Secure Compilation by Approximate Back-Translation

Dominique Devriese[1] Marco Patrignani[2] Frank Piessens[1]

[1]iMinds-DistriNet, Dept. Computer Science, KU Leuven, Belgium
first.last@cs.kuleuven.be

[2]MPI-SWS Saarbrücken, Germany
first.last@mpi-sws.org

compiler correctness direction $\Uparrow$

$$\mathbf{t_1} \overset{?}{\simeq}_{ctx} \mathbf{t_2}$$

$$\begin{array}{ccccc}
 & \mathfrak{C}[\mathbf{t_1}]\Downarrow & \overset{?}{\Rightarrow} & \mathfrak{C}[\mathbf{t_2}]\Downarrow & \\
\begin{array}{c}\mathfrak{C} \approx [\![\mathfrak{C}]\!] \\ \mathbf{t_1} \approx [\![\mathbf{t_1}]\!]\end{array} & \Downarrow (1) & & (3) \Uparrow & \begin{array}{c}\mathfrak{C} \approx [\![\mathfrak{C}]\!] \\ \mathbf{t_2} \approx [\![\mathbf{t_2}]\!]\end{array} \\
 & [\![\mathfrak{C}]\!]\left[[\![\mathbf{t_1}]\!]\right]\Downarrow & \overset{(2)}{\Rightarrow} & [\![\mathfrak{C}]\!]\left[[\![\mathbf{t_2}]\!]\right]\Downarrow &
\end{array}$$

$$[\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!]$$

compiler security direction $\Downarrow$

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle [\mathbf{t_1}] \Downarrow \quad \Rightarrow \quad \langle\!\langle \mathfrak{C} \rangle\!\rangle [\mathbf{t_2}] \Downarrow$$

$$(2)$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C} \qquad \Uparrow (1) \qquad (3) \Downarrow \qquad \langle\!\langle \mathfrak{C} \rangle\!\rangle \approx \mathfrak{C}$$

$$\mathbf{t_1} \approx [\![\mathbf{t_1}]\!] \qquad\qquad\qquad\qquad \mathbf{t_2} \approx [\![\mathbf{t_2}]\!]$$

$$\mathfrak{C}\left[[\![\mathbf{t_1}]\!]\right] \Downarrow \quad \overset{?}{\Rightarrow} \quad \mathfrak{C}\left[[\![\mathbf{t_2}]\!]\right] \Downarrow$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{ctx} [\![\mathbf{t_2}]\!]$$

$$\mathbf{t_1} \simeq_{ctx} \mathbf{t_2}$$

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle_n [\mathbf{t_1}] \Downarrow\_ \quad \overset{}{\underset{(2)}{\Rightarrow}} \quad \langle\!\langle \mathfrak{C} \rangle\!\rangle_n [\mathbf{t_2}] \Downarrow\_$$

$$\begin{array}{l} \langle\!\langle \mathfrak{C} \rangle\!\rangle_n \gtrsim_n \mathfrak{C} \\ \mathbf{t_1} \gtrsim\_ [\![\mathbf{t_1}]\!] \end{array} \quad \Uparrow(1) \qquad (3)\Downarrow \quad \begin{array}{l} \langle\!\langle \mathfrak{C} \rangle\!\rangle_n \lesssim\_ \mathfrak{C} \\ \mathbf{t_2} \lesssim\_ [\![\mathbf{t_2}]\!] \end{array}$$

$$\mathfrak{C}\left[[\![\mathbf{t_1}]\!]\right] \Downarrow_n \quad \overset{?}{\Rightarrow} \quad \mathfrak{C}\left[[\![\mathbf{t_2}]\!]\right] \Downarrow\_$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{ctx} [\![\mathbf{t_2}]\!]$$

$\Downarrow$ approx. compiler security

We devise a compiler between STLC and ULC.

$$[\![\mathbf{t}]\!]^{\mathcal{S}}_{\mathcal{T}} = \mathsf{protect}_{\tau}\,\mathtt{erase}(\mathbf{t})$$

$$\mathsf{protect}_{\mathtt{Unit}} \stackrel{\text{def}}{=} \lambda x.\, x \qquad \mathsf{protect}_{\mathtt{Bool}} \stackrel{\text{def}}{=} \lambda x.\, x$$

$$\mathsf{protect}_{\tau_1 \times \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \langle \mathsf{protect}_{\tau_1}\ y.1, \mathsf{protect}_{\tau_2}\ y.2 \rangle$$

$$\mathsf{protect}_{\tau_1 \uplus \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \mathrm{case}\ y\ \mathrm{of} \left| \begin{array}{l} \mathrm{inl}\ x \mapsto \mathrm{inl}\ (\mathsf{protect}_{\tau_1}\ x) \\ \mathrm{inr}\ x \mapsto \mathrm{inr}\ (\mathsf{protect}_{\tau_2}\ x) \end{array} \right.$$

