1 *Overture* SYNTAX AND SEMANTICS

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

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

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

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

2 *Overture* ADVERSARIAL SEMANTICS

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

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

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

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

3 *Overture* CONSTRAINT TYPING

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

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.

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

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

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

$$\lfloor x := \varepsilon\texttt{@}\iota \rfloor = x \equiv \lfloor \varepsilon\texttt{@}\iota \rfloor \qquad \lfloor \pi_1; \pi_2 \rfloor = \lfloor \pi_1 \rfloor \wedge \lfloor \pi_2 \rfloor$$

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.

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

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

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

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

3.1 Confidentiality Types

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

$$\textsc{Send}\quad \frac{R, E \vdash \lfloor \varepsilon@\iota \rfloor : T}{R, E \vdash x := \varepsilon@\iota : (x : T)} \qquad\qquad \textsc{Seq}\quad \frac{R_1, E \vdash \pi_1 : \Gamma_1 \qquad R_2, E \vdash \pi_2 : \Gamma_2}{R_1; R_2, E \vdash \pi_1; \pi_2 : \Gamma_1; \Gamma_2}$$

Definition 3.1. $R, E \vdash \pi : \Gamma$ is *valid* iff it is derivable and $\lfloor \pi \rfloor \models E$.

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

$$\frac{\Gamma, C \vdash \{\mathtt{r}[w]@\iota\} \qquad \Gamma, C \vdash \{c(\mathtt{r}[w]@\iota, T)\}}{\Gamma, C \vdash T}$$

THEOREM 3.2. *If $R, E \vdash \pi : \Gamma$ is valid and for all H, C it is not the case that $\Gamma, C \vdash \{\mathtt{s}[w]@\iota\}$ for $\iota \in H$, then π satisfies gradual release.*

