

C Algorithms

1 Algorithm idp

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

```
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  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{IdPScript} \rangle$ 
5:   stop  $\langle b, a, m' \rangle, s'$ 
6: else if  $path \equiv /authentication$  then
7:   let  $cookie := headers[Cookie]$ 
8:   let  $session := s'.SessionList[cookie]$ 
9:   let  $uname := body[uname]$ 
10:  let  $passwd := body[passwd]$ 
11:  if  $passwd \neq \text{PasswdOfUser}(uname)$  then
12:    let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{LoginFailure} \rangle$ 
13:    stop  $\langle b, a, m' \rangle, s'$ 
14:  end if
15:  let  $session[uid] := \text{UIDOfUser}(uname)$ 
16:  let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{LoginSuccess} \rangle$ 
17:  stop  $\langle b, a, m' \rangle, s'$ 
18: else if  $path \equiv /reqToken$  then
19:   let  $cookie := headers[Cookie]$ 
20:   let  $session := s'.SessionList[cookie]$ 
21:   let  $IDTokens := session[IDTokens]$ 
22:   if  $IDTokens[body[PID_{RP}]] \neq \text{null}$  then
23:     let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, IDTokens[body[PID_{RP}]] \rangle$ 
24:     stop  $\langle b, a, m' \rangle, s'$ 
25:   end if
26:   let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Unauthenticated} \rangle$ 
27:   stop  $\langle b, a, m' \rangle, s'$ 
28: else if  $path \equiv /authorize$  then
29:   let  $cookie := headers[Cookie]$ 
30:   let  $session := s'.SessionList[cookie]$ 
31:   let  $uid := session[uid]$ 
32:   if  $uid \equiv \text{null}$  then
33:     let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$ 
34:     stop  $\langle b, a, m' \rangle, s'$ 
35:   end if
36:   let  $PID_{RP} := parameters[PID_{RP}]$ 
37:   if  $\text{IsValid}(PID_{RP}) \equiv \text{FALSE}$  then
38:     let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$ 
39:     stop  $\langle b, a, m' \rangle, s'$ 
```

```

40: end if
41: if  $\text{IsInScope}(uid, body[Attr]) \equiv \text{FALSE}$  then
42:   let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$ 
43:   stop  $\langle b, a, m' \rangle, s'$ 
44: end if
45: let  $ID_U := session[uid]$ 
46: let  $PID_U := \text{Multiply}(PID_{RP}, ID_U)$ 
47: let  $Validity := \text{CurrentTime}() + s'.Validity$ 
48: let  $Content := \langle PID_{RP}, PID_U, s'.Issuer, Validity \rangle$ 
49: let  $Sig := \text{SigSign}(Content, s'.SK)$ 
50: let  $IDToken := \langle Content, Sig \rangle$ 
51: let  $session[IDTokens] := session[IDTokens] + \langle \rangle \langle PID_{RP}, IDToken \rangle$ 
52: let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, Token \rangle$ 
53: stop  $\langle b, a, m' \rangle, s'$ 
54: end if
55: stop  $\langle \rangle, s'$ 

```

2 Algorithm rp

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

```

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  $\langle b, a, m' \rangle, s'$ 
6: else if  $path \equiv /loginSSO$  then
7:   let  $m' := \langle \text{HTTPResp}, n, 302, \langle \langle \text{Location}, s'.IdP.ScriptUrl \rangle \rangle, \langle \rangle \rangle$ 
8:   stop  $\langle b, a, m' \rangle, s'$ 
9: else if  $path \equiv /startNegotiation$  then
10:  let  $cookie := headers[Cookie]$ 
11:  let  $session := s'.SessionList[cookie]$ 
12:  let  $t := body[t]$ 
13:  let  $t^{-1} := \text{Inverse}(t)$ 
14:  let  $session[t^{-1}] := t^{-1}$ 
15:  let  $session[state] := expectToken$ 
16:  let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \langle \text{Cert}_{RP}, s'.Cert_{RP} \rangle \rangle$ 
17:  stop  $\langle b, a, m' \rangle, s'$ 
18: else if  $path \equiv /uploadToken$  then
19:  let  $cookie := headers[Cookie]$ 
20:  let  $session := s'.SessionList[cookie]$ 
21:  if  $session[state] \neq expectToken$  then
22:    let  $m' := \langle \text{HTTPResp}, n, 200, \langle \rangle, \text{Fail} \rangle$ 
23:    stop  $\langle b, a, m' \rangle, s'$ 
24:  end if

