

1 Overture SYNTAX AND SEMANTICS

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

$\varepsilon ::= r[w] \mid s[w] \mid m[w] \mid p[w] \mid$ *expressions*
 $v \mid \varepsilon - \varepsilon \mid \varepsilon + \varepsilon \mid \varepsilon * \varepsilon$

$x ::= r[w]@_\iota \mid s[w]@_\iota \mid m[w]@_\iota \mid p[w] \mid \text{out}@_\iota$ *variables*

$\pi ::= m[w]@_\iota := \varepsilon@_\iota \mid p[w] := e@_\iota \mid \text{out}@_\iota := \varepsilon@_\iota \mid \pi; \pi$ *protocols*

$$\begin{aligned} \llbracket \sigma, v \rrbracket_\iota &= v \\ \llbracket \sigma, \varepsilon_1 + \varepsilon_2 \rrbracket_\iota &= \llbracket \llbracket \sigma, \varepsilon_1 \rrbracket_\iota + \llbracket \sigma, \varepsilon_2 \rrbracket_\iota \rrbracket \\ \llbracket \sigma, \varepsilon_1 - \varepsilon_2 \rrbracket_\iota &= \llbracket \llbracket \sigma, \varepsilon_1 \rrbracket_\iota - \llbracket \sigma, \varepsilon_2 \rrbracket_\iota \rrbracket \\ \llbracket \sigma, \varepsilon_1 * \varepsilon_2 \rrbracket_\iota &= \llbracket \llbracket \sigma, \varepsilon_1 \rrbracket_\iota * \llbracket \sigma, \varepsilon_2 \rrbracket_\iota \rrbracket \\ \llbracket \sigma, r[w] \rrbracket_\iota &= \sigma(r[w]@_\iota) \\ \llbracket \sigma, s[w] \rrbracket_\iota &= \sigma(s[w]@_\iota) \\ \llbracket \sigma, m[w] \rrbracket_\iota &= \sigma(m[w]@_\iota) \\ \llbracket \sigma, p[w] \rrbracket_\iota &= \sigma(p[w]) \end{aligned}$$

$$\frac{(\sigma, x := \varepsilon@_\iota) \Rightarrow \sigma\{x \mapsto \llbracket \sigma, \varepsilon \rrbracket_\iota\} \quad \begin{array}{c} (\sigma_1, \pi_1) \Rightarrow \sigma_2 \quad (\sigma_2, \pi_2) \Rightarrow \sigma_3 \\ \hline (\sigma_1, \pi_1; \pi_2) \Rightarrow \sigma_3 \end{array}}{}$$

2 Overture CONSTRAINT TYPING

2.1 Constraint Satisfiability Modulo Finite Fields

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

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.

$$\begin{aligned} \lfloor x \rfloor &= x & \lfloor \varepsilon_1 + \varepsilon_2@_\iota \rfloor &= \lfloor \varepsilon_2@_\iota \rfloor + \lfloor \varepsilon_1@_\iota \rfloor & \lfloor \varepsilon_1 - \varepsilon_2@_\iota \rfloor &= \lfloor \varepsilon_2@_\iota \rfloor - \lfloor \varepsilon_1@_\iota \rfloor \\ \lfloor \varepsilon_1 * \varepsilon_2@_\iota \rfloor &= \lfloor \varepsilon_2@_\iota \rfloor * \lfloor \varepsilon_1@_\iota \rfloor \end{aligned}$$

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

$$\lfloor x := \varepsilon@_\iota \rfloor = x \equiv \lfloor \varepsilon@_\iota \rfloor \quad \lfloor \pi_1; \pi_2 \rfloor = \lfloor \pi_1 \rfloor \wedge \lfloor \pi_2 \rfloor$$

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.

```

m[s1]@2 := (s[1] - r[local] - r[x])@1
m[s1]@3 := r[x]@1
m[s2]@1 := (s[2] - r[local] - r[x])@2
m[s2]@3 := r[x]@2
m[s3]@1 := (s[3] - r[local] - r[x])@3
m[s3]@2 := r[x]@3
p[1] := (r[local] + m[s2] + m[s3])@1
p[2] := (m[s1] + r[local] + m[s3])@2
p[3] := (m[s1] + m[s2] + r[local])@3
out@1 := (p[1] + p[2] + p[3])@1
out@2 := (p[1] + p[2] + p[3])@2
out@3 := (p[1] + p[2] + p[3])@3

