1 *Overture* SYNTAX AND SEMANTICS

$$v \in \mathbb{F}_p,\ w \in \text{String},\ \iota \in \text{Clients} \subset \mathbb{N}$$

$$\begin{array}{lcll}
\varepsilon & ::= & \mathtt{r[}w\mathtt{]} \mid \mathtt{s[}w\mathtt{]} \mid \mathtt{m[}w\mathtt{]} \mid \mathtt{p[}w\mathtt{]} \mid & \textit{expressions} \\
 & & v \mid \varepsilon - \varepsilon \mid \varepsilon + \varepsilon \mid \varepsilon * \varepsilon & \\
x & ::= & \mathtt{r[}w\mathtt{]@}\iota \mid \mathtt{s[}w\mathtt{]@}\iota \mid \mathtt{m[}w\mathtt{]@}\iota \mid \mathtt{p[}w\mathtt{]} \mid \mathtt{out@}\iota & \textit{variables} \\
\pi & ::= & \mathtt{m[}w\mathtt{]@}\iota := \varepsilon\mathtt{@}\iota \mid \mathtt{p[}w\mathtt{]} := e\mathtt{@}\iota \mid \mathtt{out@}\iota := \varepsilon\mathtt{@}\iota \mid \pi ; \pi & \textit{protocols}
\end{array}$$

$$\begin{array}{rcl}
[\![\sigma, v]\!]_\iota & = & v \\
[\![\sigma, \varepsilon_1 + \varepsilon_2]\!]_\iota & = & [\![[\![\sigma, \varepsilon_1]\!]_\iota + [\![\sigma, \varepsilon_2]\!]_\iota]\!] \\
[\![\sigma, \varepsilon_1 - \varepsilon_2]\!]_\iota & = & [\![[\![\sigma, \varepsilon_1]\!]_\iota - [\![\sigma, \varepsilon_2]\!]_\iota]\!] \\
[\![\sigma, \varepsilon_1 * \varepsilon_2]\!]_\iota & = & [\![[\![\sigma, \varepsilon_1]\!]_\iota * [\![\sigma, \varepsilon_2]\!]_\iota]\!] \\
[\![\sigma, \mathtt{r[}w\mathtt{]}]\!]_\iota & = & \sigma(\mathtt{r[}w\mathtt{]@}\iota) \\
[\![\sigma, \mathtt{s[}w\mathtt{]}]\!]_\iota & = & \sigma(\mathtt{s[}w\mathtt{]@}\iota) \\
[\![\sigma, \mathtt{m[}w\mathtt{]}]\!]_\iota & = & \sigma(\mathtt{m[}w\mathtt{]@}\iota) \\
[\![\sigma, \mathtt{p[}w\mathtt{]}]\!]_\iota & = & \sigma(\mathtt{p[}w\mathtt{]})
\end{array}$$

$$(\sigma, x := \varepsilon\mathtt{@}\iota) \Rightarrow \sigma\{x \mapsto [\![\sigma, \varepsilon]\!]_\iota\} \qquad \frac{(\sigma_1, \pi_1) \Rightarrow \sigma_2 \qquad (\sigma_2, \pi_2) \Rightarrow \sigma_3}{(\sigma_1, \pi_1; \pi_2) \Rightarrow \sigma_3}$$

1.1 *Overture* Adversarial Semantics

$$\pi \quad ::= \quad \cdots \mid \mathtt{assert}(\varepsilon = \varepsilon)$$

$$\begin{array}{lcll}
(\sigma, x := \varepsilon\mathtt{@}\iota) & \Rightarrow_{\mathcal{A}} & \sigma\{x \mapsto [\![\sigma, \varepsilon]\!]_\iota\} & \iota \in H \\
(\sigma, x := \varepsilon\mathtt{@}\iota) & \Rightarrow_{\mathcal{A}} & \sigma\{x \mapsto [\![\mathit{rewrite}_{\mathcal{A}}(\sigma_C, \varepsilon)]\!]_\iota\} & \iota \in C
\end{array}$$

