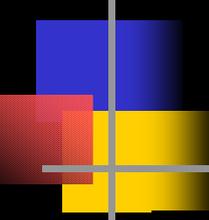


Logical Relations for Encryption

Eijiro Sumii

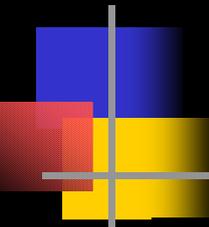
University of Tokyo

Joint work with Benjamin Pierce,
University of Pennsylvania



Overview

- Introduction
- The cryptographic λ -calculus
- Logical relations
- Application: protocol encoding
- Extensions
- Related work
- Conclusion

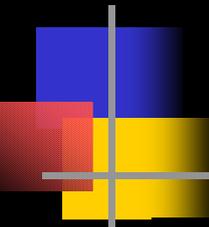


Motivation

Two approaches to information hiding:

- Encryption
 - mainly studied in security systems
- Type abstraction
 - mainly studied in programming languages (polymorphism, modules, objects, etc.)

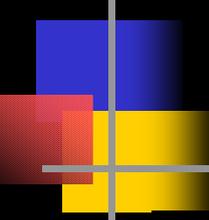
How are these related?



Results

Adapting the theory of type abstraction
for encryption

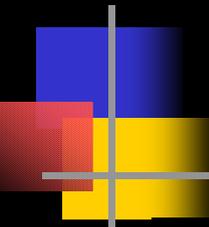
- Cryptographic λ -calculus +
 - Logical relation of the polymorphic λ -calculus
- ⇒ Method of proving secrecy in programs
using encryption



Example

A program $p(i)$ consisting of

- a secret integer i and
- an interface function $\lambda x. x \bmod 2$

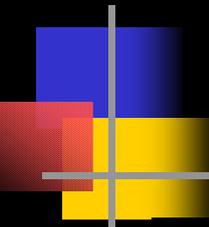


Example

A program $p(i)$ consisting of

- a secret integer i and
- an interface function $\lambda x. x \bmod 2$
- Information hiding by type abstraction

$$p(i) = \text{pack int}, \langle i, \lambda x. x \bmod 2 \rangle \\ \text{as } \exists \alpha. \alpha \times (\alpha \rightarrow \text{int})$$



Example

A program $p(i)$ consisting of

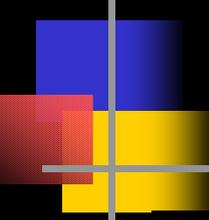
- a secret integer i and
- an interface function $\lambda x. x \bmod 2$

■ Information hiding by type abstraction

$$p(i) = \text{pack int}, \langle i, \lambda x. x \bmod 2 \rangle \\ \text{as } \exists \alpha. \alpha \times (\alpha \rightarrow \text{int})$$

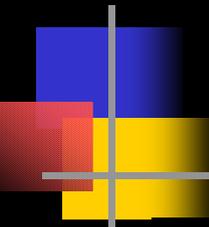
■ Information hiding by encryption

$$p(i) = \text{new } k \text{ in } \langle \{i\}_k, \lambda \{x\}_k. x \bmod 2 \rangle$$



Overview

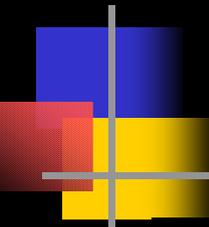
- Introduction
- The cryptographic λ -calculus
- Logical relations
- Application: protocol encoding
- Extensions
- Related work
- Conclusion



The Cryptographic λ -Calculus

Simply typed call-by-value λ -calculus
+ (perfect) cryptographic primitives

$$e ::= \{e_1\}_{e_2} \mid \text{let } \{x\}_{e_1} = e_2 \text{ in } e_3 \text{ else } e_4 \\ \mid \text{new } x \text{ in } e \mid k \mid \dots$$
$$\tau ::= \text{bits}[\tau] \mid \text{key}[\tau] \mid \dots$$



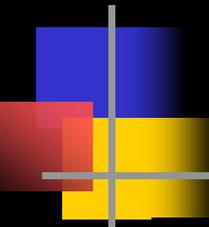
The Cryptographic λ -Calculus

Simply typed call-by-value λ -calculus
+ (perfect) cryptographic primitives

$$e ::= \{e_1\}_{e_2} \mid \text{let } \{x\}_{e_1} = e_2 \text{ in } e_3 \text{ else } e_4 \\ \mid \text{new } x \text{ in } e \mid k \mid \dots$$
$$\tau ::= \text{bits}[\tau] \mid \text{key}[\tau] \mid \dots$$

$\text{new } x \text{ in } e \rightarrow [k/x]e$ (k fresh)

$\text{let } \{x\}_{k_1} = \{v\}_{k_2} \text{ in } e_1 \text{ else } e_2 \\ \rightarrow [v/x]e_1$ (if $k_1 = k_2$) or e_2 (if $k_1 \neq k_2$)

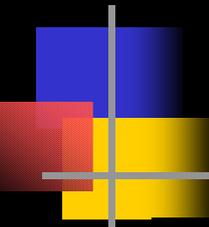


Secrecy \cong Non-Interference \cong Contextual Equivalence

[Q] How to state the (partial) secrecy of the value of i ?