$$\mathsf{protect}_{\tau_1 \to \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \lambda x. \mathsf{protect}_{\tau_2}\ (y\ (\mathsf{confine}_{\tau_1}\ x))$$

$$\mathsf{confine}_{\mathtt{Unit}} \stackrel{\text{def}}{=} \lambda y.\, (y; \mathtt{unit})$$

$$\mathsf{confine}_{\mathtt{Bool}} \stackrel{\text{def}}{=} \lambda y.\, \mathrm{if}\ y\ \mathrm{then}\ \mathtt{true}\ \mathrm{else}\ \mathtt{false}$$

$$\mathsf{confine}_{\tau_1 \times \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \langle \mathsf{confine}_{\tau_1}\ y.1, \mathsf{confine}_{\tau_2}\ y.2 \rangle$$

$$\mathsf{confine}_{\tau_1 \uplus \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \mathrm{case}\ y\ \mathrm{of} \left| \begin{array}{l} \mathrm{inl}\ x \mapsto \mathrm{inl}\ (\mathsf{confine}_{\tau_1}\ x) \\ \mathrm{inr}\ x \mapsto \mathrm{inr}\ (\mathsf{confine}_{\tau_2}\ x) \end{array} \right.$$

$$\mathsf{confine}_{\tau_1 \to \tau_2} \stackrel{\text{def}}{=} \lambda y.\, \lambda x.\, \mathsf{confine}_{\tau_2}\ (y\ (\mathsf{protect}_{\tau_1}\ x))$$

We need a type for back-translated terms:

$$\mathrm{UVal}_0 \triangleq \texttt{Unit}$$

$$\mathrm{UVal}_{n+1} \triangleq \begin{array}{l}\texttt{Unit} \uplus \texttt{Unit} \uplus \texttt{Bool} \uplus (\mathrm{UVal}_n \times \mathrm{UVal}_n) \uplus \\ \quad (\mathrm{UVal}_n \uplus \mathrm{UVal}_n) \uplus (\mathrm{UVal}_n \to \mathrm{UVal}_n)\end{array}$$

We need a type for back-translated terms:

$$\mathrm{UVal}_0 \triangleq \mathtt{Unit}$$

$$\mathrm{UVal}_{n+1} \triangleq \begin{array}{l} \mathtt{Unit} \uplus \mathtt{Unit} \uplus \mathtt{Bool} \uplus (\mathrm{UVal}_n \times \mathrm{UVal}_n) \uplus \\ \quad (\mathrm{UVal}_n \uplus \mathrm{UVal}_n) \uplus (\mathrm{UVal}_n \to \mathrm{UVal}_n) \end{array}$$

$$\begin{aligned}
\mathtt{in}_{\mathrm{unk};n} &: \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\mathtt{Unit};n} &: \mathtt{Unit} \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\mathtt{Bool};n} &: \mathtt{Bool} \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\times;n} &: (\mathrm{UVal}_n \times \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\uplus;n} &: (\mathrm{UVal}_n \uplus \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\to;n} &: (\mathrm{UVal}_n \to \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1}
\end{aligned}$$

We need a type for back-translated terms:

$$\mathrm{UVal}_0 \triangleq \mathtt{Unit}$$

$$\mathrm{UVal}_{n+1} \triangleq \begin{array}{l}\mathtt{Unit} \uplus \mathtt{Unit} \uplus \mathtt{Bool} \uplus (\mathrm{UVal}_n \times \mathrm{UVal}_n) \uplus \\ \quad (\mathrm{UVal}_n \uplus \mathrm{UVal}_n) \uplus (\mathrm{UVal}_n \to \mathrm{UVal}_n)\end{array}$$

$$\begin{aligned}
\mathtt{in}_{\mathrm{unk};n} &: \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\mathtt{Unit};n} &: \mathtt{Unit} \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\mathtt{Bool};n} &: \mathtt{Bool} \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\times;n} &: (\mathrm{UVal}_n \times \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\uplus;n} &: (\mathrm{UVal}_n \uplus \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1} \\
\mathtt{in}_{\to;n} &: (\mathrm{UVal}_n \to \mathrm{UVal}_n) \to \mathrm{UVal}_{n+1}
\end{aligned}$$

$$\mathtt{case}_{\tau;n} : \mathrm{UVal}_{n+1} \to \tau$$

We need to back-translate terms:

