A Calculus for Game-Based Security Proofs[1]

David NOWAK

Research Center for Information Security, AIST, Tokyo

FAIS Organized Session, JSIAM 2010
September 6, 2010

[1] Joint 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** $\pi^n_j(x_1, \ldots, x_n) = x_j$
  iii. **Successors** $s_i(x) = xi \quad$ for $i \in \{0, 1\}$
  iv. **smash** $2^{|x_1|.|x_2|}$
  v. **Recursion** $\quad f(0, \overline{x}) = g(\overline{x})$
  $f(yi, \overline{x}) = h_i(y, \overline{x}, f(y, \overline{x})) \quad$ for $yi \neq 0$
  $|f(y, \overline{x})| <= |j(y, \overline{x})|$
  where g, h_0, h_1 and j are in this class
  vi. **Composition** $\quad f(\overline{x}) = h(\overline{r}(\overline{x}))$
  where h and $\overline{r}$ 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):
  i. **Constant** 0
  ii. **Projection** $\pi_j^{m,n}(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+n}) = x_j$
  iii. **Successors** $s_i(; a) = ai \quad$ for $i \in \{0, 1\}$
  iv. **Predecessor** $p(; 0) = 0 \quad$ and $\quad p(; ai) = a$
  v. **Recursion** $f(0, \overline{x}; \overline{a}) = g(\overline{x}; \overline{a})$
  $f(yi, \overline{x}; \overline{a}) = h_i(y, \overline{x}; \overline{a}, f(y, \overline{x}; \overline{a}))$ for $yi \neq 0$
  where g, h_0 and h_1 are in this class
  vi. **Composition** $f(\overline{x}; \overline{a}) = h(\overline{r}(\overline{x};); \overline{t}(\overline{x}; \overline{a}))$
  where h, $\overline{r}$ and $\overline{t}$ 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 $\overline{a}$ and all w, $|w| \geq p_f(\overline{|a|})$ implies $f(\overline{a}) = f'(w; \overline{a})$.

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

Proof. Let $f(\overline{x})$ be a function on Cobham's class.
Let f' and p_f be obtained using the above lemma.
One can construct a $b(\overline{x};)$ such that $|b(\overline{x})| \geq p_f(\overline{|x|})$ and obtain $f''(\overline{x}) = f'(b(\overline{x};); \overline{x})$ 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 $\overline{x}$ and $\overline{a}$, $|f(\overline{x};\overline{a})| \leq q_f(|\overline{x}|) + max(|\overline{a}|)$.

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:

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

  - It denotes a function f whose size is bounded:

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

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

Examples of SLR functions and their inferred types

$$
\begin{array}{rcl}
\lambda x^{A}.x & : & A \multimap A \\
 & \vdots & \\
\lambda f^{A \to B}.\lambda x^{A}.f\ x & : & (A \to B) \multimap A \to B \\
 & \vdots & \\
\lambda f^{\Box A \to B}.\lambda x^{A}.f\ x & : & (\Box A \to B) \multimap \Box A \to B \\
 & \vdots & \\
\lambda f^{\Box A \to B}.\lambda g^{A \to A}.\lambda x^{A}.f(g\ x) & : & (\Box A \to B) \multimap \Box(A \to A) \to \Box A \to B
\end{array}
$$

Safe recursion in SLR

- SLR comes with a safe recursor:

$$\mathrm{saferec}_A \quad : \quad \Box N \to A \to (\Box N \to A \to A) \to A$$

- Its semantics is:

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

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

$$sq : \Box N \to N \quad = \quad \lambda x^N.\mathrm{saferec}_N\ x\ 1\ (\lambda y^N.\lambda q^N.s_0(s_0 q))$$

- We can iterate sq: $\lambda x^N.sq(sqx) \quad : \quad \Box N \to N$
- But the following exponentially-growing function is ill-typed:

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

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 $\left((f_1^{m,0}, \ldots, f_{m'}^{m,0}), (f_1^{m,n}, \ldots, f_{n'}^{m,n})\right)$
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.

$$\begin{aligned} [\![N]\!] &= \mathrm{Hom}_{\mathcal{C}}(-, (0,1)) \\ [\![A \to B]\!] = [\![A \multimap B]\!] &= [\![A]\!] \Rightarrow [\![B]\!] \\ [\![\Box A \to B]\!] = [\![\Box A \multimap B]\!] &= \Box[\![A]\!] \Rightarrow [\![B]\!] \end{aligned}$$

3. **Theorem** (Hofmann) There is a bijection between the set of natural transformations from $[\![N]\!]^m \times [\![N]\!]^n$ to $[\![N]\!]$ 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:

$$\tau ::= \cdots \mid \mathsf{T}\tau$$

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 CLSR but were identified to the number 0 in SLR.

