Securid.prc

Name: SecurID
Creator: SoID
Type: appl
Size: 37867
Mod. Date: 5/16/96 11:14

Sn000000100011.prc

Name: SecurID Key File
Creator: ScID
Type: Auth
Size: 157
Mod. Date: 5/2/00 15:01

1) Delete passphrase

→ SecurID key file modified, size the same
exported to "SecurID key file_nopass.prc"

Passphrase dependent on HotSync User Name.

- The first time SecurID is run, modifies key file for no passphrase & useable only with given HotSync Name.

⇒ Need to step through initialization to see how things are being set-up.

MD5 ("Palm OS Emulator") = 66 76 36 79 fe 2a 59 8a fd 19 68 d8 8f 18 a5 f4

-hex -seperator -v -t systraps.txt -sp

~~Form $44E "Option Alerts"~~
~~"Ask, Second, Third Octal Byte"~~

FRM $3E8 Main Form

ALERTS

$44C Have not installed key file
$456 Malformed seed file
$460 Incorrect passphrase
$46A phassphrase required
$474 Authentication key locked
$47E Confirm delete passphrase
$488 Bad passphrase match (during change)
$492 You are in demo mode
$6A4 Debug Alert #1
$6AE Debug Alert #2

StartApplication → 1) check for key file
2) if yes, modify if first use?

MainFormHandleEvent → check for passphrase before drawing/calculating

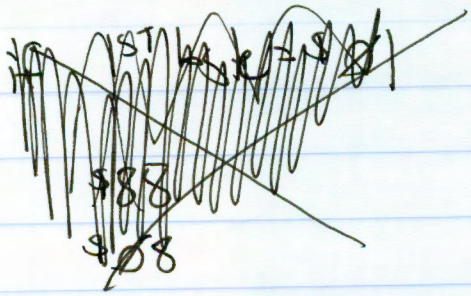
---

* Modify SECURID.PRC for Debuffer

	old		new	
Ø1A5C:	BNE L191	$00DE	NOP	$4E71
	MOVE.W #$44C...		TRAP #8	$4E48
			BRA L191	$6000
			4 bytes less	$00DA

⇒ SECURID-STARTAPP.PRC

Can load new key files into POSE w/o modifying memory map ← Good.



70CC6 MemMove (1C7F6, 230E6, $18 24 bytes)
dst src num

↓

initial string from sv00000010001l.prc

passphrase

00000000 3CD6B404 9F56 9A0A
8F3246AE 8739CAB2

70CD4 MemMove (1C7EE, 1C7F6, $8)
dst src num

↓

passphrase

00 00 00 00 00 00 00 00

MD5 (1B078, 706B8, 0)

↓ prep. string initialization vector

↓ data (null)

↓ length?

1C604 - new string to write to key file (24 bytes)
How is this generated?

→ Passphrase does not affect bitmode generation. Added measure of security to prevent unauthorized users from using SecurID app. If cracked, attacker could use - like stealing hardware token.

1) Passphrase
2) HotSync user name
3) Memory cell creation date, low word

65(A4) 0x1B278 *B340 = "45888" - creation date
0x1B1F8 "Palm OS Emulator" - HotSync user name
604(A4)
0x1B078 *6745 2301 EFCD AB89 - MD5 context
220(A4) 98BA DCFE 1032 5476

0x706B8 "" null (1st time) - passphrase?
0x1C604 *2C5C 0001 C60E 0007 passphrase string in reg file

1st time

@ 70D86 MD5 final: 'passphrase', username, creation date 2nd time
digest 0x1C7C6 md5 "" = *D41D 8CD9 8F00 B204 0x1C5D4
E980 0998 ECF8 427E
context 0x1B078 = *D98C 1DD4 ...
@ 70DD8 MD5 final: md5 hash of above, passphrase "" =
context 0x1B078 = *4EB2 AD59 02BE CDF3 5
395B F097 3F45 2758
digest 0x1C7C6 = *59AD B24E F3CD BE02 97F0 5B39 5827 453F

SSLeay source

MD5_Init
MD5_Update
MD5_Final
MD5_version

Problems

Time-based:

1) Replay attack - set time on Pilot forward to generate threads in advance. Token should invalidate app if time is changed.

