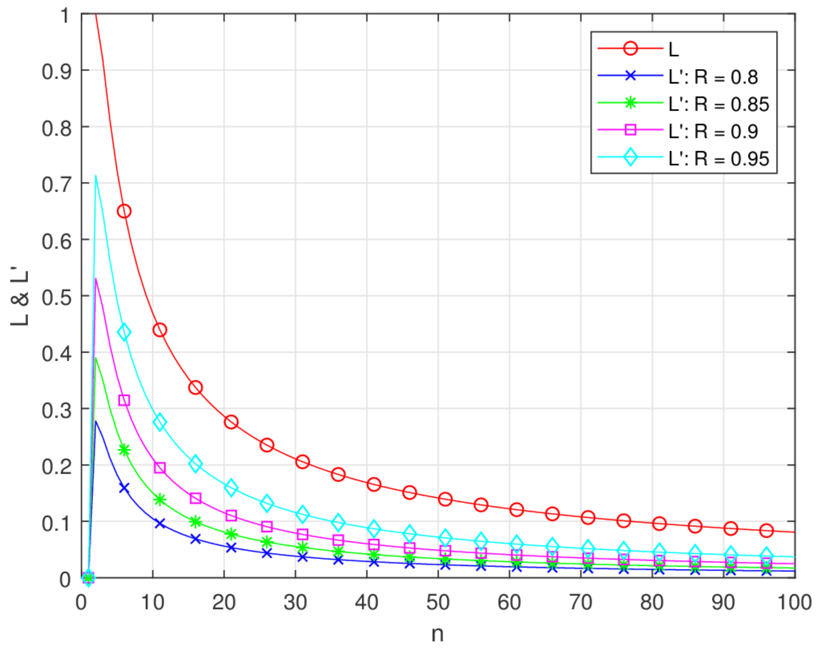

Figure 1. *Leakage Values.*

The Figure 1 shows that with noisy output, the leakage of password checker is always not greater than the one without noise in the outcomes. As we expected, the higher value of reliability we want to ensure, the higher value of leakage the attacker might get. When $\mathcal{R} = 1$, it is trivial that M and M' are identical, and thus, their values of leakage are the same.

6. Real-world Applications

Here it raises a question of how this noisy policy can be applied to real-world applications, when it changes the behavior of a system. A noisy-output policy enhances the security, but it also reduces the reliability of a system, i.e., the system does not always work in a proper way. Therefore, firstly, we require that a noisy-output should guarantee at least a certain level of reliability. Besides, secondly, we discuss how the drawback of the reduced reliability can be overcome.

To make it clearer, we analyze the case when an attacker try the correct password to the system. Without the noisy-output policy, the system always accepts the correct password. With the policy, the system might reject it; and thus, make the system more secure in case the correct password is applied by the attacker.

In case the system rejects the password, we need to distinguish the different situations between the user and the attacker. If the system rejects the password applied by the user: since the user knows the policy, then he would try it again. Notice that with a very high probability, the system work properly. Therefore, the system will accept the password on the second try.

On the contrary, the attacker does not know the policy. Therefore, when the correct password is rejected, he will try another password; and with a very high probability, the system would also reject it. For applications such as login systems or banking systems, we can allow the user/attacker try three times before locking him out.

Notice again that to make a system more secure with the noisy-output policy, we have to accept that in some cases, the system accepts a wrong password. This is the reason that the policy should guarantee a high reliability. Notice that in this situation, the system accepts the login, but no private information is leaked, since the attacker still does not know the correct password. Thus, in the next login, there is a high chance that the system will reject this wrong password.

This situation that the system accepts a wrong password can be avoided by modifying the probabilistic password checker as follows, where noise is only added to the selected output.

$$\texttt{if}\ (S = L)\ \texttt{then}\ \{O := 1\ {}_{0.9}\| \ O := 0\}\ \texttt{else}\ \{O := 0\}.$$

In this case, $\mathcal{H}(S|O) = 1.35$, and thus, $\mathcal{L}'(P, \pi) = 2 - 1.35 = 0.65$. This leakage is smaller than the case without noise, but greater than the situation where noise is added to all outputs.

Besides, to guarantee a higher reliability, the system might also implement two-factor authentication, i.e., in addition to asking for something that only the user knows (e.g., username, password, personal identification number), the system also requires something that only the user has (e.g., the bank card, the smart card). The automated teller machine scenario illustrates the basic concept of two-factor authentication systems, i.e., without the combination of both card and personal identification number verification, authentication does not succeed.

7. Related Work

The idea of adding noise to the output comes from the *differential privacy* control. Differential privacy is a mathematical framework for ensuring the privacy of individuals in the datasets. It can provide a strong guarantee of privacy by allowing data to be analyzed without revealing sensitive information about any individual in the datasets. The differential privacy control is the problem of protecting the privacy of database's participants when performing *statistical* queries [1–3, 10]. The differential privacy control uses some output perturbation mechanism to report a noisy answer among the correct ones for the queries. Thus, while the attacker is still able to learn properties of the population as a whole, he cannot learn the value of an individual.

