情報流解析における制約付き機密度パラメータ

Bounded Secrecy Parameters in Information Flow Analysis

桑原 寛明[*] **國枝 義敏**[†]

Summary. This paper proposes bounded secrecy parameters in information flow analysis. Although secrecy parameters make it possible to define classes or functions without specifying a concrete secrecy for each data, programs that include secrecy parameters are required to satisfy noninterference with any substitution for secrecy parameters. Bounded secrecy parameters relax this too restrictive requirement and make more programs typable. We show a type system for information flow analysis of imperative programs with bounded secrecy parameters and a simple example of type checking.

1 はじめに

プログラムの静的解析により機密情報が外部に漏れないことを検査する手法として，型検査に基づく情報流解析が提案されている [1] [2] [3] [4]．型検査に基づく情報流解析では，データの機密度を型として利用し，型付け可能なプログラムが非干渉性を満たすように型システムを構築する．非干渉性は，機密度の低いデータが機密度の高いデータに直接および間接的に依存しないことを表し，機密データ自体に加え機密データを推測できる情報も漏らさないという意味でよい性質である．

型検査に基づく情報流解析では，プログラム中の変数や関数の返り値の型として機密度を指定する．変数の機密度は，その変数に格納可能なデータの機密度の上限を表す．型として機密度を指定する際，一般には具体的な機密度を指定する必要がある．そのため，扱うデータの機密度を事前に決定できない汎用的なコレクションフレームワークのようなプログラムの場合，指定する機密度を変えながら同じようなプログラムを記述することになる．これは，出現する機密度のみが異なるクローンを多数作り出すことに相当するため望ましくない．

この問題に対し，機密度のパラメータ化が提案されている [5]．この手法では，Java 言語におけるジェネリックなクラスのように，機密度を表す機密度パラメータを用いてクラスを定義し，機密度パラメータに具体的な機密度を割り当ててクラスのインスタンスを生成する．これにより，機密度のみが異なるクローンを多数記述する必要はない．[5] では，機密度パラメータに対応した情報流解析のための型システムも提案されており，型付け可能なプログラムは機密度パラメータにどのような機密度を割り当てても非干渉性を満たすことが保証されている．

しかし，任意の機密度の割り当てに対して非干渉性を満たす必要があることは強い制限となっている．例えば，2 段階の機密度のもとで 2 種類の機密度パラメータを含むプログラムを考えると，機密度の割り当て方は 4 通りあり，そのすべてについて非干渉性を満たすことが求められる．この制限を満たしながら実用的なプログラムを作成することは容易ではない．非干渉性を満たさない機密度の割り当て方が残ることは許容しつつ，そのような割り当て方が実際に行われていればそれを検出する方が現実的である．

本稿では，情報流解析における制約付き機密度パラメータを提案し，対応する型システムを構築する．非干渉性を満たす機密度の割り当てと満たさない割り当てを区別するために，機密度パラメータに対して制約を与える．制約を満たす機密度の割り当てに対して非干渉性を満たせばよいとする．さらに，機密度の具体的な割り当て方が制約を満たしているか検査する．以上により，従来手法における制限が緩

[*]Hiroaki Kuwabara, 南山大学情報センター

[†]Yoshitoshi Kunieda, 立命館大学情報理工学部

和されプログラムが作成しやすくなることが期待される.

2 制約付き機密度パラメータ

型検査に基づく情報流解析では，プログラム中の変数や関数の返り値の型として機密度を指定する．例えば，機密度が L と H の 2 段階で L より H の方が高いとすると，2 つの引数の値が等しいか判定する関数の 1 つは

```
L equalsLL(L x1, L x2) { return x1 == x2; }
```

のように定義できる．この関数は，2 つの引数の機密度がともに L と指定されており，返り値の機密度も L となっている．機密度が 2 種類あるため，この他に

```
H equalsLH(L x1, H x2) { return x1 == x2; }
H equalsHH(H x1, H x2) { return x1 == x2; }
```

の 2 つの関数も必要である [1]．返り値の機密度は引数の機密度に依存するため，いずれの関数も返り値の機密度は H である．これら 3 つの関数は引数と返り値の機密度は異なるが，いずれも本体は同じコードで実装されており，一種のコードクローンとなっている.