```

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

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

2.2 Confidentiality Types

```

t ::= x | c(x, T)
T ∈ 2t
Γ ::= ∅ | Γ; x : T

```

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

$\frac{\text{DEFTY}}{\emptyset, E \vdash \phi : \text{vars}(\phi)}$	$\frac{\text{ENCODE} \quad E \models \phi \equiv \phi' \oplus r[w]@_t \quad \oplus \in \{+, -\} \quad R, E \vdash \phi' : T}{R; \{r[w]@_t\}, E \vdash \phi : \{c(r[w]@_t, T)\}}$
$\frac{\text{SEND} \quad R, E \vdash [\varepsilon@_t] : T}{R, E \vdash x := \varepsilon@_t : (x : T)}$	$\frac{\text{SEQ} \quad R_1, E \vdash \pi_1 : \Gamma_1 \quad R_2, E \vdash \pi_2 : \Gamma_2}{R_1; R_2, E \vdash \pi_1; \pi_2 : \Gamma_1; \Gamma_2}$

Definition 2.2. $R, E \vdash \pi : \Gamma$ is *valid* iff it is derivable and $[\pi] \models E$.

$\frac{\iota \in C}{\Gamma, C \vdash_{\text{leak}} \Gamma(m[w]@_t)}$	$\frac{\Gamma, C \vdash_{\text{leak}} T_1 \cup T_2}{\Gamma, C \vdash_{\text{leak}} T_1}$	$\frac{\Gamma, C \vdash_{\text{leak}} \{m[w]@_t\}}{\Gamma, C \vdash_{\text{leak}} \Gamma(m[w]@_t)}$
$\frac{\Gamma, C \vdash_{\text{leak}} \{r[w]@_t\} \quad \Gamma, C \vdash_{\text{leak}} \{c(r[w]@_t, T)\}}{\Gamma, C \vdash_{\text{leak}} T}$		

THEOREM 2.3. *If $R, E \vdash \pi : \Gamma$ is valid and there exists no H, C and $s[w]@_t$ for $\iota \in H$ with $\Gamma, C \vdash_{\text{leak}} \{s[w]@_t\}$, 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 }

```

```

50 m[x]@1 := s2(s[x], -r[x], r[x])@2
51
52 // m[x]@1 == s[x]@2 + -r[x]@2
53 // m[x]@1 : { c(r[x]@2, { s[x]@2 }) }
54
55 m[y]@1 := OT(s[y]@1, -r[y], r[y])@2
56
57 // m[y]@1 == s[y]@1 + -r[y]@2
58 // m[y]@1 : { c(r[y]@2, { s[y]@1 }) }
59

```

3 Overture ADVERSARIAL SEMANTICS

$$\begin{aligned}
 (\sigma, x := \varepsilon @ \iota) &\Rightarrow_{\mathcal{A}} \sigma \{x \mapsto \llbracket \sigma, \varepsilon \rrbracket_{\iota}\} & \iota \in H \\
 (\sigma, x := \varepsilon @ \iota) &\Rightarrow_{\mathcal{A}} \sigma \{x \mapsto \llbracket \text{rewrite}_{\mathcal{A}}(\sigma_C, \varepsilon) \rrbracket_{\iota}\} & \iota \in C
 \end{aligned}$$

$$\begin{aligned}
 (\sigma, \text{assert}(\varepsilon_1 = \varepsilon_2) @ \iota) &\Rightarrow_{\mathcal{A}} \sigma & \text{if } \llbracket \sigma, \varepsilon_1 \rrbracket_{\iota} = \llbracket \sigma, \varepsilon_2 \rrbracket_{\iota} \text{ or } \iota \in C \\
 (\sigma, \text{assert}(\phi(\varepsilon)) @ \iota) &\Rightarrow_{\mathcal{A}} \perp & \text{if } \neg \phi(\sigma, \llbracket \sigma, \varepsilon \rrbracket_{\iota})
 \end{aligned}$$

$$\begin{aligned}
 &\frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \sigma_2 \quad (\sigma_2, \pi_2) \Rightarrow_{\mathcal{A}} \sigma_3}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \sigma_3} & \frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \perp}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \perp} \\
 &\frac{(\sigma_1, \pi_1) \Rightarrow_{\mathcal{A}} \sigma_2 \quad (\sigma_2, \pi_2) \Rightarrow_{\mathcal{A}} \perp}{(\sigma_1, \pi_1; \pi_2) \Rightarrow_{\mathcal{A}} \perp}
 \end{aligned}$$

3.1 Compositional Type Verification in Prelude

Note the redefinition of x impacts the definition of T , Γ , ϕ , and E .