$$\begin{array}{rcll}
(\sigma, \mathtt{assert}(\varepsilon_1 = \varepsilon_2)\mathtt{@}\iota) & \Rightarrow_{\mathcal{A}} & \sigma & \text{if } [\![\sigma, \varepsilon_1]\!]_\iota = [\![\sigma, \varepsilon_2]\!]_\iota \text{ or } \iota \in C \\
(\sigma, \mathtt{assert}(\phi(\varepsilon))\mathtt{@}\iota) & \Rightarrow_{\mathcal{A}} & \bot & \text{if } \neg\phi(\sigma, [\![\sigma, \varepsilon]\!]_\iota)
\end{array}$$

$$\frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \sigma_2 \qquad (\sigma_2, \pi_2) \Rightarrow_{\mathcal{A}} \sigma_3}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \sigma_3} \qquad \frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \bot}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \bot}$$

$$\frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \sigma_2 \qquad (\sigma_2, \pi_2) \Rightarrow_{\mathcal{A}} \bot}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \bot}$$

2 *Overture* CONSTRAINT TYPING

2.1 Constraint Satisfiability Modulo Finite Fields

$$\begin{array}{lcl}
\phi & ::= & x \mid \phi + \phi \mid \phi - \phi \mid \phi * \phi \\
E & ::= & \phi \equiv \phi \mid E \wedge E
\end{array}$$

We write $E_1 \models E_2$ iff every model of E_1 is a model of E_2. Note that this relation is reflexive and transitive.

$$\lfloor x \rfloor = x \qquad \lfloor \varepsilon_1 + \varepsilon_2\mathtt{@}\iota \rfloor = \lfloor \varepsilon_2\mathtt{@}\iota \rfloor + \lfloor \varepsilon_1\mathtt{@}\iota \rfloor \qquad \lfloor \varepsilon_1 - \varepsilon_2\mathtt{@}\iota \rfloor = \lfloor \varepsilon_2\mathtt{@}\iota \rfloor - \lfloor \varepsilon_1\mathtt{@}\iota \rfloor$$

$$\lfloor \varepsilon_1 * \varepsilon_2\mathtt{@}\iota \rfloor = \lfloor \varepsilon_2\mathtt{@}\iota \rfloor * \lfloor \varepsilon_1\mathtt{@}\iota \rfloor$$

$$\lfloor \mathtt{OT}(\varepsilon_1@\iota_1, \varepsilon_2, \varepsilon_3)@\iota_2 \rfloor = (\lfloor \varepsilon_1@\iota_1 \rfloor \wedge \lfloor \varepsilon_3@\iota_2 \rfloor) \vee (\neg\lfloor \varepsilon_1@\iota_1 \rfloor \wedge \lfloor \varepsilon_2@\iota_2 \rfloor)$$

$$\lfloor x := \varepsilon@\iota \rfloor = x \equiv \lfloor \varepsilon@\iota \rfloor \qquad \lfloor \mathtt{assert}(\varepsilon_1 = \varepsilon_2)\iota \rfloor = \lfloor \varepsilon_1@\iota \rfloor \equiv \lfloor \varepsilon_2@\iota \rfloor \qquad \lfloor \pi_1; \pi_2 \rfloor = \lfloor \pi_1 \rfloor \wedge \lfloor \pi_2 \rfloor$$

The motivating idea is that we can interpret any protocol π as a set of equality constraints $\lfloor \pi \rfloor$ and use an SMT solver to verify properties relevant to correctness, confidentiality, and integrity. Further, we can leverage entailment relation is critical for efficiency– we can use annotations to obtain a weakened precondition for relevant properties. That is, given π, program annotations or other cues can be used to find a minimal E with $\lfloor \pi \rfloor \models E$ for verifying correctness and security.

2.1.1 *Example: Correctness of 3-Party Addition.*

$$\begin{array}{lcl}
\mathtt{m}[s1]@2 & := & (\mathtt{s[1]} - \mathtt{r[local]} - \mathtt{r}[x])@1 \\
\mathtt{m}[s1]@3 & := & \mathtt{r}[x]@1 \\
\mathtt{m}[s2]@1 & := & (\mathtt{s[2]} - \mathtt{r[local]} - \mathtt{r}[x])@2 \\
\mathtt{m}[s2]@3 & := & \mathtt{r}[x]@2 \\
\mathtt{m}[s3]@1 & := & (\mathtt{s[3]} - \mathtt{r[local]} - \mathtt{r}[x])@3 \\
\mathtt{m}[s3]@2 & := & \mathtt{r}[x]@3 \\
\mathtt{p[1]} & := & (\mathtt{r[local]} + \mathtt{m}[s2] + \mathtt{m}[s3])@1 \\
\mathtt{p[2]} & := & (\mathtt{m}[s1] + \mathtt{r[local]} + \mathtt{m}[s3])@2 \\
\mathtt{p[3]} & := & (\mathtt{m}[s1] + \mathtt{m}[s2] + \mathtt{r[local]})@3 \\
\mathtt{out}@1 & := & (\mathtt{p[1]} + \mathtt{p[2]} + \mathtt{p[3]})@1 \\
\mathtt{out}@2 & := & (\mathtt{p[1]} + \mathtt{p[2]} + \mathtt{p[3]})@2 \\
\mathtt{out}@3 & := & (\mathtt{p[1]} + \mathtt{p[2]} + \mathtt{p[3]})@3
\end{array}$$

