

A Calculus for Game-Based Security Proofs¹

David NOWAK

Research Center for Information Security, AIST, Tokyo

FAIS Organized Session, JSIAM 2010
September 6, 2010

¹Joint Work with Yu ZHANG, ISCAS, China

Introduction

- ▶ It is now common practice to write security proofs for cryptographic constructions as sequences of games.
- ▶ Such proofs require sanity checks:
 - ▶ Each game transformation must be justified;
 - ▶ When building an attacker, one has to be sure that it is Probabilistic Polynomial Time (PPT).
- ▶ We propose a language and an equational theory that makes such proofs purely syntactic.
- ▶ Computers are based on binary digits, thus our language too.
 - ▶ The only base type is for bitstrings.
 - ▶ This way we are closer to implementations.

Introduction

Characterizing polytime functions: From Cobham to Zhang

Computational SLR

Introduction

Characterizing polytime functions: From Cobham to Zhang

Computational SLR

Cobham's characterization of polytime

- ▶ The class of Cobham (1964):
 - i. **Constant 0**
 - ii. **Projection** $\pi_j^n(x_1, \dots, x_n) = x_j$
 - iii. **Successors** $s_i(x) = x_i$ for $i \in \{0, 1\}$
 - iv. **smash** $2^{|x_1| \cdot |x_2|}$
 - v. **Recursion**
 $f(0, \bar{x}) = g(\bar{x})$
 $f(y_i, \bar{x}) = h_i(y, \bar{x}, f(y, \bar{x}))$ for $y_i \neq 0$
 $|f(y, \bar{x})| \leq |j(y, \bar{x})|$
where g, h_0, h_1 and j are in this class
 - vi. **Composition** $f(\bar{x}) = h(\bar{r}(\bar{x}))$
where h and \bar{r} are in this class
- ▶ This is exactly the class of functions computable in polynomial time on a deterministic Turing machine.
- ▶ This is not a fully syntactic characterization:
A bound has to be proved for recursion.

Safe recursion

- ▶ The class of Bellantoni and Cook (1992):

- Constant 0**
- Projection** $\pi_j^{m,n}(x_1, \dots, x_m; x_{m+1}, \dots, x_{m+n}) = x_j$
- Successors** $s_i(; a) = ai$ for $i \in \{0, 1\}$
- Predecessor** $p(; 0) = 0$ and $p(; ai) = a$
- Recursion** $f(0, \bar{x}; \bar{a}) = g(\bar{x}; \bar{a})$
 $f(yi, \bar{x}; \bar{a}) = h_i(y, \bar{x}; \bar{a}, f(y, \bar{x}; \bar{a}))$ for $yi \neq 0$
where g , h_0 and h_1 are in this class
- Composition** $f(\bar{x}; \bar{a}) = h(\bar{r}(\bar{x};); \bar{t}(\bar{x}; \bar{a}))$
where h , \bar{r} and \bar{t} are in this class

- ▶ This class is equivalent to the Cobham's one.
- ▶ But there is no bound to prove.
- ▶ Instead, there are two kinds of variables: “normal” and “safe”.
- ▶ It is not allowed to recur on safe variables.

From Cobham's class to Bellantoni and Cook's class

Recursion Simulation Lemma

For all f in Cobham's class, there exists an f' in Bellantoni and Cook's class and a monotone polynomial p_f such that for all \bar{a} and all w , $|w| \geq p_f(|\bar{a}|)$ implies $f(\bar{a}) = f'(w; \bar{a})$.

Theorem Bellantoni and Cook's class contains Cobham's class.

Proof. Let $f(\bar{x})$ be a function on Cobham's class.

Let f' and p_f be obtained using the above lemma.

One can construct a $b(\bar{x};)$ such that $|b(\bar{x})| \geq p_f(|\bar{x}|)$ and obtain $f''(\bar{x}) = f'(b(\bar{x};); \bar{x})$ in Bellantoni and Cook's class.

From Bellantoni and Cook's class to Cobham's class

Polymax Bounding Lemma

For all f in Bellantoni and Cook's class,
there exists a monotone polynomial q_f such that
forall \bar{x} and \bar{a} , $|f(\bar{x}; \bar{a})| \leq q_f(|\bar{x}|) + \max(|\bar{a}|)$.

