1 *Overture* SYNTAX AND SEMANTICS

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

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

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

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

1.1 *Overture* Adversarial Semantics

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

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

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

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

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

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.

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

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

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

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

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.*

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

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

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

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 $R_1 \cap R_2 = \varnothing$.

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

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

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 $E_{pre} \wedge \lfloor \pi \rfloor \models E$.

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

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

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

$$\begin{array}{rcl} \varsigma & ::= & \text{High} \mid \text{Low} \\ \Gamma & ::= & \varnothing \mid \Gamma; x : \varsigma \end{array}$$

VALUE
$$\Gamma, E \vdash_\iota v : \text{High}$$

SECRET
$$\Gamma, E \vdash_\iota \mathtt{s[}w\mathtt{]} : \mathcal{L}(\iota)$$

RANDO
$$\Gamma, E \vdash_\iota \mathtt{r[}w\mathtt{]} : \mathcal{L}(\iota)$$

MESG
$$\Gamma, E \vdash_\iota \mathtt{m[}w\mathtt{]} : \Gamma(\mathtt{m[}w\mathtt{]@}\iota)$$

PUBM
$$\Gamma, E \vdash_\iota \mathtt{p[}w\mathtt{]} : \Gamma(\mathtt{p[}w\mathtt{]})$$

BINOP
$$\frac{\Gamma, E \vdash_\iota \varepsilon_1 : \varsigma \qquad \Gamma, E \vdash_\iota \varepsilon_2 : \varsigma \qquad \oplus \in \{+, -, *\}}{\Gamma, E \vdash_\iota \varepsilon_1 \oplus \varepsilon_2 : \varsigma}$$

INTEGRITYWEAKEN
$$\frac{\Gamma, E \vdash_\iota \varepsilon : \varsigma_1 \qquad \varsigma_1 \preceq \varsigma_2}{\Gamma, E \vdash_\iota \varepsilon : \varsigma_2}$$

SEND
$$\frac{\Gamma, E \vdash_\iota \varepsilon : \mathcal{L}(\iota)}{\Gamma, E \vdash x := \varepsilon @ \iota : \Gamma; x : \mathcal{L}(\iota)}$$

SEQ
$$\frac{\Gamma_1, E \vdash \pi_1 : \Gamma_2 \qquad \Gamma_2, E \vdash \pi_2 : \Gamma_3}{\Gamma_1, E \vdash \pi_1; \pi_2 : \Gamma_3}$$

MAC
$$\frac{E \models \lfloor \mathtt{assert}(\psi_{BDOZ}(w))\mathtt{@}\iota \rfloor}{\Gamma, E \vdash \mathtt{assert}(\psi_{BDOZ}(w))\mathtt{@}\iota : \Gamma; \mathtt{m[}ws\mathtt{]@}\iota : \text{High}}$$

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

Definition 2.4. Given H, C and pre-processing predicate E_{pre} defining M, define $init(E_{pre}) \triangleq \Gamma$ where for all $\texttt{m[}w\texttt{]@}\iota \in M$ we have $\Gamma(\texttt{m[}w\texttt{]@}\iota) = \mathcal{L}(\iota)$.

Definition 2.5. Given pre-processing predicate E_{pre} and protocol π, for all H, C the judgement $init(E_{pre}), E \vdash \pi : \Gamma$ is valid iff it is derivable and $E_{pre} \wedge \lfloor \pi \rfloor \models E$.

Theorem 2.6. *Given pre-processing predicate E_{pre} and protocol π, if for all H, C the judgement $init(E_{pre}), E \vdash \pi : \Gamma$ is valid and for all $x \in V_{H \triangleright C}$ we have $\Gamma(x) =$ High, then cheating is detectable in π.*

3 COMPOSITIONAL TYPE VERIFICATION IN *Prelude*

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

$$x \quad ::= \quad \texttt{r[}e\texttt{]@}e \mid \texttt{s[}e\texttt{]@}e \mid \texttt{m[}e\texttt{]@}e \mid \texttt{p[}e\texttt{]} \mid \texttt{out@}e$$

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

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

$$R \Vdash x : (\varnothing, \{x\}) \qquad \frac{R \Vdash \phi : (R_1, T) \qquad \texttt{r[}w\texttt{]@}\iota \notin R \qquad \oplus \in \{+, -\}}{R_1 \Vdash \phi \oplus \texttt{r[}w\texttt{]@}\iota : (R_1 \cup \{\texttt{r[}w\texttt{]@}\iota\}, \{c(\texttt{r[}w\texttt{]@}\iota, T)\})}$$

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

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

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

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

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

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

Mesg

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

Encode

$$\frac{e_1 \Rightarrow w \qquad e_2 \Rightarrow \iota \qquad e_3 \Rightarrow \phi \qquad E \models \lfloor \varepsilon @ \iota \rfloor \equiv \phi \qquad R_1 \Vdash \phi : (R_2, T)}{R_1 \vdash \mathtt{m}[e_1]@e_2\ \mathtt{as}\ e_3 : \{E\}\ \mathtt{m}[w]@\iota : T, R_1; R_2\ \{E \wedge \mathtt{m}[w]@i \equiv \phi\}}$$

App

$$\frac{\begin{array}{c} \mathrm{sig}(f) = \Pi x_1, \ldots, x_n . \{\breve{E}_1\}\ \breve{\Gamma}, \breve{R}\ \{\breve{E}_2\} \qquad e_1 \Rightarrow v_1 \ \cdots \ e_n \Rightarrow v_n \\ \rho = [v_1/x_1] \cdots [v_n/x_n] \qquad \rho(\{\breve{E}_1\}\ \breve{\Gamma}, \breve{R}\ \{\breve{E}_2\}) \Rightarrow \{E_1\}\ \Gamma, R\ \{E_2\} \qquad E \models E_1 \end{array}}{R_1 \vdash f(e_1, \ldots, e_n) : \{E\}\ \Gamma, R_1; R\ \{E \wedge E_2\}}$$

Seq

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

Sig

$$\frac{\begin{array}{c} C(f) = x_1, \ldots, x_n, \mathbf{c} \qquad \rho = [v_1/x_1] \cdots [v_n/x_n] \\ \rho(\{\breve{E}_1\}\ \breve{\Gamma}, \breve{R}\ \{\breve{E}_2\}) \Rightarrow \{E_1\}\ \Gamma, R\ \{E_2\} \qquad \varnothing \vdash \rho(\mathbf{c}) : \{E_1\}\ \Gamma, R\ \{E\} \qquad E \models E_2 \end{array}}{f : \Pi x_1, \ldots, x_n . \{\breve{E}_1\}\ \breve{\Gamma}, \breve{R}\ \{\breve{E}_2\}}$$

Definition 3.1. 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 3.2. *If* sig *is verified and* $\varnothing \vdash e : \{\varnothing\}\ \Gamma, R\ \{E\}$ *then* $e \Rightarrow \pi$ *and* $R, E \vdash \pi : \Gamma$ *is valid.*

3.1 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 }
```

3.2 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;
```