```

```

25:   let  $IDToken := body[IDToken]$ 
26:   if  $checksig(IDToken.Content, IDToken.Sig, s'.IdP.PK) \equiv FALSE$  then

27:     let  $m' := \langle HTTPResp, n, 200, \langle \rangle, Fail \rangle$ 
28:     stop  $\langle b, a, m' \rangle, s'$ 
29:   end if
30:   let  $Time := CurrentTime()$ 
31:   let  $Content := Token.Content$ 
32:   if  $Time > Content.Validity$  then
33:     let  $m' := \langle HTTPResp, n, 200, \langle \rangle, Fail \rangle$ 
34:     stop  $\langle b, a, m' \rangle, s'$ 
35:   end if
36:   let  $PID_U := Content.PID_U$ 
37:   let  $t^{-1} := session[t^{-1}]$ 
38:   let  $Acct := Multiply(PID_U, t^{-1})$ 
39:   if  $Acct \notin ListOfUser()$  then
40:     let  $AddUser(Acct)$ 
41:   end if
42:   let  $session[user] := Acct$ 
43:   let  $s'.serviceTokens := s'.serviceTokens + \langle \rangle \langle IDToken, Acct \rangle$ 
44:   let  $m' := \langle HTTPResp, n, 200, \langle \rangle, LoginSuccess \rangle$ 
45:   stop  $\langle b, a, m' \rangle, s'$ 
46: end if
47: stop  $\langle \rangle, s'$ 

```

3 Algorithm script_idp

Input: $\langle tree, docID, scriptstate, scriptinputs, cookies, ids, secret \rangle$

```

1: let  $s' := scriptstate$ 
2: let  $command := \langle \rangle$ 
3: let  $target := PARENTWINDOW(tree, docID)$ 
4: let  $IdPDomain := s'.IdPDomain$ 
5: switch  $s'.phsae$  do
6:   case start:
7:     let  $t := Random()$ 
8:     let  $command := \langle POSTMESSAGE, target, \langle t, t \rangle, null \rangle$ 
9:     let  $s'.Parameters[t] := t$ 
10:    let  $s'.phase := expectCert$ 
11:   case expectCert:
12:     let  $pattern := \langle POSTMESSAGE, target, *, \langle Cert_{RP}, * \rangle \rangle$ 
13:     let  $input := CHOOSEINPUT(scriptinputs, pattern)$ 
14:     if  $input \neq null$  then
15:       let  $Cert_{RP} := \pi_2(\pi_4(input))$ 
16:       if  $checksig(Cert.Content, Cert.Sig, s'.PubKey) \equiv null$  then
17:         let stop  $\langle \rangle$ 
18:       end if

```

```

19:   let  $s'.Parameters[Cert] := Cert_{RP}$ 
20:   let  $t := s'.Parameters[t]$ 
21:   let  $PID_{RP} := \text{Multiply}(Cert_{RP}.ID_{RP}, t)$ 
22:   let  $s'.Parameters[PID_{RP}] := PID_{RP}$ 
23:   let  $Url := \langle URL, S, IdPDomain, /reqToken, \langle \langle PID_{RP}, PID_{RP} \rangle \rangle \rangle$ 
24:   let  $command := \langle XMLHTTPREQUEST, Url, GET, \langle \rangle, s'.refXHR \rangle$ 
25:   let  $s'.phase := expectLoginState$ 
26: end if
27: case expectReqToken:
28:   let  $pattern := \langle XMLHTTPREQUEST, Body, s'.refXHR \rangle$ 
29:   let  $input := \text{CHOOSEINPUT}(scriptinputs, pattern)$ 
30:   if  $input \neq \text{null}$  then
31:     if  $\pi_2(input) \equiv \text{Unauthenticated}$  then
32:       let  $user \in ids$ 
33:       let  $Url := \langle URL, S, IdPDomain, /authentication, \langle \rangle \rangle$ 
34:       let  $command := \langle XMLHTTPREQUEST, Url, POST, \langle \langle \text{uname}, \text{uname} \rangle, \langle \text{passwd}, \text{passwd} \rangle \rangle, s'.refXHR \rangle$ 
35:       let  $s'.phase := expectLoginResult$ 
36:     end if
37:     let  $IDToken := \pi_2(input)[IDToken]$ 
38:     let  $RPOrigin := \langle s'.Parameters[Cert].Enpt, S \rangle$ 
39:     let  $command := \langle POSTMESSAGE, target, \langle IDToken, IDToken \rangle, RPOrigin \rangle$ 
40:     let  $s.phase := stop$ 
41:   end if
42: case expectLoginResult:
43:   let  $pattern := \langle XMLHTTPREQUEST, Body, s'.refXHR \rangle$ 
44:   let  $input := \text{CHOOSEINPUT}(scriptinputs, pattern)$ 
45:   if  $input \neq \text{null}$  then
46:     if  $\pi_2(input) \neq \text{LoginSuccess}$  then
47:       let stop  $\langle \rangle$ 
48:     end if
49:     let  $PID_{RP} := s'.Parameters[PID_{RP}]$ 
50:     let  $Url := \langle URL, S, IdPDomain, /authorize, \langle \langle PID_{RP}, PID_{RP} \rangle, \langle Attr, Attr \rangle \rangle \rangle$ 
51:     let  $command := \langle XMLHTTPREQUEST, Url, GET, \langle \rangle, s'.refXHR \rangle$ 
52:     let  $s'.phase := expectToken$ 
53:   end if
54: case expectToken:
55:   let  $pattern := \langle XMLHTTPREQUEST, Body, s'.refXHR \rangle$ 
56:   let  $input := \text{CHOOSEINPUT}(scriptinputs, pattern)$ 
57:   if  $input \neq \text{null}$  then
58:     let  $IDToken := \pi_2(input)[IDToken]$ 
59:     let  $RPOrigin := \langle s'.Parameters[Cert].Enpt, S \rangle$ 
60:     let  $command := \langle POSTMESSAGE, target, \langle IDToken, IDToken \rangle, RPOrigin \rangle$ 
61:     let  $s.phase := stop$ 
62:   end if
63: end switch
64: let stop  $\langle s', cookies, localStorage, sessionStorage, command \rangle$ 