- could generate patch to set time quickly
- Palm app sets notifies when time changes
- Hackmaster app - change time every second ... in background while SecurID app is running

Generate key schedule?

70D92- inpA = 0x1C7C6 = rand 1 hash - first 8 bytes
outpA = 0x1C746 = $ 18DC 30C8 8800 8147
A844 2034 C9C8 92C0

70DA4- inpA = 0x1C7CE = rand 1 hash - second 8 bytes
outpA = 0x1C6C6 = $ 3C48 EC30 4908 CB46
2000 84CC 4B08 8B41

70DE0- inpA = 0x1C7C6 = rand 2 hash - first 8 bytes
outpA = 0x1C646 = $ A0B0 1808 CF89 0F4D
3CFC 60F4 024A 4C8B

L445: des_ecb_encrypt? 2 rounds of 3DES decrypt

70E52: (1C7FE, 1C7FE, 1C646, 0) ks3
70E60: (1C7FE, 1C7FE, 1C6C6, 1) ks2
70E6C: (1C7FE, 1C7FE, 1C746, 0) ks1

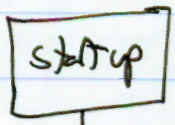
70E80: (1C806, 1C806, 1C646, 0) ks3
70E8E: (1C806, 1C806, 1C6C6, 1) ks2
70E9A: (1C806, 1C806, 1C746, 0) ks1

outpA: 64-bit unique ID for server ID algorithm?
$ B666 FE8E 1470 33D4 0808 0808 0808 0808
0x1C7FE 0x1C806

inpA: $ 3CD6 B404 9F56 9A00 8F52 46AE 8739 CAB2
0x1C7FE 0x1C806

From ORIGINAL KEY FILE!!

Process of encryption/personalization of secret stores in keyfile



↓ if 1st time

MD5 ("")

MD5 (digest, "")

Generate keyschedules (3)

3DES Decrypt 16-bytes in .PRC key file

outcome: B666 FE85 1470 33D4 0808 0808 0808 0808

64-bit unique ID for algorithm

constant?

Brute-forceable - lower 2 bytes on Palm desktop?

Personalize to specific user Palm

known

MD5 (passphrase, Hotsync user ID, mem. card creation date)

MD5 (digest, passphrase)

Generate NEW keyschedules (3)

3DES Encrypt 16-bytes and write to .PRC keyfile

MD5 seed time: rand 1 - 0x1C5D4 = $424A 33F8 2EC1 CB07 8C9F 2BD8 B6CA DA54

passphrase = ""

hotsync = "PalmOS Emulator"

memcard = "45888"

rand 2 - 0x1C5D4 = $B617 0F3F ED35 0E7F 1680 CAEC 180E 3D33

round 2 3DES: 0x1C60C @ 70E9E

$EE45 E6PB 75FE 9B94
81A7 9BBC 0480 685B

written back into PRC key file!

Generate key schedule - round 2

70DD2 - input = 0x1C5D4 = round 1 hash - first 8 bytes
output = 0x1C554 = $F058 3844 0E0A 0007
bCC8 8854 4546 00C8

70DA4 - input = 0x1C5DC = round 1 hash - second 8 bytes
output = 0x1C4D4 = $5818 B860 CF49 8E47
AC44 C4F0 8F4A 82CD

70DE0 - input = 0x1C5D4 = round 2 hash - first 8 bytes
output = 0x1C454 = $ 7CEC CC30 8E82 C04E
F4P4 0C08 9D88 8B0A

plaintext from round 1 - 1st 8 bytes
70DFC: (1C60C, 1C60C, 1C554, 1) 3DES encrypt
70E08: (1C60C, 1C60C, 1C4D4, 0)
70E16: (1C60C, 1C60C, 1C454, 1)
plaintext from round 1 - 2nd 8 bytes
70E2C: (1C614, 1C614, 1C554, 1)
70E38: (1C614, 1C614, 1C4D4, 0)
70E9A: (1C614, 1C614, 1C454, 1)

How are 1st 8-bytes in key file generated?