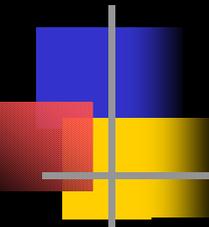
[A] By conditional non-interference:
if $i \equiv j \pmod{2}$, then $p(i)$ and $p(j)$ are
equivalent under any context

"Outsiders cannot observe
the difference of the secret"



Overview

- Introduction
- The cryptographic λ -calculus
- Logical relations
- Application: protocol encoding
- Extensions
- Related work
- Conclusion



Logical Relation

[Q] How to prove contextual equivalence?

[A] By a logical relation " \sim " between programs, defined by induction on their type

Main theorem:

$$e_1 \sim e_2 : \tau \Rightarrow e_1 \approx e_2 : \tau$$

"related programs are contextually equivalent"

Logical Relation for Simple Types (standard)

- Integers are related iff they are equal

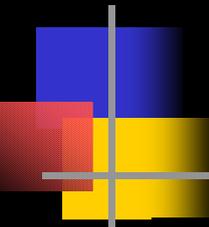
$$i \sim j : \text{int} \Leftrightarrow i = j$$

- Functions are related iff they return related results when applied to related arguments

$$f \sim g : \tau_1 \rightarrow \tau_2 \Leftrightarrow \\ f v \sim g w : \tau_2 \text{ for any } v \sim w : \tau_1$$

- Pairs are related iff their elements are related

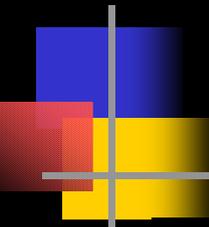
$$(v_1, v_2) \sim (w_1, w_2) : \tau_1 \times \tau_2 \Leftrightarrow \\ v_1 \sim w_1 : \tau_1 \text{ and } v_2 \sim w_2 : \tau_2$$



Logical Relation for Type Abstraction (also standard)

The relation environment φ gives the relation $\varphi(\alpha)$ between values of each abstract type α

$$\varphi \quad v_1 \sim v_2 : \alpha \iff (v_1, v_2) \in \varphi(\alpha)$$



Logical Relation for Type Abstraction (also standard)

The relation environment φ gives the relation $\varphi(\alpha)$ between values of each abstract type α

$$\varphi \quad v_1 \sim v_2 : \alpha \iff (v_1, v_2) \in \varphi(\alpha)$$

$$\varphi \quad \text{pack } \sigma_1, e_1 \text{ as } \exists \alpha. \tau$$

$$\sim \text{pack } \sigma_2, e_2 \text{ as } \exists \alpha. \tau : \exists \alpha. \tau \iff$$

$$\varphi, \alpha \mapsto r \quad e_1 \sim e_2 : \tau \text{ for some } r \subseteq \sigma_1 \times \sigma_2$$

Logical Relation for Type Abstraction (also standard)

The relation environment φ gives the relation $\varphi(\alpha)$ between values of each abstract type α

$$\varphi \quad v_1 \sim v_2 : \alpha \iff (v_1, v_2) \in \varphi(\alpha)$$

$$\varphi \quad \text{pack } \sigma_1, e_1 \text{ as } \exists \alpha. \tau$$

$$\sim \text{pack } \sigma_2, e_2 \text{ as } \exists \alpha. \tau : \exists \alpha. \tau \iff$$

$$\varphi, \alpha \mapsto r \quad e_1 \sim e_2 : \tau \text{ for some } r \subseteq \sigma_1 \times \sigma_2$$

E.g., $\text{pack int}, \langle 1, \lambda x. x \bmod 2 \rangle \text{ as } \exists \alpha. \alpha \times (\alpha \rightarrow \text{int})$
and $\text{pack int}, \langle 3, \lambda x. x \bmod 2 \rangle \text{ as } \exists \alpha. \alpha \times (\alpha \rightarrow \text{int})$
can be related by taking $\alpha \mapsto \{(1, 3)\}$

Logical Relation for Encryption (new!)

The relation environment φ gives the relation $\varphi(k)$ between values encrypted by each secret key k

$$\varphi \quad \{v_1\}_{k_1} \sim \{v_2\}_{k_2} : \text{bits}[\tau] \Leftrightarrow \\ (v_1, v_2) \in \varphi(k) \quad \text{where } k = k_1 = k_2$$

Logical Relation for Encryption (new!)

