

**Algorithm 1**  $R^r$ 

**Input:**  $\langle a, f, m \rangle, s$

- 1: **let**  $s := s'$
- 2: **let**  $n, method, path, parameters, headers, body$  **such that**  
 $\langle \text{HTTPReq}, n, method, path, parameters, headers, body \rangle \equiv m$   
**if possible; otherwise stop**  $\langle \rangle, s'$
- 3: **if**  $path \equiv /script$  **then**
- 4:   **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{RPScript} \rangle$
- 5:   **stop**  $\langle f, a, m' \rangle, s'$
- 6: **else if**  $path \equiv /login$  **then**
- 7:   **let**  $m' := \langle \text{HTTPResp}, n, 302, \langle \langle Location, s'.IdP.ScriptUrl \rangle \rangle, \langle \rangle \rangle$
- 8:   **stop**  $\langle f, a, m' \rangle, s'$
- 9: **else if**  $path \equiv /startNegotiation$  **then**
- 10:   **let**  $cookie := headers[Cookie]$
- 11:   **let**  $session := s'.sessions[cookie]$
- 12:   **let**  $N_U := parameters[N_U]$
- 13:   **let**  $PID_{RP} := \text{ModPow}(s'.ID_{RP}, N_U, s'.IdP.p)$
- 14:   **let**  $tT := \text{ExEU}(N_U, s'.IdP.q)$
- 15:   **let**  $session[N_U] := N_U$
- 16:   **let**  $session[PID_{RP}] := PID_{RP}$
- 17:   **let**  $session[t] := T$
- 18:   **let**  $session[state] := expectRegistration$
- 19:   **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \langle Cert, s'.Cert \rangle \rangle$
- 20:   **stop**  $\langle f, a, m' \rangle, s'$
- 21: **else if**  $path \equiv /registrationResult$  **then**
- 22:   **let**  $cookie := headers[Cookie]$
- 23:   **let**  $session := s'.sessions[cookie]$
- 24:   **if**  $session[state] \neq expectRegistration$  **then**
- 25:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 26:     **stop**  $\langle f, a, m' \rangle, s'$
- 27:   **end if**
- 28:   **let**  $RegistrationResult := body[RegistrationResult]$
- 29:   **let**  $Content := RegistrationResult.Content$
- 30:   **if**  $\text{checksig}(Content, RegistrationResult.Sig, s'.IdP.PubKey) \equiv \text{FALSE}$  **then**
- 31:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 32:     **let**  $session := \text{null}$
- 33:     **stop**  $\langle f, a, m' \rangle, s'$
- 34:   **end if**
- 35:   **if**  $Content.Result \neq OK$  **then**
- 36:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 37:     **let**  $session := \text{null}$
- 38:     **stop**  $\langle f, a, m' \rangle, s'$
- 39:   **end if**
- 40:   **let**  $PID_{RP} := session[PID_{RP}]$
- 41:   **let**  $N_U := session[N_U]$
- 42:   **let**  $Nonce := \text{Hash}(N_U)$
- 43:   **let**  $Time := \text{CurrentTime}()$
- 44:   **if**  $PID_{RP} \neq Content.PID_{RP} \vee Nonce \neq Content.Nonce \vee Time > Content.Validity$  **then**
- 45:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 46:     **let**  $session := \text{null}$
- 47:     **stop**  $\langle f, a, m' \rangle, s'$
- 48:   **end if**
- 49:   **let**  $session[PIDValidity] := Content.Validity$
- 50:   **let**  $Endpoint \in s'.Endpoints$
- 51:   **let**  $session[state] := expectToken$
- 52:   **let**  $Nonce' := \text{Random}()$
- 53:   **let**  $session[Nonce] := Nonce'$
- 54:   **let**  $Body := \langle PID_{RP}, Endpoint, Nonce' \rangle$
- 55:   **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, Body \rangle$
- 56:   **stop**  $\langle f, a, m' \rangle, s'$
- 57: **else if**  $path \equiv /uploadToken$  **then**
- 58:   **let**  $cookie := headers[Cookie]$
- 59:   **let**  $session := s'.sessions[cookie]$
- 60:   **if**  $session[state] \neq expectToken$  **then**
- 61:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 62:     **stop**  $\langle f, a, m' \rangle, s'$
- 63:   **end if**
- 64:   **let**  $Token := body[Token]$
- 65:   **if**  $\text{checksig}(Token.Content, Token.Sig, s'.IdP.PubKey) \equiv \text{FALSE}$  **then**
- 66:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 67:     **stop**  $\langle f, a, m' \rangle, s'$
- 68:   **end if**
- 69:   **let**  $PID_{RP} := session[PID_{RP}]$
- 70:   **let**  $Time := \text{CurrentTime}()$
- 71:   **let**  $PIDValidity := session[PIDValidity]$
- 72:   **let**  $Content := Token.Content$
- 73:   **if**  $PID_{RP} \neq Content.PID_{RP} \vee Time > Content.Validity \vee Time > PIDValidity$  **then**
- 74:     **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$
- 75:     **stop**  $\langle f, a, m' \rangle, s'$
- 76:   **end if**
- 77:   **let**  $PID_U := Content.PID_U$
- 78:   **let**  $T := session[t]$
- 79:   **let**  $Account := \text{ModPow}(PID_U, T, s'.IdP.p)$
- 80:   **if**  $Account \in \text{ListOfUser}()$  **then**
- 81:     **let**  $\text{RegisterUser}(Account)$
- 82:   **end if**
- 83:   **let**  $session[user] := Account$
- 84:   **let**  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{LoginSuccess} \rangle$
- 85:   **stop**  $\langle f, a, m' \rangle, s'$
- 86: **end if**
- 87: **stop**  $\langle \rangle, s'$