Letting π be this protocol, we can verify correctness as:

$$\lfloor \pi \rfloor \models \mathtt{out}@3 \equiv \mathtt{s[1]}@1 + \mathtt{s[2]}@2 + \mathtt{s[3]}@3$$

2.2 Confidentiality Types

$$\begin{array}{lcl}
t & ::= & x \mid c(x, T) \\
T & \in & 2^t \\
\Gamma & ::= & \varnothing \mid \Gamma; x : T
\end{array}$$

Definition 2.1. $R_1; R_2 = R_1 \cup R_2$ iff $R_1 \cap R_2 = \varnothing$.

$$\textsc{DepTy}\quad \varnothing, E \vdash \phi : \mathit{vars}(\phi) \qquad \textsc{Encode}\quad \frac{E \models \phi \equiv \phi' \oplus \mathtt{r}[w]@\iota \qquad \oplus \in \{+, -\} \qquad R, E \vdash \phi' : T}{R; \{\mathtt{r}[w]@\iota\}, E \vdash \phi : \{c(\mathtt{r}[w]@\iota, T)\}}$$

$$\textsc{Send}\quad \frac{R, E \vdash \phi : T}{R, E \vdash x \equiv \phi : (x : T)} \qquad \textsc{Seq}\quad \frac{R_1, E \vdash \phi_1 : \Gamma_1 \qquad R_2, E \vdash \phi_2 : \Gamma_2}{R_1; R_2, E \vdash \phi_1 \wedge \phi_2 : \Gamma_1; \Gamma_2}$$

Definition 2.2. Given preprocessing predicate E_{pre} and protocol π we say $R, E \vdash E_{pre} \wedge \lfloor \pi \rfloor : \Gamma$ is *valid* iff it is derivable and $E_{pre} \wedge \lfloor \pi \rfloor \models E$.

$$\frac{\iota \in C}{\Gamma, C \vdash_{leak} \Gamma(\texttt{m[}w\texttt{]@}\iota)} \qquad \frac{\Gamma, C \vdash_{leak} T_1 \cup T_2}{\Gamma, C \vdash_{leak} T_1} \qquad \frac{\Gamma, C \vdash_{leak} \{\texttt{m[}w\texttt{]@}\iota\}}{\Gamma, C \vdash_{leak} \Gamma(\texttt{m[}w\texttt{]@}\iota)}$$

$$\frac{\Gamma, C \vdash_{leak} \{\texttt{r[}w\texttt{]@}\iota\} \qquad \Gamma, C \vdash_{leak} \{c(\texttt{r[}w\texttt{]@}\iota, T)\}}{\Gamma, C \vdash_{leak} T}$$

Theorem 2.3. *If $R, E \vdash E_{pre} \wedge \lfloor \pi \rfloor : \Gamma$ is valid and there exists no H, C and $\texttt{s[}w\texttt{]@}\iota$ for $\iota \in H$ with $\Gamma, C \vdash_{leak} \{\texttt{s[}w\texttt{]@}\iota\}$, then π satisfies gradual release.*

2.2.1 *Examples.*

```
m[s1]@2 := (s[1] - r[local] - r[x])@1
m[s1]@3 := r[x]@1

// m[s1]@2 : { c(r[x]@1, { c(r[local]@1, {s[1]@1} ) }
// m[s1]@3 : { r[x]@1 }
m[x]@1 := s2(s[x],-r[x],r[x])@2

// m[x]@1 == s[x]@2 + -r[x]@2
// m[x]@1 : { c(r[x]@2, { s[x]@2 }) }

m[y]@1 := OT(s[y]@1,-r[y],r[y])@2

// m[y]@1 == s[y]@1 + -r[y]@2
// m[y]@1 : { c(r[y]@2, { s[y]@1 }) }
```