```

50 x ::= r[e]@e | s[e]@e | m[e]@e | p[e] | out@e
51
52 ℓ ∈ Field, y ∈ EVar, f ∈ FName
53
54 e ::= v | r[e] | s[e] | m[e] | p[e] | e binop e | let y = e in e |
55       f(e, ..., e) | {ℓ = e; ...; ℓ = e} | e.ℓ
56
57 c ::= m[e]@e := e@e | p[e] := e@e | out@e := e@e | assert(e = e)@e |
58       f(e, ..., e) | c; c | m[e]@e as φ
59
60 binop ::= + | - | * | ++
61
62 v ::= w | ι | ε | {ℓ = v; ...; ℓ = v}
63
64 fn ::= f(y, ..., y){e} | f(y, ..., y){c}

```

$$\begin{aligned}
 R \Vdash x : (\emptyset, \{x\}) &\quad \frac{R \Vdash \phi : (R_1, T) \quad r[w]@_{\iota} \notin R \quad \oplus \in \{+, -\}}{R_1 \Vdash \phi \oplus r[w]@_{\iota} : (R_1 \cup \{r[w]@_{\iota}\}, \{c(r[w]@_{\iota}, T)\})} \\
 &\quad \frac{R \Vdash \phi_1 : (R_1, T_1) \quad R \Vdash \phi_2 : (R_2, T_2) \quad \oplus \in \{+, -, *\}}{R_1 \Vdash \phi_1 \oplus \phi_2 : (R_1; R_2, T_1 \cup T_2)}
 \end{aligned}$$

$$\begin{array}{c}
\frac{e[v/y] \Rightarrow v'}{\text{let } y = v \text{ in } e \Rightarrow v'} \\
\\
\frac{C(f) = y_1, \dots, y_n, e \quad e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n \quad e[v_1/y_1] \cdots [v_n/y_n] \Rightarrow v}{f(e_1, \dots, e_n) \Rightarrow v} \\
\\
\frac{e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n}{\{\ell_1 = e_1; \dots; \ell_n = e_n\} \Rightarrow \{\ell_1 = v_1; \dots; \ell_n = v_n\}} \quad \frac{e \Rightarrow \{\dots; \ell = v; \dots\}}{e.\ell \Rightarrow v} \\
\\
\frac{e_1 \Rightarrow w_1 \quad e_2 \Rightarrow w_2}{e_1 ++ e_2 \Rightarrow w_1 w_2} \quad \frac{e \Rightarrow w}{m[e] \Rightarrow w} \quad \frac{e \Rightarrow w}{m[e]@l \Rightarrow m[w]@l} \quad \frac{e_1 \Rightarrow \varepsilon_1 \quad e_2 \Rightarrow \varepsilon_2}{e_1 + e_2 \Rightarrow \varepsilon_1 + \varepsilon_2} \\
\\
\frac{e_1 \Rightarrow x_1 \quad e_2 \Rightarrow x_2}{e_1 + e_2 \Rightarrow x_1 + x_2} \quad \frac{e_1 \Rightarrow \phi_1 \quad e_2 \Rightarrow \phi_2}{e_1 \equiv e_2 \Rightarrow \phi_1 \equiv \phi_2} \quad \frac{e_1 \Rightarrow E_1 \quad e_2 \Rightarrow E_2}{e_1 \Rightarrow e_2 \Rightarrow E_1 \Rightarrow E_2} \\
\\
\frac{\text{MSG} \quad e_1 \Rightarrow \varepsilon \quad e_2 \Rightarrow \iota \quad R_1 \Vdash \lfloor \varepsilon @ \iota \rfloor : (R_2, T)}{R_1 \vdash x := e_1 @ e_2 : \{E\} \ x : T, R_1; R_2 \ \{E \wedge x \equiv \lfloor \varepsilon @ \iota \rfloor\}} \\
\\
\frac{\text{ENCODE} \quad e_1 \Rightarrow w \quad e_2 \Rightarrow \iota \quad E \models \lfloor \varepsilon @ \iota \rfloor \equiv \phi \quad R_1 \Vdash \phi : (R_2, T)}{R_1 \vdash m[e_1]@e_2 \text{ as } \phi : \{E\} \ m[w]@l : T, R_1; R_2 \ \{E \wedge m[w]@l \equiv \phi\}} \\
\\
\frac{\text{APP} \quad \text{sig}(f) = \Pi x_1, \dots, x_n. \{E_1\} \ \Gamma, R \ \{E_2\} \quad e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n \quad \rho = [v_1/x_1] \cdots [v_n/x_n] \quad E \models \rho(E_1)}{R_1 \vdash f(e_1, \dots, e_n) : \{E\} \ \rho(\Gamma), R_1; \rho(R) \ \{E \wedge \rho(E_2)\}} \\
\\
\frac{\text{SEQ} \quad R_1 \vdash \pi_1 : \{E_1\} \ \Gamma_2, R_2 \ \{E_2\} \quad R_1 \vdash \pi_2 : \{E_2\} \ \Gamma_3, R_3 \ \{E_3\}}{R_1 \vdash \pi_1; \pi_2 : \{E_1\} \ \Gamma_2; \Gamma_3, R_2; R_3 \ \{E_3\}} \\
\\
\frac{\text{SIG} \quad C(f) = x_1, \dots, x_n, \mathbf{c} \quad \rho = [v_1/x_1] \cdots [v_n/x_n] \quad \emptyset \vdash \rho(\mathbf{c}) : \{\rho(E_1)\} \ \rho(\Gamma), \rho(R) \ \{E\} \quad E \models \rho(E_2)}{f : \Pi x_1, \dots, x_n. \{E_1\} \ \Gamma, R \ \{E_2\}}
\end{array}$$

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

The following theorem holds for protocols with default preprocessing.

THEOREM 3.2. If sig is verified and $\emptyset \vdash e : \{\emptyset\} \ \Gamma, R \ \{E\}$ then $e \Rightarrow \pi$ and $R, E \vdash \pi : \Gamma$ is valid.

3.1.1 Examples.

andtableygc(g, x, y)