3.1.1 *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 }) }
```

3.2 Compositional Type Verification in *Prelude*

(*Need to fix the following to allow reduction of x. – Chris*)

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

$$\textsc{Encode}\quad \frac{e_1 \Rightarrow \varepsilon \qquad e_2 \Rightarrow \iota \qquad e_3 \Rightarrow \phi \qquad E \models \lfloor \varepsilon @ \iota \rfloor \equiv \phi \qquad R_1, E \Vdash \phi : (R_2, T)}{R_1, E \vdash x := e_1 @ e_2 \ \texttt{as}\ e_3 : (x : T, R_1; R_2, E \wedge x \equiv \phi)}$$

$$\textsc{App}\quad \frac{\begin{array}{c}\mathrm{sig}(f) = \{E_1\}\ x_1, \ldots, x_n\ \{\Gamma, R, E_2\} \\ e_1 \Rightarrow v_1 \ \cdots \ e_n \Rightarrow v_n \qquad \rho = [v_1/x_1] \cdots [v_n/x_n] \qquad E \models \rho(E_1)\end{array}}{R_1, E \vdash f(e_1, \ldots, e_n) : (\rho(\Gamma), R_1; \rho(R), E \wedge \rho(E_2))}$$

$$\textsc{Seq}\quad \frac{R_1, E_1 \vdash \pi_1 : (\Gamma_2, R_2, E_2) \qquad R_2, E_2 \vdash \pi_2 : (\Gamma_3, R_3, E_3)}{R_1, E_1 \vdash \pi_1; \pi_2 : (\Gamma_2; \Gamma_3, R_3, E_3)}$$

$$\textsc{Sig}\quad \frac{C(f) = x_1, \ldots, x_n, \mathbf{c} \qquad \rho = [v_1/x_1] \cdots [v_n/x_n] \qquad \varnothing, \rho(E_1) \vdash \mathbf{c} : (\rho(\Gamma), \rho(R), \rho(E_2))}{f : \{E_1\}\ x_1, \ldots, x_n\ \{\Gamma, R, E_2\}}$$

Definition 3.3. sig is *verified* iff $f : \mathrm{sig}(f)$ is valid for all $f \in \mathrm{dom}(\mathrm{sig})$.

The following theorem holds for protocols with default preprocessing.

THEOREM 3.4. *If* sig *is verified and* $\varnothing, \varnothing \vdash e : (\Gamma, R, E)$ *then* $e \Rightarrow \pi$ *and* $R, E \vdash \pi : \Gamma$ *is valid.*

3.3 Integrity Types

$$\frac{\textsc{Value}}{\Gamma, \varnothing, E \vdash_\iota v : \varnothing \cdot \text{High}} \qquad \frac{\textsc{Secret}}{\Gamma, \varnothing, E \vdash_\iota \mathtt{s}[w] : \{\mathtt{s}[w]@\iota\} \cdot \mathcal{L}(\iota)} \qquad \frac{\textsc{Rando}}{\Gamma, \varnothing, E \vdash_\iota \mathtt{r}[w] : \{\mathtt{r}[w]@\iota\} \cdot \mathcal{L}(\iota)}$$

$$\frac{\textsc{Mesg}}{\Gamma, \varnothing, E \vdash_\iota \mathtt{m}[w] : \Gamma(\mathtt{m}[w]@\iota)} \qquad \frac{\textsc{PubM}}{\Gamma, \varnothing, E \vdash_\iota \mathtt{p}[w] : \Gamma(\mathtt{p}[w])} \qquad \textsc{IntegrityWeaken}\ \frac{\Gamma, R, E \vdash_\iota \varepsilon : T \cdot \varsigma_1 \qquad \varsigma_1 \preceq \varsigma_2}{\Gamma, R, E \vdash_\iota \varepsilon : T \cdot \varsigma_2}$$

$$\textsc{Encode}\ \frac{\Gamma, \varnothing, E \vdash_\iota \varepsilon : T \cdot \varsigma \qquad E \models \lfloor \varepsilon@\iota \rfloor = \phi \oplus \mathtt{r}[w]@\iota' \qquad \oplus \in \{+, -\}}{\Gamma, \mathtt{r}[w]@\iota, E \vdash_\iota \varepsilon : \{c(\mathtt{r}[w]@\iota', \Gamma(\phi))\} \cdot \varsigma}$$

