報告番号	※甲　　第　　号

主　論　文　の　要　旨

論文題目　Towards Practically Applicable Quantitative Information Flow Analysis
（実用化を目指した量的情報流解析に関する研究）

氏　　名　CHU Bao Trung

論　文　内　容　の　要　旨

Quantitative Information Flow (QIF) was introduced as a formal notion of the information leakage of a confidential input of a program that can be obtained by analyzing a publicly observable output. A program of which QIF is under a predefined threshold is regarded as safe even if the program does not have the noninterference property. Intuitively, QIF of a program is defined as the difference between the *initial uncertainty* and the *remaining uncertainty*, which are the uncertainties about confidential input values before and after executing that program, respectively. There are several formal definitions which elaborate on that intuition, namely Shannon entropy and Min-entropy based notions and belief-based notion. These QIF definitions and the traditional methods of computing QIF are reviewed in Chapter 2. However, all these notions fail to represent reasonable quantitites of the leakage when there are some output values, which reveal much more information about the confidential input than others, because those definitions take the average leakage on all possible output values. The contribution of this doctoral thesis includes:

• a framework consists of recognizable, algebraic and tree power series that helps answer: Is there any method of counting models that directly analyzes variables with complex structures as what they are: strings, arrays, linked lists, trees and so on?.

• an attempt to answer: “How can we address reasonably and naturally the leakage of secure information when executing programs through a specific output value?”, by proposing two new notions for dynamic leakage, the dynamic version of QIF.

• a method to accelerate the calculation of dynamic leakage in “divide and conquer” fashion by answering: “How to calculate the dynamic leakage in executing a program P when the dynamic leakages of the sub-programs of P are known?”.

Addressing the lack of calculation model for the information leakage when observing a specific output value, in the first part of Chapter 3, we propose two notions called QIF_1 and QIF_2. Both of QIF_1 and QIF_2 are notions of dynamic leakage metric while the ordinary QIF

is a static one which is independent of a specific output value. There are cases in which QIF_2 gives more reasonable quantity than QIF_1 while calculating QIF_1 is generally easier than QIF_2. We define the problems of computing QIF_1 and QIF_2 for Boolean programs and investigate the computational complexity of these problems. Even for *deterministic loop-free* programs, where QIF_1 coincides with QIF_2, the problem is proved to be #P-hard, meaning that the problem is not easier than the #SAT problem, which is the counting version of the well-known SAT problem. This implies that, though many off-the-shelf model counting tools are available, there is a need to speed up calculation of dynamic leakage. We conducted experiments on 14 programs when utilizing different model counting tools. All the experiments were carried out under the assumption that the program under analysis (PUA) is deterministic and has uniformly distributed input. Those experiments are to reaffirm the model counting-based calculation, but the method does not scale up well.

In the second part of Chapter 3, we present an attempt to improve the performance of quantifying dynamic leakage. The idea is to decompose a given PUA to sub-programs so that leakage of the original program is calculated from the sub-programs. We propose two such types: *value domain-based composition* and *sequential composition*. The former is suitable for utilizing parallel computing to speed up calculation of dynamic leakage. *Value domain-based composition*, however, does not help much when the PUA is too big to analyze as a whole, or is constituted from distributed parts, because the sizes and complexities of value domain-based decomposed sub-programs are almost the same to the PUA. In such cases, sequential composition helps by making much simpler sub-programs of smaller sizes than the original PUA. Thus, those two types of composition can also be combined to produce a better performance. For example, we first horizontally decompose the PUA into sub-programs using the sequential composition, and then vertically decompose the value domains of sub-programs. We experimented three simple benchmarks to confirm the feasibility of our methods. It is shown that parallel computing accelerates the model counting 25 times as fast as single-threaded computation.

In Chapter 4, we present our attempt to speed up the model counting of string constraints which contributes to QIF analysis for programs with strings. Existing string counting methods use finite automata to represent string constraints and the precision of approximated counting are low for recursive constraints. The proposed method uses Context-Free Grammars (CFGs) to represent string constraints. Similarly, constraints on the shape of trees can be represented using recognizable tree series. An experiment was conducted to compare the performance of the proposed method with ABC, the state-of-the-art automaton-based string counting tool, on 17,544 files of Kaluza benchmark. Our prototype provided the same answers to ABC on 17,183 files, different but exact or better answers on other 370 files, where the execution time of the prototype and ABC are almost the same on all the cases. These experimental results show the effectiveness of the proposed method of QIF calculation based on CFGs.

Keywords: quantitative information flow, dynamic leakage, string constraint, model c ounting, compositional reasoning, formal language theory, information theory