- Expressions are extended with probabilistic computations:

$$e ::= \cdots \mid \mathsf{rand} \mid \mathsf{return}(e) \mid x \xleftarrow{\$} e_1;\ e_2$$

An example of CSLR function

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

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

- The random bitstring generation:

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

- Input: a bitstring
  Output: a random bitstring of the same length
- One can check that $\vdash \textbf{\textit{rs}} : \square\mathrm{Bits} \longrightarrow \mathrm{TBits}$

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:

$$
\begin{aligned}
\textbf{\textit{zrand}} \stackrel{\text{def}}{=} \lambda n\,.\,\lambda t\,.\,&\texttt{if } t \stackrel{?}{=} \texttt{nil}\\
&\texttt{then return}(0^{|n|})\\
&\texttt{else } v \stackrel{\$}{\leftarrow} \textbf{\textit{rs}}(n);\\
&\qquad \texttt{if } v \geq n\\
&\qquad \texttt{then } \textbf{\textit{zrand}}(n, \textbf{\textit{tail}}(t))\\
&\qquad \texttt{else return}(v)
\end{aligned}
$$

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 $\vdash \mathcal{A} : \Box\mathsf{Bits} \to \tau \to \mathsf{TBits}$ and every positive polynomial P, there exists some $N \in \mathbb{N}$ such that for all bitstring η with $|\eta| \geq N$*

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

$$\Downarrow$$

- *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 $\vdash \mathcal{A} : \Box\mathsf{Bits} \to \mathsf{T}\tau$, and every positive polynomial P, there exists some $N \in \mathbb{N}$ such that for all bitstring η with $|\eta| \geq N$,*

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

Security as game indistinguishability

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

$$\lambda\eta\,.\,\lambda(\mathcal{A},\mathcal{A}')\,.\,(pk,sk) \xleftarrow{\$} \boldsymbol{KG}(\eta);\; (m_0,m_1,\mathcal{A}') \xleftarrow{\$} \mathcal{A}(\eta,pk);\; b \xleftarrow{\$} \mathtt{rand};\; c \xleftarrow{\$} \boldsymbol{Enc}(\eta,m_b,pk);\; b' \xleftarrow{\$} \mathcal{A}'(c);\; \mathtt{return}(b' \stackrel{?}{=} b) \approx \lambda\eta\,.\,\lambda\mathcal{A}\,.\,\mathtt{rand}$$

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

$$\lambda\eta\,.\,\lambda\mathcal{A}\,.\,s \xleftarrow{\$} \boldsymbol{zrand}(q,\eta);\; u \leftarrow F(\eta,s);\; b \xleftarrow{\$} \mathcal{A}(\eta,\boldsymbol{tail}(u));\; \mathtt{return}(b \stackrel{?}{=} \boldsymbol{head}(u)) \approx \lambda\eta\,.\,\lambda\mathcal{A}\,.\,\mathtt{rand}$$

DDH assumption as computational indistinguishability

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

$$\textbf{\textit{DDHBL}} \simeq \textbf{\textit{DDHBR}}$$

where

$$\begin{aligned}
\textbf{\textit{DDHBL}} \stackrel{\text{def}}{=} \quad & \lambda\eta\,. \quad x \xleftarrow{\$} \textbf{\textit{zrand}}(q,\eta); \\
& \qquad\quad y \xleftarrow{\$} \textbf{\textit{zrand}}(q,\eta); \\
& \qquad\quad \texttt{return}(\gamma^x, \gamma^y, \gamma^{xy})
\end{aligned}$$

$$\begin{aligned}
\textbf{\textit{DDHBR}} \stackrel{\text{def}}{=} \quad & \lambda\eta\,. \quad x \xleftarrow{\$} \textbf{\textit{zrand}}(q,\eta); \\
& \qquad\quad y \xleftarrow{\$} \textbf{\textit{zrand}}(q,\eta); \\
& \qquad\quad z \xleftarrow{\$} \textbf{\textit{zrand}}(q,\eta); \\
& \qquad\quad \texttt{return}(\gamma^x, \gamma^y, \gamma^z)
\end{aligned}$$

- 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+

- 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)