$$\textsc{Binop}\ \frac{\Gamma, R_1, E \vdash_\iota \varepsilon_1 : T_1 \cdot \varsigma \qquad \Gamma, R_2, E \vdash_\iota \varepsilon_2 : T_2 \cdot \varsigma \qquad \oplus \in \{+, -, *\}}{\Gamma, R_1; R_2, E \vdash_\iota \varepsilon_1 \oplus \varepsilon_2 : T_1 \cup T_2 \cdot \varsigma}$$

$$\textsc{Send}\ \frac{\Gamma, R, E \vdash_\iota \varepsilon : T \cdot \mathcal{L}(\iota) \qquad E' \models E \wedge x = \lfloor \varepsilon@\iota \rfloor}{\Gamma, R, E \vdash x := \varepsilon@\iota : \Gamma; x : T \cdot \mathcal{L}(\iota), E'} \qquad \textsc{Assert}\ \frac{E \models \lfloor \varepsilon_1@\iota \rfloor = \lfloor \varepsilon_2@\iota \rfloor}{\Gamma, R, E \vdash \mathtt{assert}(\varepsilon_1 = \varepsilon_2)@\iota : \Gamma, E}$$

$$\textsc{Seq}\ \frac{\Gamma_1, R_1, E_1 \vdash \pi_1 : \Gamma_2, E_2 \qquad \Gamma_2, R_2, E_2 \vdash \pi_2 : \Gamma_3, E_3}{\Gamma_1, R_1; R_2, E_1 \vdash \pi_1; \pi_2 : \Gamma_3, E_3} \qquad \textsc{Constraint}\ \frac{\Gamma_1, R, E_1 \vdash \pi : \Gamma_2, E_2 \qquad E_1' \models E_1' \qquad E_2 \models E_2'}{\Gamma_1, R, E_1' \vdash \pi : \Gamma_2, E_2'}$$

$$\text{MAC}\ \frac{E \models \mathtt{m}[wm]@\iota = \mathtt{m}[wk]@\iota + (\mathtt{m}[delta]@\iota * \mathtt{m}[ws]@\iota) \qquad \Gamma(\mathtt{m}[ws]@\iota) = T \cdot \varsigma}{\Gamma, R, E \vdash \mathtt{assert}(\mathtt{m}[wm] = \mathtt{m}[wk] + (\mathtt{m}[delta] * \mathtt{m}[ws]))@\iota : \Gamma; \mathtt{m}[ws]@\iota : T \cdot \text{High}, E}$$

4 *Prelude* SYNTAX AND SEMANTICS

$$\ell \in \text{Field}, \ y \in \text{EVar}, \ f \in \text{FName}$$

$$\begin{array}{rcl}
e & ::= & v \mid \mathtt{r}[e] \mid \mathtt{s}[e] \mid \mathtt{m}[e] \mid \mathtt{p}[e] \mid e \ \mathit{binop}\ e \mid \mathtt{let}\ y = e\ \mathtt{in}\ e \mid \\
 & & f(e, \ldots, e) \mid \{\ell = e; \ldots; \ell = e\} \mid e.\ell \\
\mathbf{c} & ::= & \mathtt{m}[e]@e := e@e \mid \mathtt{p}[e] := e@e \mid \mathtt{out}@e := e@e \mid \mathtt{assert}(e = e)@e \mid \\
 & & f(e, \ldots, e) \mid \mathbf{c}; \mathbf{c} \mid \mathtt{pre}(E) \mid \mathtt{post}(E) \\
\mathit{binop} & ::= & + \mid - \mid * \mid {+}{+} \\
v & ::= & w \mid \iota \mid \varepsilon \mid \{\ell = v; \ldots; \ell = v\} \\
\mathit{fn} & ::= & f(y, \ldots, y)\{e\} \mid f(y, \ldots, y)\{\mathbf{c}\} \\
\phi & ::= & \mathtt{r}[e]@e \mid \mathtt{s}[e]@e \mid \mathtt{m}[e]@e \mid \mathtt{p}[e] \mid \mathtt{out}@e \mid \phi + \phi \mid \phi - \phi \mid \phi * \phi \\
E & ::= & \phi \equiv \phi \mid E \wedge E
\end{array}$$

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

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

$$\frac{e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n}{\{\ell_1 = e_1; \ldots; \ell_n = e_n\} \Rightarrow \{\ell_1 = v_1; \ldots; \ell_n = v_n\}} \qquad \frac{e \Rightarrow \{\ldots; \ell = v; \ldots\}}{e.\ell \Rightarrow v} \qquad \frac{e_1 \Rightarrow w_1 \qquad e_2 \Rightarrow w_2}{e_1 {+\!+} e_2 \Rightarrow w_1 w_2}$$