```

4 Algorithm script_rp

Input: $\langle tree, docID, scriptstate, scriptinputs, cookies, ids, secret \rangle$

```
1: let  $s' := scriptstate$ 
2: let  $command := \langle \rangle$ 
3: let  $IdPWindow := SUBWINDOW(tree, docnonce).winID$ 
4: let  $RPDomain := s'.RPDomain$ 
5: let  $IdPOrigin := \langle s'.IdPDomain, S \rangle$ 
6: switch  $s'.phase$  do
7:   case start:
8:     let  $Url := \langle URL, S, RPDomain, /loginSSO, \langle \rangle \rangle$ 
9:     let  $command := \langle IFRAME, Url, SELF \rangle$ 
10:    let  $s'.phase := expectt$ 
11:   case expectt:
12:     let  $pattern := \langle POSTMESSAGE, target, *, \langle t, * \rangle \rangle$ 
13:     let  $input := CHOOSEINPUT(scriptinputs, pattern)$ 
14:     if  $input \neq null$  then
15:       let  $t := \pi_2(\pi_4(input))[t]$ 
16:       let  $Url := \langle URL, S, RPDomain, /startNegotiation, \langle \rangle \rangle$ 
17:       let  $command := \langle XMLHTTPREQUEST, Url, POST, \langle \langle t, t \rangle \rangle, s'.refXHR \rangle$ 
18:       let  $s'.phase := expectCert$ 
19:     end if
20:   case expectCert:
21:     let  $pattern := \langle XMLHTTPREQUEST, Body, s'.refXHR \rangle$ 
22:     let  $input := CHOOSEINPUT(scriptinputs, pattern)$ 
23:     if  $input \neq null$  then
24:       let  $Cert_{RP} := \pi_2(input)[Cert_{RP}]$ 
25:       let  $command := \langle POSTMESSAGE, IdPWindow, \langle \langle Cert, Cert \rangle \rangle, IdPOrigin \rangle$ 
26:       let  $s'.phase := expectToken$ 
27:     end if
28:   case expectToken:
29:     let  $pattern := \langle POSTMESSAGE, target, *, \langle IDToken, * \rangle \rangle$ 
30:     let  $input := CHOOSEINPUT(scriptinputs, pattern)$ 
31:     if  $input \neq null$  then
32:       let  $IDToken := \pi_2(input)[IDToken]$ 
33:       let  $Url := \langle URL, S, RPDomain, /uploadToken, \langle \rangle \rangle$ 
34:       let  $command := \langle XMLHTTPREQUEST, Url, POST, \langle \langle IDToken, IDToken \rangle \rangle, s'.refXHR \rangle$ 
35:       let  $s'.phase := expectLoginResult$ 
36:     end if
37:   case expectLoginResult:
38:     let  $pattern := \langle XMLHTTPREQUEST, Body, s'.refXHR \rangle$ 
39:     let  $input := CHOOSEINPUT(scriptinputs, pattern)$ 
40:     if  $input \neq null$  then
41:       if  $\pi_2(input) \equiv LoginSuccess$  then
42:         let Load Homepage
```

```
43:         end if  
44:     end if  
45: end switch  
46: let stop  $\langle s', cookies, localStorage, sessionStorage, command \rangle$ 
```