In [11], Köpf et al. also explore a similar idea to cope with timing attacks for cryptosystems, i.e., randomizing each cipher-text before decryption. As a consequence, the strength of the security guarantee is enhanced, while the efficiency of the cryptosystem is decreased, since the execution time of the cryptographic algorithm is increased.

The idea of adding noise is also popular in the field of universal filtered multicarrier. In [17], authors introduce artificial noise generated by hyperchaotic system to pseudo expand the distribution of quantum noise, which will make it more difficult for eavesdroppers to crack plaintext information violently. In [15], we discuss the noisy-output policy in the *one-try guessing model*, i.e., observing the public outcomes, the attacker is allowed to guess the value of S by only one try. For the password checker, this one-try guessing model can be understood as that an attacker is allowed to try a string to the system only once. If the entered string is not the correct password, the system will block the account. However, we realize that this model of attack is not suitable in real-life situations. In this paper, we consider the *multiple-try* attack model, i.e., the attacker is able to guess the password by many tries. This is a more practical situation, where we quantify information flow with Shannon entropy, instead of min-entropy.

8. Conclusions

This paper proposes a method to decrease the leakage of the password checker by adding noise to the outcomes, i.e., the noise misleads the attacker's belief about the secret, and thus, it increases the final uncertainty. This paper also discusses how to apply this to real-world applications.

ORCID

0009-0003-8254-7201 (Tri Minh Ngo)

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. S. Alvim and M. E. Andrés. On the relation between differential privacy and quantitative information flow. In Proceedings of the 38th international conference on Automata, languages and programming - Volume Part II, ICALP'11, pages 60-76. Springer-Verlag, 2011.

[2] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi. Differential privacy: on the trade-off between utility and information leakage. CoRR, abs/1103.5188, 2011.

[3] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, and C. Palamidessi. Quantitative information flow and applications to differential privacy. In A. Aldini and R. Gorrieri, editors, Foundations of security analysis and design VI, pages 211-230. Springer-Verlag, 2011.

[4] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith. The Science of Quantitative Information Flow. Information Security and Cryptography. Springer, Springer Nature, United States, 2020.

[5] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. In Proceedings of the 2nd international conference on Trustworthy global computing, TGC'06, pages 281-300. Springer-Verlag, 2007.

[6] D. Clark, S. Hunt, and P. Malacaria. Quantitative information flow, relations and polymorphic types. J. Log. and Comput., 15: 181-199, 2005.

[7] M. R. Clarkson, A. C. Myers, and F. B. Schneider. Belief in information flow. In In Proc. 18th IEEE Computer Security Foundations Workshop, pages 31-45, 2005.

[8] J. A. Goguen and J. Meseguer. Security policies and security models. In IEEE Symposium on Security and Privacy, pages 11-20, 1982.

[9] M. Huisman and T. M. Ngo. Scheduler-specific confidentiality for multi-threaded programs and its logic-based verification. In Proceedings of the 2011 international conference on Formal Verification of Object-Oriented Software, FoVeOOS'11, pages 178-195. Springer-Verlag, 2012.

[10] S. Jiao, L. Cai, X. Wang, K. Cheng, and X. Gao. A differential privacy federated learning scheme based on adaptive gaussian noise. CMES - Computer Modeling in Engineering and Sciences, 138(2): 1679-1694, 2023.

[11] B. Köpf and M. Dürmuth. A provably secure and efficient countermeasure against timing attacks. In Proceedings of the 2009 22Nd IEEE Computer Security Foundations Symposium, CSF'09, pages 324-335. IEEE Computer Society, 2009.

[12] P. Malacaria. Risk assessment of security threats for looping constructs. J. Comput. Secur., 18: 191-228, 2010.

[13] P. Malacaria and H. Chen. Lagrange multipliers and maximum information leakage in different observational models. In Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security, PLAS'08, pages 135-146. ACM, 2008.

[14] I.S. Moskowitz, R. E. Newman, D.P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. In Proceedings of the 2003 ACM workshop on Privacy in the electronic society, WPES'03, pages 79-88. ACM, 2003.

[15] T. M. Ngo and M. Huisman. Quantitative security analysis for programs with low input and noisy output. In Proceedings of the 6th international conference on Engineering Secure Software and Systems, ESSoS'14, pages 77-94. Springer-Verlag, 2014.

[16] G. Smith. On the foundations of quantitative information flow. In Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures, FOSSACS'09, pages 288- 302. Springer-Verlag, 2009.

[17] Y. Xu, M. Gao, Y., B. Chen, and W. Shao. Diffusionassisted quantum noise stream cipher for physical layer security in ufmc. Optics and Laser Technology, 171: 110407, 2024.

[18] S. Zdancewic and A. C. Myers. Observational determinism for concurrent program security. In Proceedings of 16th IEEE Computer Security Foundations Workshop, CSFW'03, pages 29-43. IEEE Computer Society, 2000.

[19] Y. Zhu and R. Bettati. Anonymity vs. information leakage in anonymity systems. In Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS'05, pages 514-524. IEEE Computer Society, 2005.