これら 3 つの関数は，2 つの引数と返り値の機密度それぞれに機密度パラメータ X1，X2，Y を用いると

```
Y equals(X1 x1, X2 x2) { return x1 == x2; }
```

のように定義できる．3 つの機密度パラメータすべてに L を割り当てれば equalsLL に相当する関数が得られる．機密度の割り当て方は全部で 8 通りあるが，X1，X2 の一方か双方に H を割り当て，Y に L を割り当てた場合，この関数定義は非干渉性を満たさない．非干渉性を満たさない機密度の割り当て方が存在するため，従来手法では型付け不能である．2 つの変数 [2]が依存関係にあれば，それらの機密度の間には順序が存在するが，このような変数の機密度をともに機密度パラメータで表した場合には順序を守らないような機密度の割り当て方が存在し得る.

そこで，機密度間の順序を機密度パラメータ間に反映するために，機密度パラメータに制約を与えられるように拡張する．例えば，equals 関数の返り値は 2 つの引数に依存していることから，返り値の機密度は x1 の機密度以上かつ x2 の機密度以上でなければならないため，

```
<X1⊑Y, X2⊑Y> Y equals(X1 x1, X2 x2) { return x1 == x2; }
```

のように定義する．ここで，X1 ⊑ Y は Y に割り当てられる機密度は X1 に割り当てられる機密度以上でなければならないことを表す．機密度パラメータに対する制約を満たす割り当てについてのみ非干渉性が満たされればよいとすれば，X1，X2 に H，Y に L を割り当てるような非干渉性を満たさない割り当ては制約を満たさないため除外できる.

3 型システム

3.1 対象言語

本稿では [6] の簡単な手続き型言語の変種を対象に，関数定義に制約付き機密度パラメータを導入する．機密度パラメータを提案した [5] では，クラスベースのオブジェクト指向言語を対象としてクラス定義に機密度パラメータを導入しているが，本稿では制約付き機密度パラメータのみに着目するため，仕様の簡潔な手続き型言語を対象とする．簡単のために機密度定数は L と H の 2 種類とし，$L \sqsubseteq H$ を満たす機密度束 $(\{L, H\}, \sqsubseteq)$ を仮定する．機密度定数と機密度パラメータをまとめて機密度と呼ぶ.

対象言語の文法を図 1 に示す．η は機密度定数，X は機密度パラメータを表し，機

[1] equalsLH(L,H) と equalsHL(H,L) は一方で他方を置換可能なので一方のみでよい.

[2] return は返り値を表す特殊な変数への代入とみなす.

$$
\begin{array}{rl}
\eta & ::= L \mid H \\
\tau & ::= \eta \mid X \\
\rho & ::= \tau \mid \rho \bowtie \rho \mid \rho \bowtie \rho \\
P & ::= \overline{F} \\
F & ::= \langle \Gamma \rangle\ \tau\ f(\overline{\tau\ x})\ B \\
\Gamma & ::= \overline{\tau \sqsubseteq \tau} \\
B & ::= \{\overline{\tau\ x;}\ \overline{S;}\} \\
S & ::= x = e \mid \text{if } (e)\ B \text{ else } B \\
e & ::= x \mid \text{true} \mid \text{false} \mid e \odot e \mid f[\overline{X \mapsto \tau}](\overline{e})
\end{array}
$$

図 1　対象言語の文法

密度 τ は機密度定数か機密度パラメータのいずれかである．データ型を bool 型のみに限定して記述を省略し，変数や関数の返り値の型には機密度のみを記述する．$\bowtie$ は機密度パラメータに拡張された結び演算子であり，$\rho_1 = \eta_1, \rho_2 = \eta_2$ ならば $\rho_1 \bowtie \rho_2 = \eta_1 \sqcup \eta_2$，$\rho_1 = \rho_2$ ならば $\rho_1 \bowtie \rho_2 = \rho_1$ と定義する．$\bowtie$ は拡張された交わり演算子であり同様に定義される．ρ はプログラム中には出現しないが，型付け規則の中に出現する．プログラム P は関数定義の並びである．$\overline{A}$ は長さ 0 以上の有限リストを表す略記である．F は関数定義であり，f が関数名を表す．Γ は F 中に出現する機密度パラメータに対する制約のリストである．F は制約がなく Γ に出現しない機密度パラメータを含む場合もある．機密度パラメータのスコープは F 内に限られる．B はブロック，S は文，e は式である．ブロックの先頭でローカル変数を宣言可能である．x は変数，true と false は真偽値リテラルである．関数の返り値は予約変数 *result* への代入によって設定する．$\odot$ は適当な 2 項演算を指す．$f[\overline{X \mapsto \tau}](\overline{e})$ は関数 f を呼び出す式であるが，f の定義中に出現する機密度パラメータ X_i に機密度 τ_i を割り当てて呼び出すことを表す．