→ 0x1C5FC = $7C5C 0001 C60E 0007

Before round 2, after round 1   L134

70EF4: des-set-key. input = 0x1C7AE = plaintext, unique 64-bit ID
output = 0x1B14A = $D884 E4AC 8C85 8C88
705C 7428 8C87 C648

generated w/ des-set-key routine
a round

used for pref. storage? 2 addresses, rotated

14:31	34307492
14:32	86217916
14:33	35438127
14:34	52117759

8-byte can be set to $8100 756C 2715 0007
$71EE 0001 C60E 0004
$7C5C 0001 C60E 0007

L353_main (pin, time, secret, 8, 2, 4, 0, 0)

- pin ↑ NULL
- 8 ↑ token code length
- 2 ↑ C
- 4 ↑ C
- 0 ↑ C — in4
- 0 ↑ unused

if (in4==0)
L384 (&data, subkey)
convert tokencode

---

t_e - even minutes → first 8-bytes of tokencode

t_{e+1} - odd minutes ⟶ second 8-bytes of tokencode

0x707D4 - ~~Beginning of system clock?~~

midnight

D3 = GMT since 1/1/1904 = $B534DBA7
= 3040140199

0x707F0: ADD $83DA4F80 to D3 = $39F2B27
= 957295399

Time since known date?

31557600 seconds/year
3029529600 seconds from 1/1/1904 → 1/1/2000
2629800 seconds/month

GMT: 91399 since 5/1/2000
4999 since 5/2/2000

→ subtract $7C25B080 = 2082844800 seconds
66 years, 12 hours

43200 seconds

may 3 time

⇒ GMT time from 1/1/1970, 12 noon

A3 — used for GMT conversion — result 1 — result 2

7080C: ($8, $2, $6C, $1C868, $1C864)

↓ D5, ↓ D6, ↓ D3, ↓ A2, ↓ A3

A∅ = GMT = $B534E3E8 from 1/1/1904 midnight

Add $65C1D180 = 1707200896 seconds

A∅ = $1AF6B568 = 452375912

subtract $9A3E2E80 = 2587766400

82 years, 12 hours

⇓

GMT from 1/1/1986, 12 noon

31557600 sec./year

L85

#1 0x70578: L508 ($1AF6B568, $1E) — prepare

(30 sec'd)

↑ time ↑ constant

D∅ D1

0x75BF6: L573 ( ) — DIVIDE!

⇒ D4 = $00E61710

⇒ D∅ = $0002

#2 0x70586: L508 ($10, $8) — D5

#3 0x70590 = L512 ($2, $2) — D∅ D6

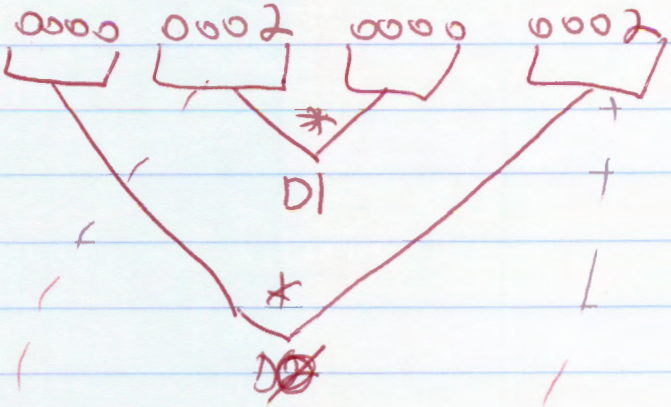
#4 0x7059C L508 (mangled time D4, result of #3 $4)

#5 L512 (result of #4,) = E6171C → (A2)

#6 L508 (D4 - result of #5, result of #3 $2 D6) = ∅ → (A3)

Multiply 2 32-bit numbers!