```
{
  let table = (~r[g], ~r[g], ~r[g], r[g])
```

```

148     in permute4(r[x],r[y],table)
149 }
150
151 m[x]@1 := s2(s[x],r[x],~r[x])@2;
152 m[x]@1 as s[x]@2 xor r[x]@2;
153
154 // m[x]@1 : { c(r[x]@2, { s[x]@2 }) }
155
156 m[y]@1 := OT(s[y]@1,r[y],~r[y])@2;
157 m[y]@1 as s[y]@1 xor r[y]@2;
158
159 // m[y]@1 : { c(r[y]@2, { s[y]@1 }) }
160
161 m[ag]@1 := OT4(m[x]@1, m[y]@1, andtable(ag,r[x],r[y]))@2;
162 m[ag]@1 as ~((r[x]@2 = m[x]@1) and (r[y]@2 = m[y]@1)) xor r[ag]@2;
163
164 // m[ag]@1 : { c(r[ag]@2, {r[x]@2, r[y]@2, m[x]@1, m[y]@1}) }
165
166 p[o] := OT2(m[ag]@1, perm2(r[ag],(false,true)))@2
167
168 // p[o] : { c(r[ag]@2, {r[x]@2, r[y]@2, m[x]@1, m[y]@1}), r[ag]@2 }
169
170 out@1 := p[o]@1
171
172 // out@1 == s[x] and s[y]
173
174 encodegmw(in, i1, i2) {
175     m[in]@i2 := (s[in] xor r[in])@i1;
176     m[in]@i1 := r[in]@i1
177 }
178
179 andtablegmw(x, y, z) {
180     let r11 = r[z] xor (m[x] xor true) and (m[y] xor true) in
181     let r10 = r[z] xor (m[x] xor true) and (m[y] xor false) in
182     let r01 = r[z] xor (m[x] xor false) and (m[y] xor true) in
183     let r00 = r[z] xor (m[x] xor false) and (m[y] xor false) in
184     { row1 = r11; row2 = r10; row3 = r01; row4 = r00 }
185 }
186
187 andgmw(z, x, y) {
188     let table = andtablegmw(x,y,z) in
189     m[z]@2 := OT4(m[x],m[y],table,2,1);
190     m[z]@2 as ~((m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2)) xor r[z]@1;
191     m[z]@1 := r[z]@1
192 }
193
194 // and gate correctness postcondition
195 {} andgmw { m[z]@1 xor m[z]@2 == (m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2) }
196

```