3.2　型付け規則

型付け規則を図 2 に示す．PROGRAM 規則がプログラム全体，FDEC 規則が関数定義，BLOCK 規則がブロック，ASSIGN, IF, SUB-S 規則が文，その他が式の型付け規則である．Δ は型環境であり，変数名からその機密度への関数である．Γ は機密度パラメータに対する制約のリスト，*result* は関数の返り値を表す変数，*ftype* は関数のシグネチャを取得する関数である．関数のシグネチャは機密度パラメータに対する制約のリストも含む．$\mathrm{sat}(\Gamma)$ は，Γ 中のすべての制約を同時に満たすように Γ 中の各機密度パラメータに機密度定数を割り当てられること，すなわち Γ が充足可能であることを表す．機密度パラメータに対する制約 c に対し，$\Gamma \triangleright c$ は図 3 の規則に従って Γ から c を導出できることを表す．$\Gamma \triangleright c$ ならば，Γ を充足するような割り当ての下で c は満たされる．割り当て $\sigma = [\ldots, X \mapsto \tau, \ldots]$ に対し，$\eta\sigma = \eta$, $X\sigma = \tau$, $Y\sigma = Y$ (ただし $Y \neq X$)，$(\tau_1 \sqsubseteq \tau_2)\sigma = \tau_1\sigma \sqsubseteq \tau_2\sigma$，$\Gamma\sigma = \overline{(\tau_1 \sqsubseteq \tau_2)\sigma}$ (ただし $\Gamma = \overline{\tau_1 \sqsubseteq \tau_2}$) とする．

プログラムを構成するすべての関数定義が型付け可能であればプログラムは型付け可能である．引数と返り値に関する型環境および機密度パラメータに対する制約リストの下で関数本体が型付け可能，かつ制約リストが充足可能であれば関数定義は型付け可能である．プログラムと関数定義は型付けできるか否かが重要であり，プログラムと関数定義の具体的な型は定義しない．ブロックの型判定式 $\Delta;\Gamma \vdash B : \rho$ および文の型判定式 $\Delta;\Gamma \vdash S : \rho$ は，それぞれ型環境 Δ と制約リスト Γ の下で B あるいは S の実行によって変更される変数の機密度が ρ 以上であることを表す．式の型判定式 $\Delta;\Gamma \vdash e : \rho$ は，型環境 Δ と制約リスト Γ の下で式 e の機密度が ρ 以下であることを表す．

いくつかの型付け規則には条件が存在する．例えば ASSIGN 規則の $\Gamma \triangleright \rho_e \sqsubseteq \tau_x$

$$\frac{\vdash F_i \quad i \in \{1,\ldots,n\}}{\vdash F_1 \ \ldots \ F_n}\ [\text{PROGRAM}] \qquad \frac{\overline{x : \tau_x}, \mathit{result} : \tau_r; \Gamma \vdash B : \rho_B \quad \mathrm{sat}(\Gamma)}{\vdash \langle\Gamma\rangle\ \tau_r\ f(\overline{\tau_x\ x})\ B}\ [\text{FDEC}]$$

$$\frac{\Delta, \overline{x : \tau_x}; \Gamma \vdash S_i : \rho_i \quad i \in \{1,\ldots,n\}}{\Delta; \Gamma \vdash \{\overline{\eta_x\ x;}\ S_1; \ldots S_n;\} : \boxtimes_i\, \rho_i}\ [\text{BLOCK}]$$

$$\frac{\Delta; \Gamma \vdash e : \rho_e \quad \tau_x = \Delta(x) \quad \Gamma \triangleright \rho_e \sqsubseteq \tau_x}{\Delta; \Gamma \vdash x = e : \tau_x}\ [\text{ASSIGN}]$$