$$
\begin{aligned}
\mathrm{emulate}_{\mathsf{n}}(\mathtt{t}) &: \mathrm{UVal}_{\mathsf{n}} \\
\mathrm{emulate}_{\mathsf{n}}(\mathtt{unit}) &\triangleq \mathrm{downgrade}_{\mathsf{n};1}\ (\mathtt{in}_{\mathtt{Unit};\mathtt{n}}\ \mathtt{unit}) \\
\mathrm{emulate}_{\mathsf{n}}(\mathtt{true}) &\triangleq \mathrm{downgrade}_{\mathsf{n};1}\ (\mathtt{in}_{\mathtt{Bool};\mathtt{n}}\ \mathtt{true}) \\
\mathrm{emulate}_{\mathsf{n}}(\mathtt{false}) &\triangleq \mathrm{downgrade}_{\mathsf{n};1}\ (\mathtt{in}_{\mathtt{Bool};\mathtt{n}}\ \mathtt{false}) \\
\mathrm{emulate}_{\mathsf{n}}(\mathsf{x}) &\triangleq \mathbf{x} \\
\mathrm{emulate}_{\mathsf{n}}(\lambda \mathsf{x}.\,\mathtt{t}) &\triangleq \mathrm{downgrade}_{\mathsf{n};1}\ (\mathtt{in}_{\to;\mathtt{n}}\ (\lambda \mathbf{x} : \mathrm{UVal}_{\mathsf{n}}.\,\mathrm{emulate}_{\mathsf{n}}(\mathtt{t}))) \\
&\dots
\end{aligned}
$$

We need to relate back-translated terms:

$$\mathcal{V}[\![\mathtt{EmulDV}_{\mathtt{0;p}}]\!]^{\rho}_{\square} \triangleq \{(\underline{\mathsf{W}}, \mathbf{v}, \mathsf{v}) \mid \mathbf{v} = \mathtt{unit} \text{ and } p = \mathtt{imprecise}\}$$

$$\mathcal{V}[\![\mathtt{EmulDV}_{\mathtt{n+1;p}}]\!]^{\rho}_{\square} \triangleq$$

$$\left\{ (\underline{\mathsf{W}}, \mathbf{v}, \mathsf{v}) \;\middle|\; \begin{array}{l} \mathbf{v} \in \mathbf{oftype}(\mathrm{UVal}_{n+1}) \text{ and one of the following holds:} \\ \qquad \begin{cases} \mathbf{v} = \mathtt{in}_{\mathrm{unk;n}} \text{ and } p = \mathtt{imprecise} \\ \exists \mathbf{v}'.\, \mathbf{v} = \mathtt{in}_{\mathtt{Unit;n}}\ \mathbf{v}' \text{ and } (\underline{\mathsf{W}}, \mathbf{v}', \mathsf{v}) \in \mathcal{V}[\![\mathtt{Unit}]\!]^{\rho}_{\square} \\ \exists \mathbf{v}'.\, \mathbf{v} = \mathtt{in}_{\mathtt{Bool;n}}\ \mathbf{v}' \text{ and } (\underline{\mathsf{W}}, \mathbf{v}', \mathsf{v}) \in \mathcal{V}[\![\mathtt{Bool}]\!]^{\rho}_{\square} \\ \exists \mathbf{v}'.\, \mathbf{v} = \mathtt{in}_{\times} \mathbf{n}\ \mathbf{v}' \text{ and} \\ \quad (\underline{\mathsf{W}}, \mathbf{v}', \mathsf{v}) \in \mathcal{V}[\![\mathtt{EmulDV}_{\mathtt{n;p}} \times \mathtt{EmulDV}_{\mathtt{n;p}}]\!]^{\rho}_{\square} \\ \exists \mathbf{v}'.\, \mathbf{v} = \mathtt{in}_{\uplus} \mathbf{n}\ \mathbf{v}' \text{ and} \\ \quad (\underline{\mathsf{W}}, \mathbf{v}', \mathsf{v}) \in \mathcal{V}[\![\mathtt{EmulDV}_{\mathtt{n;p}} \uplus \mathtt{EmulDV}_{\mathtt{n;p}}]\!]^{\rho}_{\square} \\ \exists \mathbf{v}'.\, \mathbf{v} = \mathtt{in}_{\rightarrow} \mathbf{n}\ \mathbf{v}' \text{ and} \\ \quad (\underline{\mathsf{W}}, \mathbf{v}', \mathsf{v}) \in \mathcal{V}[\![\mathtt{EmulDV}_{\mathtt{n;p}} \rightarrow \mathtt{EmulDV}_{\mathtt{n;p}}]\!]^{\rho}_{\square} \end{cases} \end{array} \right\}$$

We prove these results:

This statement

$$\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\tau;n}[\mathbf{t}]$$

$$\square_n$$

$$\mathfrak{C}[[\![\mathbf{t}]\!]]$$

expands to this

$$\overbrace{\mathrm{emulate}_n(\mathfrak{C})[\ \mathtt{inject}_{\tau;n}}^{\langle\!\langle \mathfrak{C} \rangle\!\rangle_{\tau;n}}\ \mathbf{t}\ ]$$

$$\square_n \qquad \square_n \qquad \square_n$$

$$\mathfrak{C}[\ \underbrace{\mathrm{protect}_{\tau}\ \mathtt{erase}(\mathbf{t})}_{[\![\mathbf{t}]\!]}\ ]$$

What now?

What now?

We devise a compiler between SYSF and LSEAL

WIP:

- some correctness direction proofs
- definition of UVal
- EMulDV and its logical relation
- emulate for seal-related operators
- all security direction proofs