```

197
198 // and gate type
199 andgmw :
200   Pi z,x,y .
201   {}
202   { { r[z]@1 },
203     (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 })} ) ,
204     m[z]@1 xor m[z]@2 == (m[x]@1 xor m[x]@2) and (m[y]@1 xor m[y]@2)}
205
206 xorgmw(z, x, y) {
207   m[z]@1 := (m[x] xor m[y])@1; m[z]@2 := (m[x] xor m[y])@2;
208 }
209
210 decodegmw(z) {
211   p["1"] := m[z]@1; p["2"] := m[z]@2;
212   out@1 := (p["1"] xor p["2"] )@1;
213   out@2 := (p["1"] xor p["2"] )@2
214 }
215
216 prot() {
217   encodegmw("x",2,1);
218   encodegmw("y",2,1);
219   encodegmw("z",1,2);
220   andgmw("g1", "x", "z");
221   xorgmw("g2", "g1", "y");
222   decodegmw("g2")
223 }
224
225 {} prot { out@1 == (s["x"]@1 and s["z"]@2) xor s["y"]@1 }
226
227
228
229

```

3.2 Integrity Types

VALUE	SECRET	RANDO
$\Gamma, \emptyset, E \vdash_i v : \emptyset \cdot \text{High}$	$\Gamma, \emptyset, E \vdash_i s[w] : \{s[w]@i\} \cdot \mathcal{L}(i)$	$\Gamma, \emptyset, E \vdash_i r[w] : \{r[w]@i\} \cdot \mathcal{L}(i)$
MESG	PUBM	INTEGRITYWEAKEN
$\Gamma, \emptyset, E \vdash_i m[w] : \Gamma(m[w]@i)$	$\Gamma, \emptyset, E \vdash_i p[w] : \Gamma(p[w])$	$\frac{\Gamma, R, E \vdash_i \varepsilon : T \cdot \varsigma_1 \quad \varsigma_1 \leq \varsigma_2}{\Gamma, R, E \vdash_i \varepsilon : T \cdot \varsigma_2}$
ENCODE		
$\Gamma, \emptyset, E \vdash_i \varepsilon : T \cdot \varsigma \quad E \models \lfloor \varepsilon @i \rfloor = \phi \oplus r[w]@i' \quad \oplus \in \{+, -\}$	$\Gamma, r[w]@i, E \vdash_i \varepsilon : \{c(r[w]@i', \Gamma(\phi))\} \cdot \varsigma$	
BINOP		
$\Gamma, R_1, E \vdash_i \varepsilon_1 : T_1 \cdot \varsigma \quad \Gamma, R_2, E \vdash_i \varepsilon_2 : T_2 \cdot \varsigma \quad \oplus \in \{+, -, *\}$	$\Gamma, R_1; R_2, E \vdash_i \varepsilon_1 \oplus \varepsilon_2 : T_1 \cup T_2 \cdot \varsigma$	

$$\begin{array}{c}
\text{SEND} \\
\frac{\Gamma, R, E \vdash_{\iota} \varepsilon : T \cdot \mathcal{L}(\iota) \quad E' \models E \wedge x = \lfloor \varepsilon @_{\iota} \rfloor}{\Gamma, R, E \vdash x := \varepsilon @_{\iota} : \Gamma; x : T \cdot \mathcal{L}(\iota), E'} \\
\\
\text{SEQ} \\
\frac{\Gamma_1, R_1, E_1 \vdash \pi_1 : \Gamma_2, E_2 \quad \Gamma_2, R_2, E_2 \vdash \pi_2 : \Gamma_3, E_3}{\Gamma_1, R_1; R_2, E_1 \vdash \pi_1; \pi_2 : \Gamma_3, E_3} \\
\\
\text{MAC} \\
\frac{E \models m[\text{wm}]@_{\iota} = m[\text{wk}]@_{\iota} + (m[\text{delta}]@_{\iota} * m[\text{ws}]@_{\iota}) \quad \Gamma(m[\text{ws}]@_{\iota}) = T \cdot \varsigma}{\Gamma, R, E \vdash \text{assert}(m[\text{wm}] = m[\text{wk}] + (m[\text{delta}] * m[\text{ws}]))@_{\iota} : \Gamma; m[\text{ws}]@_{\iota} : T \cdot \text{High}, E} \\
\\
\text{ASSERT} \\
\frac{E \models \lfloor \varepsilon_1 @_{\iota} \rfloor = \lfloor \varepsilon_2 @_{\iota} \rfloor}{\Gamma, R, E \vdash \text{assert}(\varepsilon_1 = \varepsilon_2)@_{\iota} : \Gamma, E} \\
\\
\text{CONSTRAINT} \\
\frac{\Gamma_1, R, E_1 \vdash \pi : \Gamma_2, E_2 \quad E'_1 \models E'_1 \quad E_2 \models E'_2}{\Gamma_1, R, E'_1 \vdash \pi : \Gamma_2, E'_2}
\end{array}$$

4 Prelude SYNTAX AND SEMANTICS

$$\begin{array}{l}
\ell \in \text{Field}, y \in \text{EVar}, f \in \text{FName} \\
e ::= v \mid r[e] \mid s[e] \mid m[e] \mid p[e] \mid e \text{ binop } e \mid \text{let } y = e \text{ in } e \mid \\