$$\frac{\begin{array}{l}\Delta; \Gamma \vdash e : \rho_e \quad \Delta; \Gamma \vdash B_t : \rho_t \quad \Delta; \Gamma \vdash B_f : \rho_f \\ \Gamma \triangleright \rho_e \sqsubseteq \rho_t \quad \Gamma \triangleright \rho_e \sqsubseteq \rho_f\end{array}}{\Delta; \Gamma \vdash \text{if } (e)\ B_t \text{ else } B_f : \rho_t \boxtimes \rho_f}\ [\text{IF}]$$

$$\frac{\Delta; \Gamma \vdash S : \rho \quad \Gamma \triangleright \rho' \sqsubseteq \rho}{\Delta; \Gamma \vdash S : \rho'}\ [\text{SUB-S}] \qquad \frac{\Delta; \Gamma \vdash e : \rho \quad \Gamma \triangleright \rho \sqsubseteq \rho'}{\Delta; \Gamma \vdash e : \rho'}\ [\text{SUB-E}]$$

$$\frac{\tau = \Delta(x)}{\Delta; \Gamma \vdash x : \tau}\ [\text{VAR}] \qquad \frac{t \in \{\text{true,false}\}}{\Delta; \Gamma \vdash t : L}\ [\text{CONST}] \qquad \frac{\Delta; \Gamma \vdash e_1 : \rho_1 \quad \Delta; \Gamma \vdash e_2 : \rho_2}{\Delta; \Gamma \vdash e_1 \odot e_2 : \rho_1 \bowtie \rho_2}\ [\text{OP}]$$

$$\frac{\begin{array}{l}\Delta; \Gamma \vdash e_i : \rho_{e_i} \quad i \in \{1,\ldots,n\} \\ y_1 : \tau_1, \ldots, y_n : \tau_n \to \tau_r; \Gamma_f = \mathit{ftype}(f) \\ \sigma = [\overline{X \mapsto \tau}] \quad \Gamma \triangleright \rho_{e_i} \sqsubseteq \tau_i\sigma \quad \forall c \in \Gamma_f\sigma\ .\ \Gamma \triangleright c\end{array}}{\Delta; \Gamma \vdash f[\overline{X \mapsto \tau}](e_1, \ldots, e_n) : \rho_r\sigma}\ [\text{CALL}]$$

図 2　型付け規則

は，代入文の型付けの条件として右辺式の機密度が代入先変数の機密度以下でなければならないことを表す．ρ_e や τ_x には機密度パラメータが含まれる可能性があり，これらの機密度パラメータに対する制約を表す Γ の下で条件が満たされるか検査する．関数呼び出し式に対する CALL 規則では 2 種類の検査を行う．1 つは，実引数を仮引数に代入するために実引数の機密度が仮引数の機密度以下になっていることの検査である．この時，仮引数の機密度が機密度パラメータで表されている可能性があるため，機密度パラメータに関数呼び出し時に指定された機密度を割り当ててから検査する．もう 1 つは，関数定義時の機密度パラメータに対する制約が関数呼び出し時に指定された割り当てを適用しても充足されることの検査である．この時，呼び出し先関数の機密度パラメータに対して呼び出し元関数における機密度パラメータが割り当てられる可能性があるため，呼び出し元関数における機密度パラメータに対する制約リスト Γ の下で各制約が充足されるか検査する．

3.3　適用例

提案する型システムの適用例として図 4 のプログラムを考える．ここで，X_1, X_2, Y および A_1, A_2, A_3, B は機密度パラメータである．== は等価判定演算，&& は論理積演算であり，いずれも型付けには OP 規則が適用される．equals2 は 2 引数の値が等しいか判定する関数である．引数と返り値の機密度はすべて機密度パラメータを用いて表されている．2 引数はともに返り値に影響を与えるため，2 引数の機密度パラメータと返り値の機密度パラメータの間には制約が存在する．equals3 は 3 引数の値が等しいか判定する関数であり，equals2 関数を使って実装されている．equals3

$$\frac{\rho \sqsubseteq \rho' \in \Gamma}{\Gamma \rhd \rho \sqsubseteq \rho'} \qquad \frac{}{\Gamma \rhd L \sqsubseteq H} \qquad \frac{}{\Gamma \rhd \rho \sqsubseteq \rho}$$

$$\frac{\Gamma \rhd \rho \sqsubseteq \rho' \quad \Gamma \rhd \rho' \sqsubseteq \rho''}{\Gamma \rhd \rho \sqsubseteq \rho''} \qquad \frac{\Gamma \rhd \rho' \sqsubseteq \rho \quad \Gamma \rhd \rho'' \sqsubseteq \rho}{\Gamma \rhd \rho' \bowtie \rho'' \sqsubseteq \rho} \qquad \frac{\Gamma \rhd \rho \sqsubseteq \rho' \quad \Gamma \rhd \rho \sqsubseteq \rho''}{\Gamma \rhd \rho \sqsubseteq \rho' \bowtie \rho''}$$