2.3 Integrity Types

$$\begin{array}{rcl} \varsigma & ::= & \text{High} \mid \text{Low} \\ \Delta & ::= & \varnothing \mid \Delta; x : \iota \cdot V \end{array}$$

Value $\vdash_\iota v : \varnothing$ Secret $\vdash_\iota \texttt{s[}w\texttt{]} : \varnothing$ Rando $\vdash_\iota \texttt{r[}w\texttt{]} : \varnothing$ Mesg $\vdash_\iota \texttt{m[}w\texttt{]} : \{\texttt{m[}w\texttt{]@}\iota\}$ PubM $\vdash_\iota \texttt{p[}w\texttt{]} : \{\texttt{p[}w\texttt{]}\}$

Binop

$$\frac{\vdash_\iota \varepsilon_1 : V_1 \qquad \vdash_\iota \varepsilon_2 : V_2 \qquad \oplus \in \{+, -, *\}}{\vdash_\iota \varepsilon_1 \oplus \varepsilon_2 : V_1 \cup V_2}$$

Send

$$\frac{\vdash_\iota \varepsilon : V}{E \vdash x := \varepsilon @ \iota : (x : \iota \cdot V)}$$

Seq

$$\frac{E \vdash \pi_1 : \Delta_1 \qquad E \vdash \pi_2 : \Delta_2}{E \vdash \pi_1; \pi_2 : \Delta_1; \Delta_2}$$

MAC

$$\frac{E \models \lfloor \texttt{assert}(\psi_{BDOZ}(w))\texttt{@}\iota \rfloor}{E \vdash \texttt{assert}(\psi_{BDOZ}(w))\texttt{@}\iota : (\texttt{m[}ws\texttt{]@}\iota : \iota \cdot \varnothing)}$$

$$\psi_{BDOZ}(w) \triangleq \texttt{m[}wm\texttt{]} = \texttt{m[}wk\texttt{]} + (\texttt{m[delta]} * \texttt{m[}ws\texttt{]})$$

$$\varnothing \underset{H,C}{\rightsquigarrow} \mathcal{L}_{H,C} \qquad \frac{\Delta \underset{H,C}{\rightsquigarrow} \mathcal{L} \qquad \iota \in H}{\Delta; x : \iota \cdot V \underset{H,C}{\rightsquigarrow} \mathcal{L}\{x \mapsto \mathrm{High} \wedge (\bigwedge_{x \in V} \mathcal{L}_2(x))\}} \qquad \frac{\Delta \underset{H,C}{\rightsquigarrow} \mathcal{L} \qquad \iota \in C}{\Delta; x : \iota \cdot V \underset{H,C}{\rightsquigarrow} \mathcal{L}\{x \mapsto \mathrm{Low}\}}$$

Definition 2.4. Given pre-processing predicate E_{pre} and protocol π, we say $E \vdash \pi : \Delta$ is *valid* iff it is derivable and $E_{pre} \wedge \lfloor \pi \rfloor \models E$.

Definition 2.5. Given H, C, define $\mathcal{L}_{H,C}$ such that for all $\mathtt{m}[w]@\iota$ we have $\mathcal{L}_{H,C}(\mathtt{m}[w]@\iota) = \mathrm{High}$ if $\iota \in H$ and Low otherwise.

THEOREM 2.6. *Given pre-processing predicate E_{pre} and protocol π with views$(\pi) = V$, if $E \vdash \pi : \Delta$ is valid and for all H, C with $\Delta \underset{H,C}{\rightsquigarrow} \mathcal{L}$ we have $\mathcal{L}(x) = \mathrm{High}$ for all $x \in V_{H \triangleright C}$, then cheating is detectable in π.*

