

Lecture 24 - Zero Knowledge Identification System

Videos

https://youtu.be/x1papRZw_68 - Lect-24-4010-5010-pt1.mp4

<https://youtu.be/tDBTTqFuSw0> - Lect-24-4010-5010-pt2.mp4

<https://youtu.be/ifDcsPiglv0> - Lect-24-4010-5010-pt3.mp4

From Amazon S3 - for download (same as youtube videos)

<http://uw-s20-2015.s3.amazonaws.com/Lect-24-4010-5010-pt1.mp4>

<http://uw-s20-2015.s3.amazonaws.com/Lect-24-4010-5010-pt2.mp4>

<http://uw-s20-2015.s3.amazonaws.com/Lect-24-4010-5010-pt3.mp4>

Reading

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

```

```

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",
            MiscLib.ColorCyan, y, MiscLib.ColorReset)
116    }
117    {
118        dbr := database["alice"]
119        dbr.y = y
120        database["alice"] = dbr
121    }
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 calculation
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,
157     MiscLib.ColorYellow, MiscLib.ColorReset)
158
159 // -----
160 // Response 2 - Success/Fail message from server back to client
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",
169             MiscLib.ColorGreen, MiscLib.ColorReset)
170         fmt.Printf(`%sResponse: {"status":"success",
171             "msg":"'alice' is logged in"}%s`+"\n",
172             MiscLib.ColorCyan, MiscLib.ColorReset)
173     } else {
174         fmt.Printf("%sNope nope nope%s\n", MiscLib.ColorRed, MiscLib.ColorReset)
175         fmt.Printf(`%sResponse: {"status":"error",
176             "msg":"'alice' is not a valid user"}%s`+"\n",
177             MiscLib.ColorRed, MiscLib.ColorReset)
178     }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```
180 func genRan(m int) int64 {  
181     return int64(rand.Intn(m))  
182 }
```

Reunsts of 2 runs:

```
+==>  
+==> go run sip01.go  
Setup Complete: v=13136  
"Save 'v' for user "alice"  
/api/register-user, send-data={"user":"alice","v":13136}  
Response: {"status":"success", "username":"alice", "msg":"is registered."}  
random genrated: 227  
Send To Bob : x=86161  
/api/login, send-data={"username":"alice","x":86161}  
Server looks up in the database 'v' for "alice", v=13136  
Response: {"status":"success", "e":621}  
y=1008  
/api/login-pt1, send-data: {"username":"alice","y":1008}  
response: {"status":"success","y":1008}  
Server looks up in the database 'v','e','y' for "alice", v=13136  
z=86161  
/api/login-pt2, send-data: {"username":"alice"}  
Authoized! Yea, 'alice' is a valid user  
Response: {"status":"success","msg":"'alice' is logged in"}  
+==>  
+==>  
+==>  
+==>  
+==> go run sip01.go  
Setup Complete: v=13136  
"Save 'v' for user "alice"  
/api/register-user, send-data={"user":"alice","v":13136}  
Response: {"status":"success", "username":"alice", "msg":"is registered."}  
random genrated: 954  
Send To Bob : x=26648  
/api/login, send-data={"username":"alice","x":26648}  
Server looks up in the database 'v' for "alice", v=13136  
Response: {"status":"success", "e":874}  
y=984  
/api/login-pt1, send-data: {"username":"alice","y":984}  
response: {"status":"success","y":984}  
Server looks up in the database 'v','e','y' for "alice", v=13136  
z=26648  
/api/login-pt2, send-data: {"username":"alice"}  
Authoized! Yea, 'alice' is a valid user  
Response: {"status":"success","msg":"'alice' is logged in"}  
+==>
```