LSW: SP = xxxx xxxx



D1 + D0 = 0

swap MS high word of D0

*

D1 = 4

⇒ D0 = 4

$12345678 * $ABCDEF01 = $55065E78

$1234 * $EF01 = $10FE9E34 = D0

$5678 * $ABCD = $3A076618 = D1

D0 = D0 + D1 = $4B06044C

Swap = $044C4B06

$5678 * $EF01 = $50BA5E78

+ $044C0000

$55065E78

: MODULUS %.

LSB - calls from 18 locations

$390f2B27 / $3C = $F373EB
D0 D1

$F373EB * $3C = $390f2B14 round down
D0 D1

subtract

390f2B27 - 390f2B14 = $13 → return
D1 D0 D1

19d

mod time with 60sec. to see if new minute - time to recalculate

---

1AFb ABO7 - GMT 6MT/30sec

-38
L85 rslt → (A6) 0x1C846 = $E616C4
-$26

L513 rslt → $390f28F7 % 60sec. → $3B (59)

→ -42(A6) = 0x1C842
→ $2A(A6)

0x709B4 L353 (0x1C85A, $E616C4, 0x1B141, 8, 2, 4, 0, 0)
data 6MT from VM86 secret keycode length?

"00000000" or 1C86E =

→ keycode output!

→ enter PIN input
(if in "PIN" mode)

$B66 FE8E1470 3304
64-bit secret from key

GMT offset

default: $60: -5 hours

112 possible offsets

$88 = +14 hrs. maximum
$87 = +13.75 hrs.
$97 = +5.75
$94 = +5
$82 = +0.5
$80 = 0
$68 = -8
$50 = -12
$49 = -13.75 hrs.
$48 = -14 hrs. minimum

1/1/1970 → 1/1/1986 12 noon
16 years, 12 hours = 504921600 + 43200
= 504964800 seconds

Time calculated from 12/31/1985, 23:00

data time sect in1 in2 in3 in4 in5

L353 (0x1C85A, &E616F8, 0x1B141, 8, 2, 4, 0, 0)

D7 = time

0x1B141 = A3 = sect

D3 = in1 = 8

D6 = in2

D5 = in4

D4 = in5

sect constant

0x7231A - L391 (0x1B141, 4) 1st 4 bits - upper long

D0 = same

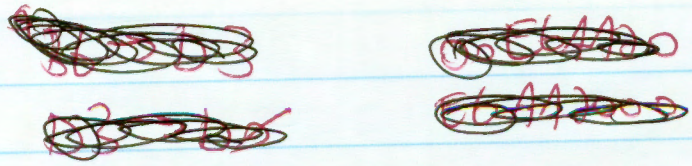
0x7237A - L391 (0x1B145, 4) 2nd 4 bits - lower long

D0 = same ⇒ moves to 0x1C7DE

0x7233C - L405 (0x1C7DE, 0x1B141, 8)

A1 A0 ↑ constant

Move A1 → A0, 8-bits



D0 = $10

D0/D3 = $2
D6 * D0 = $4

(16/in1) * in2 = $4

E61760 time/$4 = $398508 150>9784

$398508 * $4 = E61760

round down

in3 $4
4, 0

0x7836C: rounded down time set at
L359(0xE61760, 0x1B141, 0x1C7D6,)

0x1C7D6 = data = 5019 5699 7878 3816

L389(50195699, 0x1C7E6)
L389(78783816, 0x1C7EA)

data = 0x1C7E6

L396 (0x1C7E6, 0x1C7EE, $10)
dst aSpA 16-bits

↓

aSpA=16-bit blende=5019569978283876

33094467013624O3

L359 ~~Expand~~ time

A3 = 0x1C784
D3 = 0 = in4
D4 = secret
D5 = 4 = in3

L388 (random time, 0x1C784)
D0 A0

D0 = 00E6181C
D1 = E6181C00

D1 = D0 + D1 = E6181C1C
(A0) = $E6181C1C E6181C1C = 0x1C784

More secret → 0x1C77C

0x724A4 - L360 (1C784, 1C77C, 4, 0)
expanded time select in3 in4
D6 D5 D3 D4

L353 → L359 → L360

0x724DC - L359 (1C784, 1C77C)
A1

inpA: expanded time
outA

var1 = -8(A6)
var2 = -4(A6)
var3 = -24(A6)
var4 = -12(A6)
var5 = -10(A6)