Theorem Cobham's class contains Bellantoni and Cook's class.

Proof. Initial functions and composition are easily translated.
The above bound can be programmed in Cobham's class and gives
the function j used for recursion.

SLR: Generalization to higher order

- ▶ SLR (Hofmann, 1997): a simply-typed lambda calculus with:
 - ▶ an S4 modality \Box , and
 - ▶ linear function spaces $(-\multimap)$.
- ▶ It generalizes Bellantoni and Cook's scheme to higher-order.
 - ▶ A function with m normal and n safe variables has type:

$$(\Box N)^m \rightarrow N^n \rightarrow N$$

- ▶ It denotes a function f whose size is bounded:

$$|f(\bar{x}; \bar{a})| \leq P(|\bar{x}|) + \max(|\bar{a}|)$$

- ▶ Linear functions are not needed to characterize polytime:
They are provided for convenience.
- ▶ Subtyping: $A \multimap B <: A \rightarrow B <: \Box A \rightarrow B$
- ▶ There is a type inference algorithm.

Examples of SLR functions and their inferred types

$$\lambda x^A. x : A \multimap A$$

⋮

$$\lambda f^{A \rightarrow B}. \lambda x^A. f x : (A \rightarrow B) \multimap A \rightarrow B$$

⋮

$$\lambda f^{\Box A \rightarrow B}. \lambda x^A. f x : (\Box A \rightarrow B) \multimap \Box A \rightarrow B$$

⋮

$$\lambda f^{\Box A \rightarrow B}. \lambda g^{A \rightarrow A}. \lambda x^A. f(g x) : (\Box A \rightarrow B) \multimap \Box(A \rightarrow A) \rightarrow \Box A \rightarrow B$$

Safe recursion in SLR

- ▶ SLR comes with a safe recursor:

$$\text{saferec}_A : \square N \rightarrow A \rightarrow (\square N \rightarrow A \rightarrow A) \rightarrow A$$

- ▶ Its semantics is:

$$\text{saferec}_A 0 g h = g$$

$$\text{saferec}_A n g h = h n (\text{saferec}_A \lfloor n/2 \rfloor g h) \quad \text{when } n \neq 0$$

- ▶ Example: $sq \ x$ computes a value in the order of x^2 :

$$sq : \square N \rightarrow N = \lambda x^N . \text{saferec}_N x 1 (\lambda y^N . \lambda q^N . s_0(s_0 q))$$

- ▶ We can iterate sq : $\lambda x^N . sq(sq x) : \square N \rightarrow N$

- ▶ But the following exponentially-growing function is ill-typed:

$$\lambda x^N . \text{saferec}_N x 1 (\lambda y^N . \lambda x^N . sq x)$$

Relation between Bellantoni and Cook's class and SLR

1. Define the category \mathcal{C} of Bellantoni and Cook's functions.
 - ▶ Objects are pair of natural numbers
(meant to be numbers of normal and safe arguments)
 - ▶ A morphisms from (m, n) to (m', n') is a pair of Bellantoni and Cook's functions $((f_1^{m,0}, \dots, f_{m'}^{m,0}), (f_1^{m,n}, \dots, f_{n'}^{m,n}))$
2. Embed \mathcal{C} in the category $\hat{\mathcal{C}}$ of presheaves over \mathcal{C}
(i.e., the category of contravariant functors from \mathcal{C} to Set).

It is a standard application of Yoneda Lemma to embed first-order functions into a model of a higher-order typed language.

$$\begin{aligned} \llbracket N \rrbracket &= \text{Hom}_{\mathcal{C}}(-, (0, 1)) \\ \llbracket A \rightarrow B \rrbracket &= \llbracket A \multimap B \rrbracket = \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket \\ \llbracket \Box A \rightarrow B \rrbracket &= \llbracket \Box A \multimap B \rrbracket = \Box \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket \end{aligned}$$

3. **Theorem** (Hofmann) There is a bijection between the set of natural transformations from $\llbracket N \rrbracket^m \times \llbracket N \rrbracket^n$ to $\llbracket N \rrbracket$ and the set of $(m+n)$ -ary functions in Bellantoni and Cook's class.