3 COMPOSITIONAL TYPE VERIFICATION IN *Prelude*

3.1 Syntax and Semantics

$$\begin{array}{rcl}
x & ::= & \mathtt{r}[e]@e \mid \mathtt{s}[e]@e \mid \mathtt{m}[e]@e \mid \mathtt{p}[e] \mid \mathtt{out}@e \\
\multicolumn{3}{l}{\ell \in \mathrm{Field},\ y \in \mathrm{EVar},\ f \in \mathrm{FName}} \\
e & ::= & v \mid \mathtt{r}[e] \mid \mathtt{s}[e] \mid \mathtt{m}[e] \mid \mathtt{p}[e] \mid e\ \mathit{binop}\ e \mid \mathtt{let}\ y = e\ \mathtt{in}\ e \mid \\
& & f(e, \ldots, e) \mid \{\ell = e; \ldots; \ell = e\} \mid e.\ell \\
\mathbf{c} & ::= & \mathtt{m}[e]@e := e@e \mid \mathtt{p}[e] := e@e \mid \mathtt{out}@e := e@e \mid \mathtt{assert}(e = e)@e \mid \\
& & f(e, \ldots, e) \mid \mathbf{c}; \mathbf{c} \mid \mathtt{m}[e]@e\ \mathtt{as}\ \phi \\
\mathit{binop} & ::= & + \mid - \mid * \mid {+}{+} \\
v & ::= & w \mid \iota \mid \varepsilon \mid \{\ell = v; \ldots; \ell = v\} \\
\mathit{fn} & ::= & f(y, \ldots, y)\{e\} \mid f(y, \ldots, y)\{\mathbf{c}\}
\end{array}$$

$$\frac{e[v/y] \Rightarrow v'}{\mathtt{let}\ y = v\ \mathtt{in}\ e \Rightarrow v'}$$

$$\frac{C(f) = y_1, \ldots, y_n,\ e \qquad e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n \qquad e[v_1/y_1] \cdots [v_n/y_n] \Rightarrow v}{f(e_1, \ldots, e_n) \Rightarrow v}$$

$$\frac{e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n}{\{\ell_1 = e_1; \ldots; \ell_n = e_n\} \Rightarrow \{\ell_1 = v_1; \ldots; \ell_n = v_n\}} \qquad \frac{e \Rightarrow \{\ldots; \ell = v; \ldots\}}{e.\ell \Rightarrow v}$$

$$\frac{e_1 \Rightarrow w_1 \qquad e_2 \Rightarrow w_2}{e_1{+}{+}e_2 \Rightarrow w_1 w_2} \qquad \frac{e \Rightarrow w}{\mathtt{m}[e] \Rightarrow \mathtt{m}[w]} \qquad \frac{e_1 \Rightarrow \varepsilon_1 \qquad e_2 \Rightarrow \varepsilon_2}{e_1 + e_2 \Rightarrow \varepsilon_1 + \varepsilon_2} \qquad \frac{e_1 \Rightarrow w \qquad e_2 \Rightarrow \iota}{\mathtt{m}[e_1]@e_2 \Rightarrow \mathtt{m}[w]@\iota}$$

$$\frac{e_1 \Rightarrow x \qquad e_2 \Rightarrow \varepsilon \qquad e_3 \Rightarrow \iota}{e_1 := e_2@e_3 \Rightarrow x := \varepsilon@\iota} \qquad\qquad \frac{e_1 \Rightarrow \pi_1 \qquad e_2 \Rightarrow \pi_2}{e_1; e_2 \Rightarrow \pi_1; \pi_2}$$

$$\frac{e_1 \Rightarrow \varepsilon_1 \qquad e_2 \Rightarrow \varepsilon_2 \qquad e_3 \Rightarrow \iota}{\texttt{assert}(e_1 = e_2)@e_3 \Rightarrow \texttt{assert}(\varepsilon_1 = \varepsilon_2)@\iota}$$

$$\frac{C(f) = y_1, \ldots, y_n, \mathbf{c} \qquad e_1 \Rightarrow v_1 \ \cdots \ e_n \Rightarrow v_n \qquad \rho = [v_1/y_1] \cdots [v_n/y_n] \qquad \rho(\mathbf{c}) \Rightarrow \pi}{f(e_1, \ldots, e_n) \Rightarrow \pi}$$

