Lecture 24 - Zero Knowledge Identification System

Paper: https://arrow.dit.ie/cgi/viewcontent.cgi?article=1031&context=itbj

Look at page 38, section 7.9. to 7.12 on page 43 (this is the page numbers in the PDF - it is the 28th page OF the PDF).

Walk through of algorithm with the example from the paper.

Also see: https://blog.cryptographyengineering.com/2017/01/21/zero-knowledge-proofs-an-illustrated-primer-part-2/

Also: the reading from last time - has a nice section on this method for identification of users.

First a tiny detour - how to authenticate a QR or RFID tag.

QR codes encode some sort of text. Some RFID tags encode just a number. Others like NFC encode a chunk of text between 84 and 1084 characters long. Usually these chunks of text are URLs to some website or set of information. People would like to use the tags as proof-of-authenticity. The problem is how to get them to be secure. NFC tags can have computation built int - but it is really a small amount - the power for the NFC is coming from radio waves being transmitted from the source. So you take you Android and its tiny transmitter and send the NFC tag a tiny bit of power - that then looses lots of power because it has to be picked up by a tiny antenna in the “chip” and then used to do a small, small amount of computation and then using that same power send back an answer. So building any kind of “authentication” that is meaningful into the chip is difficult. A QR code is a static image - so it will not do any computation at all.

So if you can’t do the authentication on the chip or device. What about just adding Two Factor Authentication after the device is scanned. Basically have the QR or RFID send you to a page where you have to authenticate using a device (iPhone, Android) and provide proof of your authenticity at that point.