A6 = 0x1C722

var6 = -16(A6)
var7 = -20(A6)
var8 = -18(A6)
var15 = -22(A6)

var9 = -28(A6)
var10 = -32(A6)
var11 = -36(A6)
var12 = -40(A6)
var13 = -42(A6)
var14 = -44(A6)

alleasoftw

↓

L374 (0x1C784)

expanded five

Second 4 bytes

A0 = 1C784 = E6158484 E6158484

D1 = E6158484 ↩

D0 = D1 + D1 = CC2B0908

D2 = D0 OR 0x1 = CC2B0909 → 4(A0)

first 4 bytes same

4th time - A0 = 1C784 = 30AC2427 30AC2427

D1 = 30AC2427 + 30AC2427

= 6158484E

Derive Subkey

0x1C784 (A2) = ~~[illegible]~~ F98A C424 E5FA 3R17

0x1C77C (A3) = ~~[illegible]~~

After 64 steps, A3 returns to original secret

$666 FE8E 1470 3304

D1 = F98A C424 ^ $666 FE8E

⇒ 4FEC3AAA

L573 SimpleTransform1 0x725B4

0x1C784 inpA: 79CC 66C1 8321 78D0

↓

D0 = 83

D1 = $64 = 100

D1 = $64 - $29 = $3B

A: 83CC 66C1 3B21 78D0

Simple Transform 2 L368

temp1 = 0x07 00000111 → 10000011

-1

= $82

temp1 = 0x82 10000010 → 01000001

-1

= $40

*9?