3.2 Dependent Hoare Type Theory

$$\frac{e_1 \Rightarrow x_1 \qquad e_2 \Rightarrow x_2}{e_1 * e_2 \Rightarrow x_1 * x_2} \qquad \frac{e_1 \Rightarrow \phi_1 \qquad e_2 \Rightarrow \phi_2}{e_1 \equiv e_2 \Rightarrow \phi_1 \equiv \phi_2} \qquad \frac{e_1 \Rightarrow E_1 \qquad e_2 \Rightarrow E_2}{e_1 \land e_2 \Rightarrow E_1 \land E_2}$$

$$\frac{e_1 \Rightarrow x \qquad \check{T} \Rightarrow T}{c(e_1, \check{T}) \Rightarrow c(x, T)} \qquad \frac{\check{t}_1 \Rightarrow t_1 \qquad \cdots \qquad \check{t}_n \Rightarrow t_n}{\{\check{t}_1, \ldots, \check{t}_n\} \Rightarrow \{t_1, \ldots, t_n\}} \qquad \frac{\check{\Gamma} \Rightarrow \Gamma \qquad e \Rightarrow x \qquad \check{T} \Rightarrow T}{\check{\Gamma}; e : \check{T} \Rightarrow \Gamma; x : T}$$

$$\frac{\check{\Delta} \Rightarrow \Delta \qquad e_1 \Rightarrow x \qquad e_2 \Rightarrow \iota \qquad \check{V} \Rightarrow V}{\check{\Delta}; e_1 : e_2 \cdot \check{V} \Rightarrow \Delta; x : \iota \cdot V}$$

$$\frac{\check{E}_1 \Rightarrow E_1 \qquad \check{\Gamma} \Rightarrow \qquad \check{R} \Rightarrow R \qquad \check{\Delta} \Rightarrow \Delta \qquad \check{E}_2 \Rightarrow E_2}{\{\check{E}_1\}\ \check{\Gamma}, \check{R} \cdot \check{\Delta}\ \{\check{E}_2\} \Rightarrow \{E_1\}\ \Gamma, R \cdot \Delta\ \{E_2\}}$$

$$\Vdash x : (\varnothing, \{x\}) \qquad\qquad \frac{\Vdash \phi : (R, T) \qquad \texttt{r[}w\texttt{]@}\iota \notin R \qquad \oplus \in \{+, -\}}{\Vdash \phi \oplus \texttt{r[}w\texttt{]@}\iota : (R \cup \{\texttt{r[}w\texttt{]@}\iota\}, \{c(\texttt{r[}w\texttt{]@}\iota, T)\})}$$

$$\frac{\Vdash \phi_1 : (R_1, T_1) \qquad \Vdash \phi_2 : (R_2, T_2) \qquad \oplus \in \{+, -, *\}}{\Vdash \phi_1 \oplus \phi_2 : (R_1; R_2, T_1 \cup T_2)}$$

$$\textsc{Mesg}\ \frac{e_1 \Rightarrow x \qquad e_2 \Rightarrow \varepsilon \qquad e_3 \Rightarrow \iota \qquad \Vdash \lfloor \varepsilon@\iota \rfloor : (R_2, T) \qquad \vdash_\iota \varepsilon : V}{\vdash e_1 := e_2@e_3 : \{E\}\ (x : T), R_1; R_2 \cdot (x : \iota \cdot V)\ \{E \land x \equiv \lfloor \varepsilon@\iota \rfloor\}}$$

$$\textsc{Encode}\ \frac{e_1 \Rightarrow w \qquad e_2 \Rightarrow \iota \qquad e_3 \Rightarrow \phi \qquad E \models \lfloor \varepsilon@\iota \rfloor \equiv \phi \qquad \Vdash \phi : (R, T)}{\vdash \texttt{m[}e_1\texttt{]@}e_2 \texttt{ as } e_3 : \{E\}\ (\texttt{m[}w\texttt{]@}\iota : T), R \cdot \varnothing\ \{E\}}$$

$$\textsc{App}\ \frac{\begin{array}{c} \mathrm{sig}(f) = \Pi y_1, \ldots, y_n.\{\check{E}_1\}\ \check{\Gamma}, \check{\Delta} \cdot \check{R}\ \{\check{E}_2\} \qquad e_1 \Rightarrow v_1 \ \cdots \ e_n \Rightarrow v_n \\ \rho = [v_1/y_1] \cdots [v_n/y_n] \qquad \rho(\{\check{E}_1\}\ \check{\Gamma}, \check{R} \cdot \check{\Delta}\ \{\check{E}_2\}) \Rightarrow \{E_1\}\ \Gamma, R \cdot \Delta\ \{E_2\} \qquad E \models E_1 \end{array}}{\vdash f(e_1, \ldots, e_n) : \{E\}\ \Gamma, R \cdot \Delta\ \{E \land E_2\}}$$

$$\textsc{Seq}\ \frac{\vdash \pi_1 : \{E_1\}\ \Gamma_1, R_1 \cdot \Delta_1\ \{E_2\} \qquad \vdash \pi_2 : \{E_2\}\ \Gamma_2, R_2 \cdot \Delta_2\ \{E_3\}}{\vdash \pi_1; \pi_2 : \{E_1\}\ \Gamma_1; \Gamma_2, R_1; R_2 \cdot \Delta_1; \Delta_2\ \{E_3\}}$$