図 3　制約の導出規則

$\langle X_1 \sqsubseteq Y, X_2 \sqsubseteq Y \rangle$ Y equals2(X_1 x_1, X_2 x_2) {
    result = $x_1 == x_2$;
}

$\langle A_1 \sqsubseteq B, A_2 \sqsubseteq B, A_3 \sqsubseteq B \rangle$ B equals3(A_1 a_1, A_2 a_2, A_3 a_3) {
    result = equals2$[Y \mapsto B, X_1 \mapsto A_1, X_2 \mapsto A_2](a_1, a_2)$
        &&
        equals2$[Y \mapsto B, X_1 \mapsto A_2, X_2 \mapsto A_3](a_2, a_3)$;
}

図 4　例題プログラム

についても equals2 と同様に，3 引数の機密度パラメータと返り値の機密度パラメータの間に制約が存在する．equals2 関数の呼び出し時に，equals2 関数の機密度パラメータと equals3 関数の機密度パラメータの対応が指定されている．

equals2 関数の本体に対する型導出木を図 5 に示す．ここで，$\Delta = x_1 : X_1, x_2 : X_2, \mathrm{result} : Y$，$\Gamma = X_1 \sqsubseteq Y, X_2 \sqsubseteq Y$ である．代入の右辺は x_1, x_2 から構成されるため機密度は $X_1 \bowtie X_2$，左辺の result の機密度は Y である．代入文の右辺の機密度は左辺の機密度以下でなければならないため $X_1 \bowtie X_2 \sqsubseteq Y$ が成り立つ必要があるが，機密度パラメータに対する制約リスト Γ の下で図 3 の規則から導出できる．そのため，代入文は型付け可能であり，関数本体も型付け可能である．

equals3 関数における 1 つ目の equals2 呼び出し式の型導出木を図 6 に示す．ここで，$\Delta = a_1 : A_1, a_2 : A_2, a_3 : A_3, \mathrm{result} : B$，$\Gamma = A_1 \sqsubseteq B, A_2 \sqsubseteq B, A_3 \sqsubseteq B$ である．Γ' は equals2 関数における機密度パラメータに対する制約リストであり，σ は equals2 関数における機密度パラメータに対する割り当てである．関数呼び出し式において，実引数の機密度は仮引数の機密度以下でなければならない．1 つ目の引数の場合，実引数の機密度は A_1，仮引数の機密度は X_1 であり，X_1 に対する割り当てが A_1 であるため，$A_1 \sqsubseteq X_1\sigma$ すなわち $A_1 \sqsubseteq A_1$ は成り立つ．2 つ目の引数についても同様である．さらに，equals2 関数における機密度パラメータに対する割り当てが Γ' で表される制約リストを充足する必要があるが，制約リストに割り当てを適用すると $\Gamma'\sigma = A_1 \sqsubseteq B, A_2 \sqsubseteq B$ が得られ，この中のすべての制約が Γ に含まれているため Γ の下で充足される．以上より，1 つ目の equals2 関数の呼び出し式は型付け可能であり，型は $Y\sigma = B$ である．同様にして 2 つ目の equals2 関数の呼び出し式も型 B で型付けできる．以下，$B \bowtie B = B \sqsubseteq B$ から equals2 関数の本体の型付けと同様にして equals3 関数の本体も型付け可能であることがわかる．