Oracle SLR

- ▶ (Mitchell et al., 1998) extend SLR with a 0,1-valued oracle.
Another standard categorical technique is used:
The Kleisli construction
- ▶ OSLR characterizes probabilistic polytime functions.
- ▶ The oracle is a kind of side-effect:
The resulting value depend of the evaluation strategy.
- ▶ It makes difficult to build a logic upon the language.
- ▶ A standard solution used by (Zhang, 2009) is to hide the side-effect with a monadic type.

Introduction

Characterizing polytime functions: From Cobham to Zhang

Computational SLR

Computational SLR

- ▶ CSLR (Zhang, 2009) extend OSLR with monadic types:

$$\tau ::= \dots \mid T\tau$$

They distinguish at type level between deterministic and probabilistic computations.

- ▶ The type N is replaced by the type Bits for bitstrings.
 - ▶ 0 and 00 (for example) are different bitstrings in CSLR but were identified to the number 0 in SLR.
- ▶ Expressions are extended with probabilistic computations:

$$e ::= \dots \mid \text{rand} \mid \text{return}(e) \mid x \stackrel{\$}{\leftarrow} e_1; e_2$$

An example of CSLR function

- ▶ To ease the reading of CSLR terms, we use syntactic sugar
 - ▶ In particular, a term F defined recursively by $\lambda n. \text{rec}_\tau(e_1, e_2, n)$ is written:

$$F \stackrel{\text{def}}{=} \lambda n. \text{if } n \stackrel{?}{=} \text{nil} \text{ then } e_1 \text{ else } e_2(n, F(\mathbf{tail}(n))),$$

- ▶ The random bitstring generation:

$$\begin{aligned} \mathbf{rs} \stackrel{\text{def}}{=} & \lambda n. \text{if } (n \stackrel{?}{=} \text{nil}) \\ & \text{then return}(\text{nil}) \\ & \text{else } b \stackrel{\$}{\leftarrow} \text{rand}; u \stackrel{\$}{\leftarrow} \mathbf{rs}(\mathbf{tail}(n)); \text{return}(b \bullet u) \end{aligned}$$

- ▶ Input: a bitstring
Output: a random bitstring of the same length
- ▶ One can check that $\vdash \mathbf{rs} : \square \text{Bits} \rightarrow \text{TBits}$

Pseudo-uniform sampling

- ▶ In theoretical proofs, arbitrary uniform sampling are used.
(for example, $x \in_R \mathbb{Z}_n^*$)
 - ▶ But in practice, computers are based on binary digits:
The cardinal of a uniform distribution has to be a power of 2.
 - ▶ The complexity class PPT is defined with probabilistic Turing machines.
But probabilistic Turing machines deal with random bits only.
- ▶ Pseudo-uniform sampling in CSLR:

```
zrand  $\stackrel{\text{def}}{=} \lambda n. \lambda t. \text{if } t \stackrel{?}{=} \text{nil}$   
    then return( $0^{|n|}$ )  
    else  $v \stackrel{\$}{\leftarrow} \mathbf{rs}(n)$ ;  
        if  $v \geq n$   
        then zrand( $n, \mathbf{tail}(t)$ )  
        else return( $v$ )
```

Tries to sample a value between 0 and n .

After a timeout $|t|$, it returns the default value $0^{|n|}$.

Indistinguishability

- ▶ Two CSLR terms f_1 and f_2 are **computationally indistinguishable** (written as $f_1 \simeq f_2$) if for every term \mathcal{A} such that $\vdash \mathcal{A} : \square\text{Bits} \rightarrow \tau \rightarrow \text{TBits}$ and every positive polynomial P , there exists some $N \in \mathbb{N}$ such that for all bitstring η with $|\eta| \geq N$

$$|\Pr[\llbracket \mathcal{A}(\eta, f_1(\eta)) \rrbracket \rightsquigarrow 1] - \Pr[\llbracket \mathcal{A}(\eta, f_2(\eta)) \rrbracket \rightsquigarrow 1]| < \frac{1}{P(|\eta|)}$$