Sig

$$\frac{C(f) = y_1, \ldots, y_n, \mathbf{c} \qquad \rho = [v_1/y_1] \cdots [v_n/y_n] \quad \rho(\{\breve{E}_1\}\ \breve{\Gamma}, \breve{\Delta} \cdot \breve{R}\ \{\breve{E}_2\}) \Rightarrow \{E_1\}\ \Gamma, R \cdot \Delta\ \{E_2\} \qquad \vdash \rho(\mathbf{c}) : \{E_1\}\ \Gamma, R \cdot \Delta\ \{E\} \qquad E \models E_2}{f : \Pi y_1, \ldots, y_n. \{\breve{E}_1\}\ \breve{\Gamma}, \breve{R} \cdot \breve{\Delta}\ \{\breve{E}_2\}}$$

Definition 3.1. sig is *verified* iff $f : \mathrm{sig}(f)$ is valid for all $f \in \mathrm{dom}(\mathrm{sig})$.

Theorem 3.2. *Given preprocessing predicate E_{pre}, program* $\mathbf{c}$*, and verified* sig*, if the judgement* $\vdash \mathbf{c} : \{E_{pre}\}\ \Gamma, R \cdot \Delta\ \{E\}$ *is derivable then* $\mathbf{c} \Rightarrow \pi$ *and:*

(1) $R, E \vdash E_{pre} \wedge \lfloor \pi \rfloor : \Gamma$ is valid.
(2) $E \vdash \pi : \Delta$ is valid.

3.3 Confidentiality Examples

```
andtableygc(g,x,y)
{
   let table = (~r[g],~r[g],~r[g],r[g])
   in permute4(r[x],r[y],table)
}

m[x]@1 := s2(s[x],r[x],~r[x])@2;
m[x]@1 as s[x]@2 xor r[x]@2;

// m[x]@1 : { c(r[x]@2, { s[x]@2 }) }

m[y]@1 := OT(s[y]@1,r[y],~r[y])@2;
m[y]@1 as s[y]@1 xor r[y]@2;

// m[y]@1 : { c(r[y]@2, { s[y]@1 }) }

m[ag]@1 := OT4(m[x]@1, m[y]@1, andtable(ag,r[x],r[y]))@2;
m[ag]@1 as  ~((r[x]@2 = m[x]@1) and (r[y]@2 = m[y]@1)) xor r[ag]@2;

// m[ag]@1 : { c(r[ag]@2, {r[x]@2, r[y]@2, m[x]@1,  m[y]@1} }

p[o] := OT2(m[ag]@1, perm2(r[ag],(false,true)))@2

// p[o] : { c(r[ag]@2, {r[x]@2, r[y]@2,  m[x]@1,  m[y]@1}), r[ag]@2  }

out@1 := p[o]@1

// out@1 == s[x] and s[y]
    encodegmw(in, i1, i2) {
      m[in]@i2 := (s[in] xor r[in])@i1;
      m[in]@i1 := r[in]@i1
    }

    andtablegmw(x, y, z) {
```

```
    let r11 = r[z] xor (m[x] xor true) and (m[y] xor true) in
    let r10 = r[z] xor (m[x] xor true) and (m[y] xor false) in
    let r01 = r[z] xor (m[x] xor false) and (m[y] xor true) in
    let r00 = r[z] xor (m[x] xor false) and (m[y] xor false) in
    { row1 = r11; row2 = r10; row3 = r01; row4 = r00 }
  }

  andgmw(z, x, y) {
    let table = andtablegmw(x,y,z) in
    m[z]@2 := OT4(m[x],m[y],table,2,1);
    m[z]@2 as ~((m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2)) xor r[z]@1);
    m[z]@1 := r[z]@1
  }

  // and gate correctness postcondition
 {} andgmw { m[z]@1 xor m[z]@2 == (m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2) }

  // and gate type
  andgmw :
   Pi z,x,y .
   {}
   { { r[z]@1 },
   (m[z]@1 : { r[z]@1 }; m[z]@2 : {c(r[z]@1, { m[x]@1, m[x]@2, m[y]@1, m[y]@2 })} ),
     m[z]@1 xor m[z]@2 == (m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2)}

  xorgmw(z, x, y) {
    m[z]@1 := (m[x] xor m[y])@1; m[z]@2 := (m[x] xor m[y])@2;
  }

  decodegmw(z) {
    p["1"] := m[z]@1; p["2"] := m[z]@2;
    out@1 := (p["1"] xor p["2"])@1;
    out@2 := (p["1"] xor p["2"])@2
  }

  prot() {
    encodegmw("x",2,1);
    encodegmw("y",2,1);
    encodegmw("z",1,2);
    andgmw("g1","x","z");
    xorgmw("g2","g1","y");
    decodegmw("g2")
  }

  {} prot { out@1 == (s["x"]@1 and s["z"]@2) xor s["y"]@1 }
```

