Declassification aware product construction

ctverif

No Institute Given

$$
\begin{aligned}
\mathrm{SelfComp}(s) &= \mathsf{okSC} := \mathsf{true}; \mathsf{countL} := 0; \mathsf{countR} := 0;\\
&\quad \mathrm{Mirror}^{1}(s); \mathrm{Mirror}^{2}(s);\\
&\quad \mathsf{okSC} := \mathsf{okSC} \wedge (\mathsf{countL} = \mathsf{countR})
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Mirror}^{lr}(\mathsf{skip}) &= \mathsf{skip}\\
\mathrm{Mirror}^{lr}(\mathsf{assume}\ e) &= \mathsf{assume}\ e\{lr\}\\
\mathrm{Mirror}^{lr}(\mathsf{assert}\ e) &= \mathsf{assert}\ e\{lr\}\\
\mathrm{Mirror}^{lr}(x := e) &= \mathrm{ProcLeak}^{lr}(L(x\{lr\})); \mathrm{ProcLeak}^{lr}(L(e\{lr\})); x\{lr\} := e\{lr\}\\
\mathrm{Mirror}^{lr}(s_1; s_2) &= \mathrm{Mirror}^{lr}(s_1); \mathrm{Mirror}^{lr}(s_2)\\
\mathrm{Mirror}^{lr}(\mathsf{if}\ b\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2) &= \mathrm{ProcLeak}^{lr}(L(b\{lr\})); \mathrm{ProcLeak}^{lr}(b\{lr\});\\
&\quad \mathsf{if}\ b\{lr\}\ \mathsf{then}\ \mathrm{Mirror}^{lr}(s_1)\ \mathsf{else}\ \mathrm{Mirror}^{lr}(s_2)\\
\mathrm{Mirror}^{lr}(\mathsf{while}\ b\ \mathsf{do}\ s) &= \mathrm{ProcLeak}^{lr}(L(b\{lr\})); \mathrm{ProcLeak}^{lr}(b\{lr\});\\
&\quad \mathsf{while}\ b\{lr\}\ \mathsf{do}\ \{\mathrm{Mirror}^{lr}(s); \mathrm{ProcLeak}^{lr}(L(b\{lr\})); \mathrm{ProcLeak}^{lr}(b\{lr\})\}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{ProcLeak}^{lr}(\epsilon) &= \mathsf{skip}\\
\mathrm{ProcLeak}^{1}(x :: xs) &= \mathsf{leakL}[\mathsf{countL}] := x; \mathsf{countL} := \mathsf{countL} + 1; \mathrm{ProcLeak}^{1}(xs)\\
\mathrm{ProcLeak}^{2}(x :: xs) &= \mathsf{okSC} := \mathsf{okSC} \wedge (\mathsf{leakL}[\mathsf{countR}] = x); \mathsf{countR} := \mathsf{countR} + 1; \mathrm{ProcLeak}^{2}(xs)
\end{aligned}
$$

Fig. 1. Leakage-aware Self-Composition. $\mathsf{okSC}, \mathsf{countL}, \mathsf{countR}$ are fresh scalar variables and $\mathsf{leakL}[]$ is a fresh array variable. The program is *secure* if okSC is true at the final state. $L(-)$ denotes the preleakage of an expression.

$$
\begin{aligned}
\mathrm{FullProduct}(s) &= \textsf{ok} := \textsf{true}; \textsf{okSC} := \textsf{true};\\
&\ \textsf{assume}\ \mathrm{EqSeq}(I\{1\}, I\{2\}); \mathrm{ProductTrf}(s);\\
&\ \textsf{assume}\ \mathrm{EqSeq}(O\{1\}, O\{2\}); \textsf{assert}\ (\textsf{ok} \wedge \textsf{okSC})
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{ProductTrf}(\textsf{skip}) &= \textsf{skip}\\
\mathrm{ProductTrf}(\textsf{assume}\ e) &= \textsf{assume}\ e\{1\}; \textsf{assume}\ e\{2\}\\
\mathrm{ProductTrf}(\textsf{assert}\ e) &= \textsf{assert}\ e\{1\}; \textsf{assert}\ e\{2\}\\
\mathrm{ProductTrf}(x := e) &= \textsf{ok} := \textsf{ok} \wedge \mathrm{EqSeq}(L(x\{1\}), L(x\{2\}));\\
&\ \textsf{ok} := \textsf{ok} \wedge \mathrm{EqSeq}(L(e\{1\}), L(e\{2\}));\\
&\ x\{1\} := e\{1\}; x\{2\} := e\{2\}\\
\mathrm{ProductTrf}(s_1; s_2) &= \mathrm{ProductTrf}(s_1); \mathrm{ProductTrf}(s_2)\\
\mathrm{ProductTrf}(\textsf{if}\ b\ \textsf{then}\ s_1\ \textsf{else}\ s_2) &= \textsf{ok} := \textsf{ok} \wedge \mathrm{EqSeq}(L(b\{1\}), L(b\{2\}));\\
&\ \textsf{ok} := \textsf{ok} \wedge b\{1\} = b\{2\};\\
&\ \textsf{if}\ (\textsf{ok})\\
&\ \quad \textsf{then}\ \{\textsf{if}\ b\{1\}\ \textsf{then}\ \mathrm{ProductTrf}(s_1)\ \textsf{else}\ \mathrm{ProductTrf}(s_2)\}\\
&\ \quad \textsf{else}\ \mathrm{SelfComp}(\textsf{if}\ b\ \textsf{then}\ s_1\ \textsf{else}\ s_2)\\
\mathrm{ProductTrf}(\textsf{while}\ b\ \textsf{do}\ s) &= \textsf{ok} := \textsf{ok} \wedge \mathrm{EqSeq}(L(b\{1\}), L(b\{2\})); \textsf{ok} := \textsf{ok} \wedge b\{1\} = b\{2\};\\
&\ \textsf{while}\ (\textsf{ok} \wedge b\{1\})\\
&\ \quad \{\mathrm{ProductTrf}(s); \textsf{ok} := \textsf{ok} \wedge \mathrm{EqSeq}(L(b\{1\}), L(b\{2\})); \textsf{ok} := \textsf{ok} \wedge b\{1\} = b\{2\}\};\\
&\ \mathrm{SelfComp}(\textsf{while}\ (!\textsf{ok} \wedge b)\ \textsf{do}\ s)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{EqSeq}(\epsilon, \epsilon) &= \textsf{true}\\
\mathrm{EqSeq}(x :: xs, y :: ys) &= (x = y) \wedge \mathrm{EqSeq}(xs, ys)
\end{aligned}
$$

Fig. 2. Full-product construction.