4　おわりに

本稿では，情報流解析における制約付き機密度パラメータと，対応する型システムを提案した．機密度パラメータにより機密度のみが異なる多数のプログラムの作

$$\dfrac{\dfrac{\dfrac{\Delta(x_1)=X_1}{\Delta;\Gamma\vdash x_1:X_1}\quad\dfrac{\Delta(x_2)=X_2}{\Delta;\Gamma\vdash x_2:X_2}}{\Delta;\Gamma\vdash x_1==x_2:X_1\bowtie X_2}\qquad\Delta(\mathrm{result})=Y\qquad\dfrac{\dfrac{X_1\sqsubseteq Y\in\Gamma}{\Gamma\rhd X_1\sqsubseteq Y}\quad\dfrac{X_2\sqsubseteq Y\in\Gamma}{\Gamma\rhd X_2\sqsubseteq Y}}{\Gamma\rhd X_1\bowtie X_2\sqsubseteq Y}}{\dfrac{\Delta;\Gamma\vdash \mathrm{result}=x_1==x_2:Y}{\Delta;\Gamma\vdash\{\mathrm{result}=x_1==x_2;\}:Y}}$$

図 5　equals2 関数本体の型導出木

$$\dfrac{\dfrac{\Delta(a_1)=A_1}{\Delta;\Gamma\vdash a_1:A_1}\quad\dfrac{\Delta(a_2)=A_2}{\Delta;\Gamma\vdash a_2:A_2}\qquad\begin{array}{l}\mathit{ftype}(\mathrm{equals2})=x_1:X_1,x_2:X_2\to Y;\Gamma'\\ \Gamma'=X_1\sqsubseteq Y,X_2\sqsubseteq Y\\ \sigma=[Y\mapsto B,X_1\mapsto A_1,X_2\mapsto A_2]\\ \Gamma\rhd A_1\sqsubseteq X_1\sigma\quad\Gamma\rhd A_2\sqsubseteq X_2\sigma\\ \Gamma'\sigma=A_1\sqsubseteq B,A_2\sqsubseteq B\\ \Gamma\rhd A_1\sqsubseteq B\quad\Gamma\rhd A_2\sqsubseteq B\end{array}}{\Delta;\Gamma\vdash\mathrm{equals2}[Y\mapsto B,X_1\mapsto A_1,X_2\mapsto A_2](a_1,a_2):B}$$

図 6　equals3 関数における 1 つ目の equals2 呼び出し式の型導出木

成を回避できるが，機密度パラメータに対してどのような機密度が割り当てられるか事前にはわからないため，任意の機密度の割り当てに対して非干渉性を満たすことが要求される．機密度パラメータに対して制約を与え，任意の機密度ではなく制約を満たす機密度の割り当てに対して非干渉性を求めることで，型付け可能なプログラムの範囲が拡大される．制約を満たさない機密度の割り当ては排除されるため，実行時に不正な情報流が発生することはない．

提案する型システムが非干渉性に対して健全であることの証明は今後の課題である．[2] と同様に，型付け可能なプログラムを機密度の高いデータのみが異なる 2 つの初期状態から実行すると，実行後の 2 状態において機密度の低いデータは等価であることを示す．型検査アルゴリズムの構築や，クラスベースのオブジェクト指向言語に対する制約付き機密度パラメータの導入も今後の課題である．データ構造に対する制約付き機密度パラメータの効果を確認する必要がある．

謝辞　本研究の一部は JSPS 科研費 JP15K00112，JP17K12666，JP18K11241 および 2018 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

参考文献

[1] Anindya Banerjee and David A. Naumann. Secure Information Flow and Pointer Confinement in a Java-like Language. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pp. 253–267, 2002.

[2] 黒川翔, 桑原寛明, 山本晋一郎, 坂部俊樹, 酒井正彦, 草刈圭一朗, 西田直樹. 例外処理付きオブジェクト指向プログラムにおける情報流の安全性解析のための型システム. 電子情報通信学会論文誌 D, Vol. J91-D, No. 3, pp. 757–770, 2008.

[3] Andrei Sabelfeld and Andrew C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications*, Vol. 21, No. 1, pp. 5–19, 2003.

[4] Dennis Volpano, Geoffrey Smith, and Cynthia Irvine. A Sound Type System for Secure Flow Analysis. *Journal of Computer Security*, Vol. 4, No. 2, pp. 167–187, 1996.

[5] 吉田真也, 桑原寛明, 國枝義敏. オブジェクト指向言語の情報流解析における機密度のパラメータ化. In *FOSE 2017*, pp. 83–92, 2017.

[6] 桑原寛明. 型検査に基づく手続き型言語向け情報流解析における型エラースライシング. コンピュータソフトウェア, Vol. 27, No. 4, pp. 221–227, 2010.