Algorithm 1 R^r

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

let $s := s'$
let $n, method, path, parameters, headers, body$ **such that** $\langle \texttt{HTTPReq}, n, method, path, parameters, headers, body\rangle \equiv m$ **if** *possible*; **otherwise** stop $\langle\rangle, s'$
if $path \equiv /script$ **then**
  let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{RPScript}\rangle$
  stop $\langle f, a, m'\rangle, s'$
else if $path \equiv /login$ **then**
  let $m' := \langle \texttt{HTTPResp}, n, 302, \langle\langle Location, s'.IdP.ScriptUrl\rangle\rangle, \langle\rangle\rangle$
  stop $\langle f, a, m'\rangle, s'$
else if $path \equiv /startNegotiation$ **then**
  let $cookie := headers[Cookie]$
  let $session := s'.sessions[cookie]$
  let $N_U := parameters[N_U]$
  let $PID_{RP} := \texttt{ModPow}(s'.ID_{RP}, N_U, s'.IdP.p)$
  let $tT := \texttt{ExEU}(N_U, s'.IdP.q)$
  let $session[N_U] := N_U$
  let $session[PID_{RP}] := PID_{RP}$
  let $session[t] := T$
  let $session[state] := expectRegistration$
  let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \langle Cert, s'.Cert\rangle\rangle$
  stop $\langle f, a, m'\rangle, s'$
else if $path \equiv /registrationResult$ **then**
  let $cookie := headers[Cookie]$
  let $session := s'.sessions[cookie]$
  if $session[state] \not\equiv expectRegistration$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $RegistrationResult := body[RegistrationResult]$
  let $Content := RegistrationResult.Content$
  if $\texttt{checksig}(Content, RegistrationResult.Sig, s'.IdP.PubKey) \equiv \texttt{FALSE}$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    let $session := \texttt{null}$
    stop $\langle f, a, m'\rangle, s'$
  end if
  if $Content.Result \not\equiv OK$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    let $session := \texttt{null}$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $PID_{RP} := session[PID_{RP}]$
  let $N_U := session[N_U]$
  let $Nonce := \texttt{Hash}(N_U)$
  let $Time := \texttt{CurrentTime}()$
  if $PID_{RP} \not\equiv Content.PID_{RP} \vee Nonce \not\equiv Content.Nonce \vee Time > Content.Validity$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    let $session := \texttt{null}$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $session[PIDValidity] := Content.Validity$
  let $Endpoint \in s'.Endpoints$
  let $session[state] := expectToken$
  let $Nonce' := \texttt{Random}()$
  let $session[Nonce] := Nonce'$
  let $Body := \langle PID_{RP}, Endpoint, Nonce'\rangle$
  let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, Body\rangle$
  stop $\langle f, a, m'\rangle, s'$
else if $path \equiv /uploadToken$ **then**
  let $cookie := headers[Cookie]$
  let $session := s'.sessions[cookie]$
  if $session[state] \not\equiv expectToken$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $Token := body[Token]$
  if $\texttt{checksig}(Token.Content, Token.Sig, s'.IdP.PubKey) \equiv \texttt{FALSE}$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $PID_{RP} := session[PID_{RP}]$
  let $Time := \texttt{CurrentTime}()$
  let $PIDValidity := session[PIDValidity]$
  let $Content := Token.Content$
  if $PID_{RP} \not\equiv Content.PID_{RP} \vee Time > Content.Validity \vee Time > PIDValidity$ **then**
    let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
    stop $\langle f, a, m'\rangle, s'$
  end if
  let $PID_U := Content.PID_U$
  let $T := session[t]$
  let $Account := \texttt{ModPow}(PID_U, T, s'.IdP.p)$
  if $Account \in \texttt{ListOfUser}()$ **then**
    let $\texttt{RegisterUser}(Account)$
  end if
  let $session[user] := Account$
  let $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{LoginSuccess}\rangle$
  stop $\langle f, a, m'\rangle, s'$
end if
stop $\langle\rangle, s'$