Zero knowledge proof for use as ID

```
1 package main
2
3 import (
4     "fmt"
5     "math/big"
```

```
 6     "math/rand"
 7     "time"
 8
 9     "github.com/pschlump/MiscLib"
10 )
11
12 // From: https://arrow.dit.ie/cgi/viewcontent.cgi?article=1031&context=itbj
13 // IdProtocalsInCrypto.pdf
14
15 type DBRecord struct {
16     v *big.Int
17     e *big.Int
18     y *big.Int
19     x *big.Int
20 }
21
22 var database map[string]*DBRecord
23
24 func init() {
25     database = make(map[string]*DBRecord)
26 }
27
28 func main() {
29
30     rand.Seed(time.Now().UnixNano())
31
32     // ------------------------------------------------------------
33     // Registration and Setup
34     // ------------------------------------------------------------
35
36     // From p40
37     p := big.NewInt(88667) // password or hash of password to convert pw to number
38
39     q := big.NewInt(1031)      // value 1 = 'q', Pre Chosen : large prime
40     alpha := big.NewInt(70322) // value 2 == 'alpha', devisor of (p-1)
41     a := big.NewInt(755)       // value 3 == 'a', alpha = (beta**((p-1)/q))mod p
42     {
43
44         // v = (alpha ^ ( q-a )) % p
45         t1 := big.NewInt(0)
46         t1.Sub(q, a)
47         v := big.NewInt(0)
48         v.Exp(alpha, t1, p) // note the 'p' is the "mod"
49
50         fmt.Printf("Setup Complete: v=%s\n", v)
51         fmt.Printf(`"Save 'v' for user "alice"` + "\n")
52
53         fmt.Printf(`%s/api/register-user%s, send-data=%s
               {"user":"alice","v":%d}%s`+"\n",
54              MiscLib.ColorYellow, MiscLib.ColorReset, MiscLib.ColorYellow,
               v, MiscLib.ColorReset)
```

```
55          // Save the validation value 'v' for "alice" in the database.
56          database["alice"] = &DBRecord{v: v}
57          fmt.Printf(`%sResponse: {"status":"success", "username":"alice",
                "msg":"is registered."}%s`+"\n",
58              MiscLib.ColorCyan, MiscLib.ColorReset)
59      }
60
61      // Alice is the Client:
62
63      // -----------------------------------------------------------
64      // Message 1 - Client to Server
65      // -----------------------------------------------------------
66
67      // Alice Chooses, and send to Bob
68      // r := big.NewInt(543) // Should be random, but for this example
69      randNum := genRan(999)
70      fmt.Printf("random genrated: %d\n", randNum)
71      r := big.NewInt(randNum)
72      x := big.NewInt(0)
73      x.Exp(alpha, r, p) // x=(alpha^r) % p
74
75      fmt.Printf("Send To Bob : x=%s\n", x)
76      fmt.Printf(`%s/api/login%s, send-data=%s{"username":"alice","x":%d}%s`+"\n",
77          MiscLib.ColorYellow, MiscLib.ColorReset,
            MiscLib.ColorYellow, x, MiscLib.ColorReset)
78      dbr := database["alice"]
79      v := dbr.v
80      fmt.Printf(`Server looks up in the database 'v' for "alice", v=%d`+"\n", v)
81
82      // ------------------------------------------------------------------------
83      // Response to Message 1, Server back to client
84      // ------------------------------------------------------------------------
85      {
86          dbr := database["alice"]
87          dbr.x = x
88          database["alice"] = dbr
89      }
90
91      y := big.NewInt(0)
92      // Bob is the Server:
93      // Bob sends the challenge 'e' back to Alice e to do the computation
94      // e := big.NewInt(1000) // how chose (random?)
95      randNum = genRan(999)
96      e := big.NewInt(randNum)
97      {
98          dbr := database["alice"]
99          dbr.e = e
100         database["alice"] = dbr
101     }
102     fmt.Printf(`%sResponse: {"status":"success", "e":%d}%s`+"\n",
            MiscLib.ColorCyan, e, MiscLib.ColorReset)
```

```
103     {
104
105         // Alice(client) now computes: y = a*e % q
106         t2 := big.NewInt(0)
107         t2.Mul(a, e)
108         t2.Mod(t2, q) // 45664
109         t2.Add(t2, r) // 851 is correct
110
111         y = t2
112         fmt.Printf("y=%s\n", y) // Prints 851
113         fmt.Printf(`%s/api/login-pt1%s, send-data:
                %s{"username":%q,"y":%d}%s`+"\n",
114             MiscLib.ColorYellow, MiscLib.ColorReset, MiscLib.ColorYellow,
                "alice", y, MiscLib.ColorReset)
115         fmt.Printf(`response: %s{"status":"success","y":%d}%s`+"\n",
116             MiscLib.ColorCyan, y, MiscLib.ColorReset)
117     }
118     {
119         dbr := database["alice"]
120         dbr.y = y
121         database["alice"] = dbr
122     }
123
124     // At this point.
125     // Alice has 'y' - by calculating it from 'e'
126     // Bob has 'y' saved in database.
127
128     // ------------------------------------------------------------------------
129     // Message 2 - Client (Alice) with response to challenge.
130     // ------------------------------------------------------------------------
131
132     z := big.NewInt(0)
133     {
134         // Bob (server) verifies: x == z == (a^y) * (v^e) % p
135         // or a Better version of the same calulation
136         // Bob (server) verifies: x == z == ((a^y)%p)*((v^e)%p) % p
137
138         dbr := database["alice"]
139         v := dbr.v // Validation value
140         e := dbr.e // random saved from earlier
141         y := dbr.y // calculated on server and saved.
142         fmt.Printf(`Server looks up in the database 'v','e','y'
                for "alice", v=%d`+"\n", v)
143
144         t3 := big.NewInt(0)
145         t3.Exp(alpha, y, p)
146         t4 := big.NewInt(0)
147         t4.Exp(v, e, p)
148         t5 := big.NewInt(0)
149         t5.Mul(t3, t4)
150         t5.Mod(t5, p)
```

```
151         z = t5
152     }
153
154     fmt.Printf("z=%s\n", z)
155     fmt.Printf(`%s/api/login-pt2%s, send-data: %s{"username":"alice"}%s`+"\n",
156         MiscLib.ColorYellow, MiscLib.ColorReset,
            MiscLib.ColorYellow, MiscLib.ColorReset)
157
158     // ------------------------------------------------------------------------
159     // Response 2 - Success/Fail message from server back to client
160     // ------------------------------------------------------------------------
161
162     {
163         // fetch 'x' from earlier
164         dbr := database["alice"]
165         x = dbr.x
166
167         if x.Cmp(z) == 0 {
168             fmt.Printf("%sAuthoized! Yea, 'alice' is a valid user%s\n",
                    MiscLib.ColorGreen, MiscLib.ColorReset)
169             fmt.Printf(`%sResponse: {"status":"success",
                        "msg":"'alice' is logged in"}%s`+"\n",
170                 MiscLib.ColorCyan, MiscLib.ColorReset)
171         } else {
172             fmt.Printf("%sNope nope nope%s\n", MiscLib.ColorRed, MiscLib.ColorReset
173             fmt.Printf(`%sResponse: {"status":"error",
                    "msg":"'alice' is not a valid user"}%s`+"\n",
174                 MiscLib.ColorRed, MiscLib.ColorReset)
175         }
176     }
177
178 }
179
180 func genRan(m int) int64 {
181     return int64(rand.Intn(m))
182 }
```

Reunsts of 2 runs:

2-runs.png