A Modal Foundation for Secure Information Flow

Kenji Miyamoto and Atsushi Igarashi
Graduate School of Informatics,
Kyoto University, JAPAN
Background

- Many type-based techniques for information flow analysis (IFA) (e.g. SLam \cite{Heintze98})

- However, the essence of the type systems is not very clear
  - Subtle differences among their cores
  - It is not clear whether the differences are essential or not
Our goal

- Clarification of the essence of type-based IFA

  ↓

- Uniform framework which can represent various type systems for IFA
Approach

- To show a relationship between
  - type-based IFA
  - modal logic

- Development of a typed calculus based on the modal logic
  - Via Curry-Howard isomorphism

- Encoding existing calculi for IFA to $\lambda^\sbot$
 Contribution

- We show modal logic of local validity corresponds to type-based IFA
- Formalization of $\lambda_s$ based on the modal logic
  - Simple proof of noninterference
- Encoding of a core of the SLam calculus to $\lambda_s$
Contents

- Information flow analysis
- Modal logic
- $\lambda_S$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
Information flow analysis

- Program analysis to ensure
  - The absence of data leakage
    - e.g. private data (your salary) does not leak to public
  - a.k.a. the noninterference property
Security level

- Level of secrecy of data
- We assign security level to each datum
- Some data have high security level
- Some data have low security level
  - For example, private data (your salary) has higher security level than public data (everybody can read)
Leakage of data

- Two kinds of leakage
  - Direct leakage of data
  - Indirect leakage of data
- IFA detects both kinds of leakage
Direct leakage of data

```c
int pub:=0L; //L means public
int salary:=400H; //H means private
...

pub:=salary;
print(pub);
```

By printing the value of pub, we can know the value of salary
Indirect leakage of data

```c
int pub:=0L; //L means public
int salary:=400H; //H means private
...
if salary>300 then pub:=1 else pub:=2;
```

By reading a value of pub, we can know whether salary is over 300 or not
Noninterference

- Correctness property of IFA

Whatever high security input is given, low security output is unchanged
Contents

- Information flow analysis
- Modal logic
- $\lambda_S$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
Relationship between IFA and modal logic

- We can consider
  - Security levels as possible worlds
  - Order of security as reachability relation
    - High security world is reachable from low security world

↓
What kind of modality is appropriate?
Local validity as modality

- “A holds at all worlds reachable from a certain world S”
  - We write it $\square_s A$

- It is appropriate because, in IFA, low security level data can be read at high security level
  - We represent low security data type as $\square_{\text{low, int}}$
Contents

- Information flow analysis
- Modal logic
- $\lambda_S$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
Term calculus for logic of local validity

Extension of simply typed lambda calculus with modal types

Type system for IFA
Syntax

S: element of poset of security levels

Type A ::= K | A → A | □ₜ A

Base type K ::= unit | int | string | ...

Term M ::= c | x | u

| (λ x:A.M) | (MM)
| (boxₜ M)
| (let boxₜ u=M in M)
box and let box

- boxₜM
  - Seals M at security level S

- let boxₜu=M in N
  - Unseals M, binds u to the unsealed value, and executes N
Main reduction rules

- $(\lambda x:A.M)N \rightarrow [N/x]M$
- let $\text{box}_S\ u = \text{box}_S\ M$ in $N \rightarrow [M/u]N$
Judgment

- Context consists of two parts:
  - Modal context $\Delta$ containing locally valid assumptions $u_1::L^1A_1, u_2::L^2A_2, \ldots$
  - Ordinary context $\Gamma$ containing truth assumptions $x_1:B_1, x_2:B_2, \ldots$
    - c.f. Davies and Pfenning’s formalization of modal logic [Davies and Pfenning POPL96]

- Judgments are of the form:
  $$\Delta ; \Gamma \vdash^S M : A$$
  - $M$ has type $A$ at level $S$, under $\Delta$ and $\Gamma$
Main typing rules (1/3)

Rule for modal variables

\[ \frac{U::S^1 A \in \Delta \quad S_1 \leq S_2}{\Delta; \Gamma \vdash^{S^2} U:A} \quad (T-Mvar) \]

- Current level \( S_2 \) must be reachable from \( U \)'s level
  - Data readable at low security level \( S_1 \)
  - Also readable at high security level \( S_2 \)
Main typing rules (2/3)

Rule for box

\[
\frac{\Delta; \cdot \vdash^S M:A}{\Delta; \Gamma \vdash^{S_2} \text{box}_{S_1} M:\Box_{S_1} A}
\]  

(T-Box)