\quad f(e, \dots, e) \mid \{\ell = e; \dots; \ell = e\} \mid e.\ell \\
c ::= m[e]@e := e@e \mid p[e] := e@e \mid \text{out}@e := e@e \mid \text{assert}(e = e)@e \mid \\
\quad f(e, \dots, e) \mid c; c \mid \text{pre}(E) \mid \text{post}(E) \\
\text{binop} ::= + \mid - \mid * \mid ++ \\
v ::= w \mid \iota \mid \varepsilon \mid \{\ell = v; \dots; \ell = v\} \\
fn ::= f(y, \dots, y)\{e\} \mid f(y, \dots, y)\{c\} \\
\phi ::= r[e]@e \mid s[e]@e \mid m[e]@e \mid p[e] \mid \text{out}@e \mid \phi + \phi \mid \phi - \phi \mid \phi * \phi \\
E ::= \phi \equiv \phi \mid E \wedge E
\end{array}$$

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

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

$$\frac{e_1 \Rightarrow v_1 \dots e_n \Rightarrow v_n}{\{\ell_1 = e_1; \dots; \ell_n = e_n\} \Rightarrow \{\ell_1 = v_1; \dots; \ell_n = v_n\}} \quad \frac{e \Rightarrow \{\dots; \ell = v; \dots\}}{e.\ell \Rightarrow v} \quad \frac{e_1 \Rightarrow w_1 \quad e_2 \Rightarrow w_2}{e_1 ++ e_2 \Rightarrow w_1 w_2}$$

$$\begin{array}{c}
\frac{e_1 \Rightarrow \varepsilon_1 \quad e_2 \Rightarrow \varepsilon_2 \quad e \Rightarrow \iota}{(\pi, (E_1, E_2), \text{on}, \text{assert}(e_1 = e_2)@e) \Rightarrow (\pi; \text{assert}(\varepsilon_1 = \varepsilon_2)@_{\iota}, (E_1, E_2 \wedge \lfloor \varepsilon_1 @_{\iota} \rfloor = \lfloor \varepsilon_2 @_{\iota} \rfloor), \text{on})} \\
\\
\frac{e_1 \Rightarrow \varepsilon_1 \quad e_2 \Rightarrow \varepsilon_2 \quad e \Rightarrow \iota}{(\pi, (E_1, E_2), \text{off}, \text{assert}(e_1 = e_2)@e) \Rightarrow (\pi; \text{assert}(\varepsilon_1 = \varepsilon_2)@_{\iota}, (E_1, E_2, \text{off}))} \\
\\
\frac{e_1 \Rightarrow w \quad e_2 \Rightarrow \iota_1 \quad e_3 \Rightarrow \varepsilon \quad e_4 \Rightarrow \iota_2}{(\pi, (E_1, E_2), \text{on}, m[e_1]@_{e_2} := e_3@_{e_4}) \Rightarrow (\pi; m[w]@_{\iota_1} := \varepsilon@_{\iota_2}, (E_1 \wedge m[w]@_{\iota_1} = \lfloor \varepsilon @_{\iota_2} \rfloor, E_2), \text{on})} \\
\\
\frac{e_1 \Rightarrow w \quad e_2 \Rightarrow \iota_1 \quad e_3 \Rightarrow \varepsilon \quad e_4 \Rightarrow \iota_2}{(\pi, (E_1, E_2), \text{off}, m[e_1]@_{e_2} := e_3@_{e_4}) \Rightarrow (\pi; m[w]@_{\iota_1} := \varepsilon@_{\iota_2}, (E_1, E_1), \text{off})} \\
\\
(\pi, (E_1, E_2), \text{on}, \text{pre}(E)) \Rightarrow (\pi, E_1, E_2 \wedge E, \text{off}) \\
\\
(\pi, (E_1, E_2), \text{off}, \text{post}(E)) \Rightarrow (\pi, (E_1 \wedge E, E_2), \text{on}) \\
\\
\frac{(\pi_1, (E_{11}, E_{12}), \text{sw}_1, \mathbf{c}_1) \Rightarrow (\pi_2, (E_{21}, E_{22}), \text{sw}_2) \quad (\pi_2, (E_{21}, E_{22}), \text{sw}_2, \mathbf{c}_2) \Rightarrow (\pi_3, (E_{31}, E_{32}), \text{sw}_3)}{(\pi_1, (E_{11}, E_{12}), \text{sw}_1, \mathbf{c}_1; \mathbf{c}_2) \Rightarrow (\pi_3, (E_{31}, E_{32}), \text{sw}_3)} \\
\\
\frac{C(f) = y_1, \dots, y_n, \mathbf{c} \quad e_1 \Rightarrow v_1 \cdots e_n \Rightarrow v_n \quad (\pi_1, (E_{11}, E_{12}), \text{sw}_1, \mathbf{c}[v_1/y_1, \dots, v_n/y_n]) \Rightarrow (\pi_2, (E_{21}, E_{22}), \text{sw}_2)}{(\pi_1, (E_{11}, E_{12}), \text{sw}_1, f(e_1, \dots, e_n)) \Rightarrow (\pi_2, (E_{21}, E_{22}), \text{sw}_2)}
\end{array}$$

5 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)@i2
}

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