↓

- ▶ Two CSLR terms g_1 and g_2 are **game indistinguishable** (written as $g_1 \approx g_2$) if for every term \mathcal{A} such that $\vdash \mathcal{A} : \square\text{Bits} \rightarrow \text{T}\tau$, and every positive polynomial P , there exists some $N \in \mathbb{N}$ such that for all bitstring η with $|\eta| \geq N$,

$$|\Pr[\llbracket g_1(\eta, \mathcal{A}) \rrbracket \rightsquigarrow 1] - \Pr[\llbracket g_2(\eta, \mathcal{A}) \rrbracket \rightsquigarrow 1]| < \frac{1}{P(|\eta|)}$$

Security as game indistinguishability

- ▶ A public-key encryption scheme $(\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ is *semantically secure* if:

$$\begin{aligned} \lambda\eta . \lambda(\mathcal{A}, \mathcal{A}') . (pk, sk) &\stackrel{\$}{\leftarrow} \mathbf{KG}(\eta); \\ (m_0, m_1, \mathcal{A}') &\stackrel{\$}{\leftarrow} \mathcal{A}(\eta, pk); \\ b &\stackrel{\$}{\leftarrow} \text{rand}; \\ c &\stackrel{\$}{\leftarrow} \mathbf{Enc}(\eta, m_b, pk); \\ b' &\stackrel{\$}{\leftarrow} \mathcal{A}'(c); \\ \text{return}(b &\stackrel{?}{=} b) \end{aligned} \quad \approx \lambda\eta . \lambda\mathcal{A} . \text{rand}$$

- ▶ An SLR-function F is *left-bit unpredictable* if:

$$\begin{aligned} \lambda\eta . \lambda\mathcal{A} . s &\stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta); \\ u &\leftarrow F(\eta, s); \\ b &\stackrel{\$}{\leftarrow} \mathcal{A}(\eta, \mathbf{tail}(u)); \\ \text{return}(b &\stackrel{?}{=} \mathbf{head}(u)) \end{aligned} \quad \approx \lambda\eta . \lambda\mathcal{A} . \text{rand}$$

DDH assumption as computational indistinguishability

- ▶ We cannot write DDH in CSLR because it involves arbitrary uniform choices.
- ▶ Instead we define DDH-Bits:

$$DDHBL \simeq DDHBR$$

where

$$DDHBL \stackrel{\text{def}}{=} \lambda \eta . x \stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta);$$
$$y \stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta);$$
$$\text{return}(\gamma^x, \gamma^y, \gamma^{xy})$$

$$DDHBR \stackrel{\text{def}}{=} \lambda \eta . x \stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta);$$
$$y \stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta);$$
$$z \stackrel{\$}{\leftarrow} \mathbf{zrand}(q, \eta);$$
$$\text{return}(\gamma^x, \gamma^y, \gamma^z)$$

- ▶ DDH-bits holds when the DDH assumption holds.

Equational proof systems

- ▶ An equational proof system for computational indistinguishability is defined and proved sound in (Zhang, 2009).
- ▶ We extend it with rules for game indistinguishability.
- ▶ The formal proof of the semantic security of ElGamal fits in one LNCS page.

- ▶ CSLR does not allow for:
 - ▶ superpolynomial-time computations, and
 - ▶ arbitrary uniform samplings.
- ▶ These restrictions make sense for cryptographic constructions and the adversary.
- ▶ But not necessary for games and assumptions.

They are just idealized constructions that are used to define security notions but are not meant to make their way into implementations.
- ▶ CSLR+ extend CSLR with:
 - ▶ arbitrary uniform sampling primitive, and
 - ▶ constant for primitives functions (including superpolynomial-time computations).

Conclusions

- ▶ We have proposed a language to formally write game-based proofs.
 - ▶ Game transformations are purely syntactic.
 - ▶ The type systems ensures that attackers are PPT.
- ▶ Possible future work include:
 - ▶ Implement it.
already in progress by Zhang
 - ▶ Show its usability with more examples of game-based proofs.
An implementation would make this easier.
 - ▶ Certify it in a proof assistant
A formalization of Bellantoni and Cook's class in Coq is in progress (with Sylvain Heraud)