- The rule corresponds to \( \Box \)-introduction
- The premise means \( \Delta; \cdot \vdash^S M:A \) can be derived for any level \( S \geq S_1 \)
  - Ordinary context is empty
  - The levels of modal variables in \( M \) are higher than \( S_1 \)
Main typing rules (3/3)

Rule for let box

\[ \Delta; \Gamma \vdash^{S_1} M : \Box_{S_2} A \quad \Delta, u ::^{S_2} A; \Gamma \vdash^{S_1} N : B \]

\[
\frac{}{\Delta; \Gamma \vdash^{S_1} \text{let box}_{S_2} u = M \text{ in } N : B} \quad \text{(T-Letbox)}
\]

- The rule corresponds to \( \Box \)-elimination
- “\( \Box_{S_2} A \) is true” turns into “\( A \) is valid at \( S_2 \)”
- We can unseal \( M : \Box_{S_2} A \) at any security level, but usage of \( u \) is limited by the rule T-Mvar
Example

The example of indirect leakage

print:\((\Box L \text{int}) \to \text{unit}\)
salary:\(\Box H \text{int}\)

print(let box_H u=salary in

box_L (if u>300 then 1 else 2))

- We cannot use u in box_L due to T-Mvar. Thus, this program is not typed.
Properties

- Subject reduction
- Church-Rosser
- Strong Normalization
- Noninterference
Noninterference Theorem

- If
  - \( u :: \text{int} \);
  - \( \vdash^T M : \text{int} \)
  - \( S > T \)

- Then
  - there exists a unique normal form \( M' \) such that
    - for any \( N \), if \( \vdash^S N : \text{int} \) then \( [N/u]M \rightarrow^* M' \)
Proof sketch

- Lemma
  - If \( u::\text{int} ; \cdot \vdash^T M:\text{int} \) and \( M \) is a normal form and \( u \in \text{FMV}(M) \) then \( S \subseteq T \)
  - \( \exists ! M' \) s.t. \( M:\text{int} \rightarrow {}^* M' : \text{int} \) and \( M' \) is normal form
  - \( [N/u]M \rightarrow {}^* [N/u]M' = M' \) (by the contraposition of the lemma)
Contents

- Information flow analysis
- Modal logic
- $\lambda_S$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
SLam calculus [Heintze & Riecke 98]

- Type-based IFA for higher-order language i.e. \( \lambda \)-calculus
- Secure types
  - Security level is attached to each type constructor
    - \( T ::= \text{unit}^S \mid \text{int}^S \mid T \rightarrow^S T \mid \ldots \)
Encoding to $\lambda_S$

- **Source:** SLam — recursion and protected
- **Overview of encoding**
  - $\Delta \vdash e: t^S \Rightarrow |\Delta| \cdot \vdash^S l e : |l t|$
  - $\text{int}^H$ is translated to $\square^H \text{int}$
  - Subsumption translates to coercion
    - $(\text{unit}, H) \leq (\text{unit}, L)$ to $\lambda x: \square^L \text{unit}. \text{let box}_L u_x = x \text{ in } u_x$
- **Properties**
  - Encoding preserves typing
  - Translated programs enjoy noninterference
Contents

- Information flow analysis
- Modal logic
- $\lambda_S$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
Related work (1/2)

- Type-based IFA for functional languages
  - Fairly complex proofs of noninterference using
    - denotational semantics [Heintze and Reicke, POPL98]
    - non-standard operational semantics [Pottier and Simonet TOPLAS03]
  - Noninterference of our system is proved in a simple manner
    - Our proof is similar to the proof of noninterference of FOb₁<-[Barthe and Serpette FLOPS99]
Related work (2/2)

- **DCC** [Abadi et al. POPL 99]
  - A calculus to unify dependency analyses
  - SLam is one of the instances of DCC
  - DCC is monadic type based

- Monadic types of DCC are similar to modal types in their roles, but
- Typing rules are rather different
Contents

- Information flow analysis
- Modal logic
- $\lambda_s$
- Encoding the SLam calculus
- Related work
- Conclusion and future work
Conclusion

- Relationship between IFA and modal logic
- $\lambda_s^\square$ enjoys subject reduction, Church-Rosser, strong normalization, and noninterference
- A translation from SLam to $\lambda_s^\square$
Future work

- To compare $\lambda_s$ with other calculi for IFA
- To figure out how modal types of $\lambda_s$ and monadic types of DCC correspond to each other
- Adding side effects and recursion
End

Kiitos
\[ \Gamma, x : s_1 \vdash e_0 : s_2 \]

\[
\Gamma \vdash (\lambda x : s_1. e_0)_{L} : (s_1 \rightarrow s_2, L)
\]