The relation environment φ gives the relation $\varphi(k)$ between values encrypted by each secret key k

$$\varphi \quad \{v_1\}_{k_1} \sim \{v_2\}_{k_2} : \text{bits}[\tau] \Leftrightarrow \\ (v_1, v_2) \in \varphi(k) \quad \text{where } k = k_1 = k_2$$

$$\varphi \quad \text{new } k \text{ in } e_1 \sim \text{new } k \text{ in } e_2 : \tau \Leftrightarrow \\ \varphi, k \mapsto r \quad e_1 \sim e_2 : \tau \text{ for some } r$$

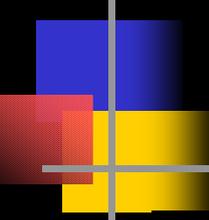
Logical Relation for Encryption (new!)

The relation environment φ gives the relation $\varphi(k)$ between values encrypted by each secret key k

$$\varphi \quad \{v_1\}_{k_1} \sim \{v_2\}_{k_2} : \text{bits}[\tau] \Leftrightarrow \\ (v_1, v_2) \in \varphi(k) \text{ where } k = k_1 = k_2$$

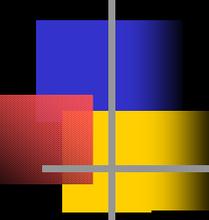
$$\varphi \quad \text{new } k \text{ in } e_1 \sim \text{new } k \text{ in } e_2 : \tau \Leftrightarrow \\ \varphi, k \mapsto r \quad e_1 \sim e_2 : \tau \text{ for some } r$$

E.g., $\text{new } k \text{ in } \langle \{1\}_k, \lambda\{x\}_k. x \bmod 2 \rangle$
and $\text{new } k \text{ in } \langle \{3\}_k, \lambda\{x\}_k. x \bmod 2 \rangle$
can be related by taking $k \mapsto \{(1, 3)\}$



Overview

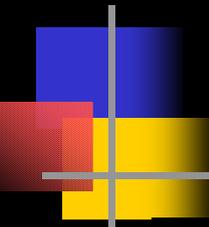
- Introduction
- The cryptographic λ -calculus
- Logical relations
- Application: protocol encoding
- Extensions
- Related work
- Conclusion



Application: Protocol Encoding

Encode:

- Sending of a message by the message itself
- Receiving of a message by a function
- Network and attacker by a context



Application: Protocol Encoding

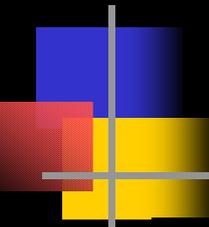
Encode:

- Sending of a message by the message itself
- Receiving of a message by a function
- Network and attacker by a context

E.g.,

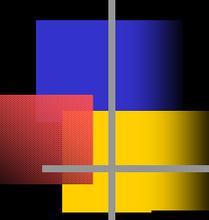
1. $A \rightarrow B \quad \{i\}_k$
2. $B \rightarrow * \quad i \bmod 2$

- $p = \text{new } k \text{ in } \langle \{i\}_k, \lambda\{x\}_k. x \bmod 2 \rangle$
- $\text{Network}(p) = \#_2(p) \#_1(p) \rightarrow^* i \bmod 2$
- $\text{Attacker}(p) = \text{any context for } p$



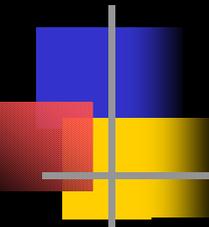
Examples

- Well-known attack on (a bad use of) Needham-Schroeder public-key protocol
- Correctness proof of (the same use of) "improved" Needham-Schroeder public-key protocol
- "Necessarily parallel" attack on ffgg protocol



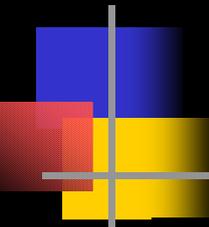
Overview

- Introduction
- The cryptographic λ -calculus
- Logical relations
- Application: protocol encoding
- Extensions
- Related work
- Conclusion



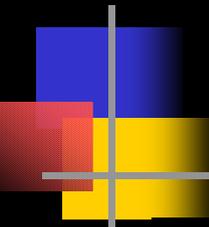
Extensions

- Recursive functions/types
 - for making the attackers Turing-complete
 - cf. [Pitts-98], [Crary-Harper], etc.
- State/linearity
 - for encoding protocols more precisely
 - cf. [Pitts-Stark-98], [Bierman-Pitts-Russo-00]



Related Work

- Logical relations
 - Relational parametricity [Reynolds-83]
 - Representation independence [Mitchell-91]
 - λ -calculus with name generation [Stark-94]
- Protocol verification
 - Various logics, theorem proving, model checking, etc. [many!]
 - In particular, spi-calculus [Abadi-Gordon]



Conclusion

- We have adapted the theory of type abstraction to encryption
- Can we do something in the other direction?
E.g., implement type abstraction by encryption
I.e., encode the polymorphic λ -calculus into the untyped cryptographic λ -calculus (while preserving contextual equivalence)
 \Rightarrow Extend the scope of type abstraction from the statically typed world to the untyped world (such as open network)