3.4 Integrity Examples

```
secopen(w1,w2,w3,i1,i2) {
```

```
      pre(m[w1++"m"]@i2 == m[w1++"k"]@i1 + (m["delta"]@i1 * m[w1++"s"]@i2 /\
          m[w1++"m"]@i2 == m[w1++"k"]@i1 + (m["delta"]@i1 * m[w1++"s"]@i2));
      let locsum =  macsum(macshare(w1), macshare(w2)) in
      m[w3++"s"]@i1 := (locsum.share)@i2;
      m[w3++"m"]@i1 := (locsum.mac)@i2;
      auth(m[w3++"s"],m[w3++"m"],mack(w1) + mack(w2),i1);
      m[w3]@i1 := (m[w3++"s"] + (locsum.share))@i1
  }


  _open(x,i1,i2){
    m[x++"exts"]@i1 := m[x++"s"]@i2;
    m[x++"extm"]@i1 := m[x++"m"]@i2;
    assert(m[x++"extm"] == m[x++"k"] + (m["delta"] * m[x++"exts"]));
    m[x]@i1 := (m[x++"exts"] + m[x++"s"])@i2
  }`

  _sum(z, x, y,i1,i2) {
      pre(m[x++"m"]@i2 == m[x++"k"]@i1 + (m["delta"]@i1 * m[x++"s"]@i2 /\
          m[y++"m"]@i2 == m[y++"k"]@i1 + (m["delta"]@i1 * m[y++"s"]@i2));
      m[z++"s"]@i2 := (m[x++"s"] + m[y++"s"])@i2;
      m[z++"m"]@i2 := (m[x++"m"] + m[y++"m"])@i2;
      m[z++"k"]@i1 := (m[x++"k"] + m[y++"k"])@i1;
      post(m[z++"m"]@i2 == m[z++"k"]@i1 + (m["delta"]@i1 * m[z++"s"]@i2)
  }

  sum(z,x,y) { _sum(z,x,y,1,2);_sum(z,x,y,2,1) }

  open(x) { _open(x,1,2); _open(x,2,1) }


  sum("a","x","d");
  open("d");
  sum("b","y","e");
  open("e");
  let xys =
      macsum(macctimes(macshare("b"), m["d"]),
             macsum(macctimes(macshare("a"), m["e"]),
                    macshare("c")))
  let xyk = mack("b") * m["d"] + mack("a") * m["e"] + mack("c")

  secopen("a","x","d",1,2);
    secopen("a","x","d",2,1);
    secopen("b","y","e",1,2);
    secopen("b","y","e",2,1);
    let xys =
      macsum(macctimes(macshare("b"), m["d"]),
             macsum(macctimes(macshare("a"), m["e"]),
```

```
                macshare("c")))
in
let xyk = mack("b") * m["d"] + mack("d") * m["d"] + mack("c")
in
secreveal(xys,xyk,"1",1,2);
secreveal(maccsum(xys,m["d"] * m["e"]),
          xyk - m["d"] * m["e"],
          "2",2,1);
out@1 := (p[1] + p[2])@1;
out@2 := (p[1] + p[2])@2;
```