$$\frac{e_1 \Rightarrow \varepsilon_1 \qquad e_2 \Rightarrow \varepsilon_2 \qquad e \Rightarrow \iota}{(\pi, (E_1, E_2), \mathsf{on}, \texttt{assert}(e_1 = e_2)\texttt{@}e) \Rightarrow (\pi; \texttt{assert}(\varepsilon_1 = \varepsilon_2)\texttt{@}\iota, (E_1, E_2 \wedge \lfloor \varepsilon_1 \texttt{@}\iota \rfloor = \lfloor \varepsilon_2 \texttt{@}\iota \rfloor), \mathsf{on})}$$

$$\frac{e_1 \Rightarrow \varepsilon_1 \qquad e_2 \Rightarrow \varepsilon_2 \qquad e \Rightarrow \iota}{(\pi, (E_1, E_2), \texttt{off}, \texttt{assert}(e_1 = e_2)\texttt{@}e) \Rightarrow (\pi; \texttt{assert}(\varepsilon_1 = \varepsilon_2)\texttt{@}\iota, (E_1, E_2, \texttt{off})}$$

$$\frac{e_1 \Rightarrow w \qquad e_2 \Rightarrow \iota_1 \qquad e_3 \Rightarrow \varepsilon \qquad e_4 \Rightarrow \iota_2}{(\pi, (E_1, E_2), \mathsf{on}, \texttt{m}[e_1]\texttt{@}e_2 := e_3\texttt{@}e_4) \Rightarrow (\pi; \texttt{m}[w]\texttt{@}\iota_1 := \varepsilon\texttt{@}\iota_2, (E_1 \wedge \texttt{m}[w]\texttt{@}\iota_1 = \lfloor \varepsilon\texttt{@}\iota_2 \rfloor, E_2), \mathsf{on})}$$

$$\frac{e_1 \Rightarrow w \qquad e_2 \Rightarrow \iota_1 \qquad e_3 \Rightarrow \varepsilon \qquad e_4 \Rightarrow \iota_2}{(\pi, (E_1, E_2), \texttt{off}, \texttt{m}[e_1]\texttt{@}e_2 := e_3\texttt{@}e_4) \Rightarrow (\pi; \texttt{m}[w]\texttt{@}\iota_1 := \varepsilon\texttt{@}\iota_2, (E_1, E_1), \texttt{off})}$$

$$(\pi, (E_1, E_2), \mathsf{on}, \texttt{pre}(E)) \Rightarrow (\pi, E_1, E_2 \wedge E, \texttt{off})$$

$$(\pi, (E_1, E_2), \texttt{off}, \texttt{post}(E)) \Rightarrow (\pi, (E_1 \wedge E, E_2), \mathsf{on})$$

$$\frac{(\pi_1, (E_{11}, E_{12}), sw_1, \mathbf{c}_1) \Rightarrow (\pi_2, (E_{21}, E_{22}), sw_2) \qquad (\pi_2, (E_{21}, E_{22}), sw_2, \mathbf{c}_2) \Rightarrow (\pi_3, (E_{31}, E_{32}), sw_3)}{(\pi_1, (E_{11}, E_{12}), sw_1, \mathbf{c}_1; \mathbf{c}_2) \Rightarrow (\pi_3, (E_{31}, E_{32}), sw_3)}$$

$$\frac{\begin{array}{c} C(f) = y_1, \ldots, y_n,\ \mathbf{c} \\ e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n \qquad (\pi_1, (E_{11}, E_{12}), sw_1, \mathbf{c}[v_1/y_1,] \cdots [v_n/y_n]) \Rightarrow (\pi_2, (E_{21}, E_{22}), sw_2) \end{array}}{(\pi_1, (E_{11}, E_{12}), sw_1, f(e_1, \ldots, e_n)) \Rightarrow (\pi_2, (E_{21}, E_{22}), sw_2)}$$

5 EXAMPLES

```
encodegmw(in, i1, i2) {
  m[in]@i2 := (s[in] xor r[in])@i2;
  m[in]@i1 := r[in]@i2
}

andtablegmw(b1, b2, r) {
  let r11 = r xor (b1 xor true) and (b2 xor true) in
  let r10 = r xor (b1 xor true) and (b2 xor false) in
  let r01 = r xor (b1 xor false) and (b2 xor true) in
  let r00 = r xor (bl xor false) and (b2 xor false) in
  { row1 = r11; row2 = r10; row3 = r01; row4 = r00 }
```

```
    }

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

    encodegmw("x",2,1);
    encodegmw("y",2,1);
    encodegmw("z",1,2);
    andgmw("g1","x","z");
    xorgmw("g2","g1","y");
    decodegmw("g2")
    pre();
    post(out@1 == (s["x"]@1 and s["z"]@2) xor s["y"]@1)


  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;
```