```



```

344         m[y++"m"]@i2 == m[y++"k"]@i1 + (m["delta"]@i1 * m[y++"s"]@i2));
345     m[z++"s"]@i2 := (m[x++"s"] + m[y++"s"]@i2);
346     m[z++"m"]@i2 := (m[x++"m"] + m[y++"m"]@i2);
347     m[z++"k"]@i1 := (m[x++"k"] + m[y++"k"]@i1);
348     post(m[z++"m"]@i2 == m[z++"k"]@i1 + (m["delta"]@i1 * m[z++"s"]@i2)
349 }
350
351 sum(z,x,y) { _sum(z,x,y,1,2);_sum(z,x,y,2,1) }
352
353 open(x) { _open(x,1,2); _open(x,2,1) }
354
355
356 sum("a","x","d");
357 open("d");
358 sum("b","y","e");
359 open("e");
360 let xys =
361     macsum(macctimes(macshare("b"), m["d"]),
362         macsum(macctimes(macshare("a"), m["e"]),
363             macshare("c")))
364 let xyk = mack("b") * m["d"] + mack("a") * m["e"] + mack("c")
365
366 secopen("a","x","d",1,2);
367 secopen("a","x","d",2,1);
368 secopen("b","y","e",1,2);
369 secopen("b","y","e",2,1);
370 let xys =
371     macsum(macctimes(macshare("b"), m["d"]),
372         macsum(macctimes(macshare("a"), m["e"]),
373             macshare("c")))
374 in
375 let xyk = mack("b") * m["d"] + mack("d") * m["d"] + mack("c")
376 in
377 secreveal(xys,xyk,"1",1,2);
378 secreveal(maccsum(xys,m["d"] * m["e"]),
379     xyk - m["d"] * m["e"],
380     "2",2,1);
381 out@1 := (p[1] + p[2])@1;
382 out@2 := (p[1] + p[2])@2;
383
384
385
386
387
388
389
390
391
392

```