---

temp1 = 0x27 00100111 → 10010011

-1

= $92

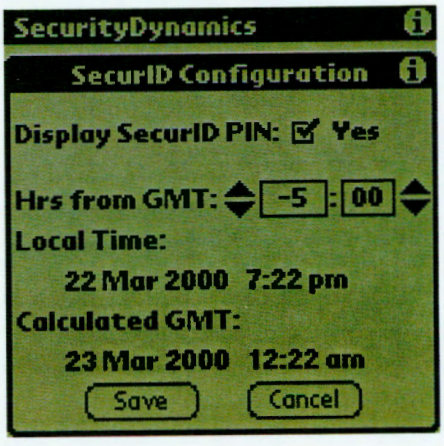
temp1 = 0x92 10010010 → 01001001

-1

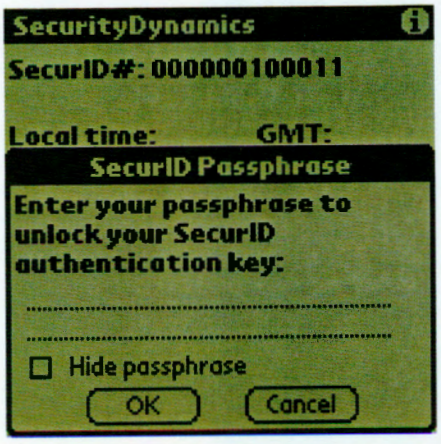
= $48

$95: 10010101 → 11001010 = $CA

$358 ManFam



$578

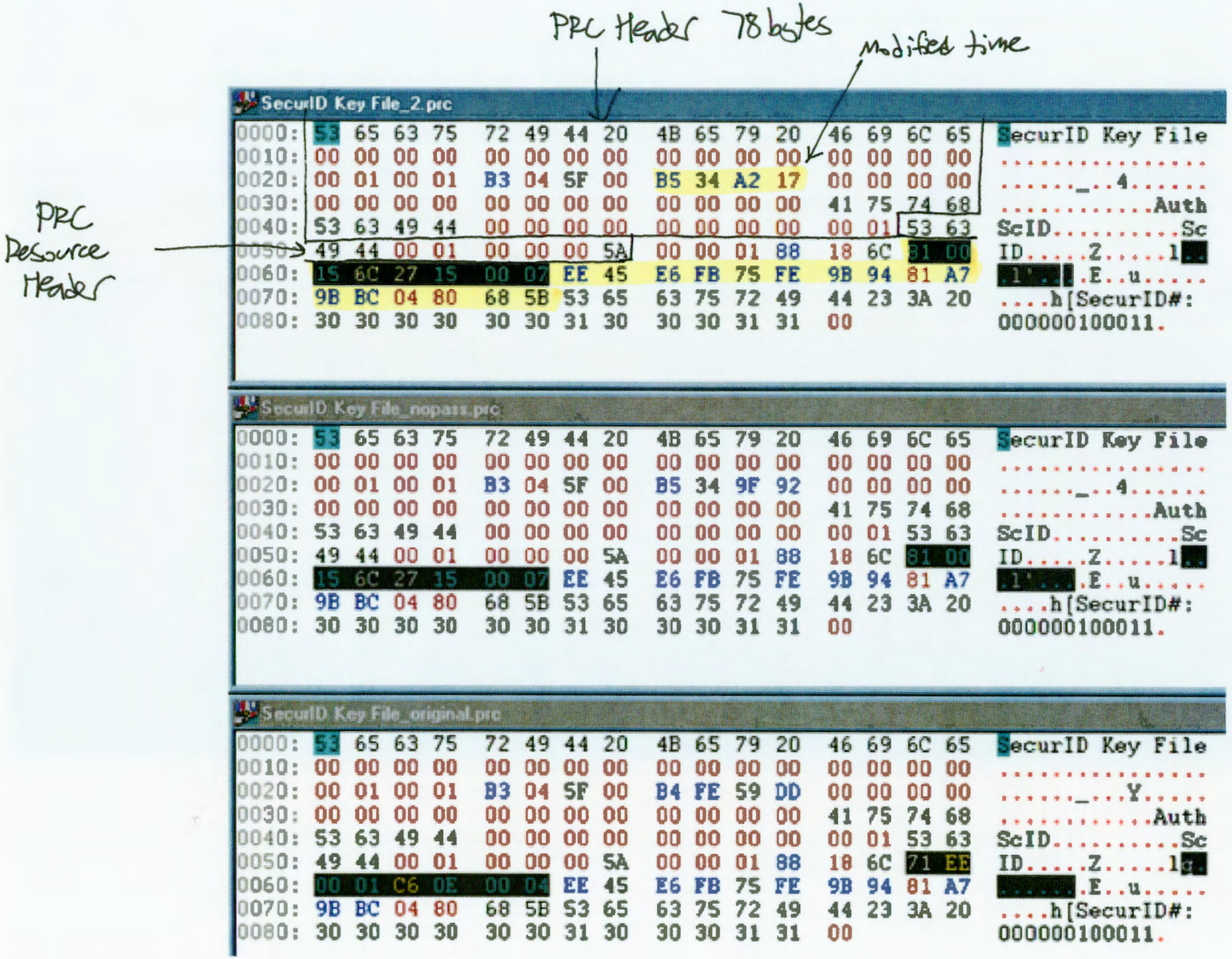


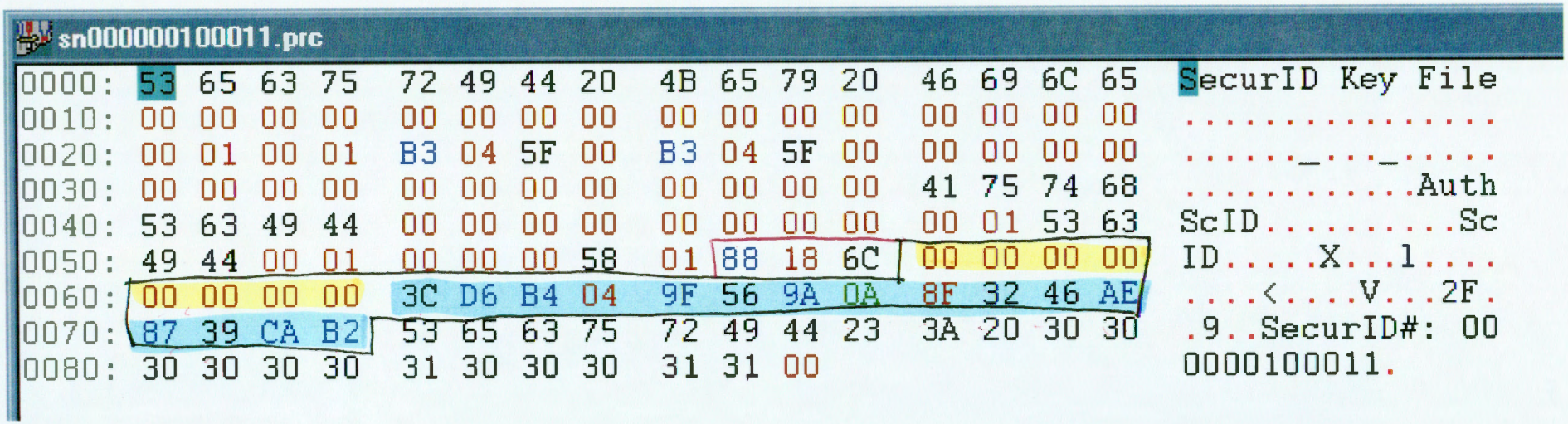
$480

$640

$514

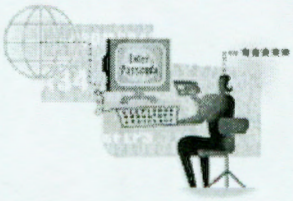
5.2.00


PRC Header 78 bytes
modified time
PRC Resource Header
SecurID Key File_2.prc
0000: 53 65 63 75 72 49 44 20 4B 65 79 20 46 69 6C 65 SecurID Key File
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0020: 00 01 00 01 B3 04 5F 00 B5 34 A2 17 00 00 00 00 ......_..4......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 41 75 74 68 ....Auth
0040: 53 63 49 44 00 00 00 00 00 00 00 00 00 01 53 63 ScID..........Sc
0050: 49 44 00 01 00 00 00 5A 00 00 01 88 18 6C 81 00 ID.....Z.....l..
0060: 15 6C 27 15 00 07 EE 45 E6 FB 75 FE 9B 94 81 A7 .l'....E..u.....
0070: 9B BC 04 80 68 5B 53 65 63 75 72 49 44 23 3A 20 ....h[SecurID#:
0080: 30 30 30 30 30 30 31 30 30 30 31 31 00 000000100011.
SecurID Key File_nopass.prc
0000: 53 65 63 75 72 49 44 20 4B 65 79 20 46 69 6C 65 SecurID Key File
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0020: 00 01 00 01 B3 04 5F 00 B5 34 9F 92 00 00 00 00 ......_..4......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 41 75 74 68 ....Auth
0040: 53 63 49 44 00 00 00 00 00 00 00 00 00 01 53 63 ScID..........Sc
0050: 49 44 00 01 00 00 00 5A 00 00 01 88 18 6C 81 00 ID.....Z.....l..
0060: 15 6C 27 15 00 07 EE 45 E6 FB 75 FE 9B 94 81 A7 .l'....E..u.....
0070: 9B BC 04 80 68 5B 53 65 63 75 72 49 44 23 3A 20 ....h[SecurID#:
0080: 30 30 30 30 30 30 31 30 30 30 31 31 00 000000100011.
SecurID Key File_original.prc
0000: 53 65 63 75 72 49 44 20 4B 65 79 20 46 69 6C 65 SecurID Key File
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0020: 00 01 00 01 B3 04 5F 00 B4 FE 59 DD 00 00 00 00 ......_...Y.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 41 75 74 68 ....Auth
0040: 53 63 49 44 00 00 00 00 00 00 00 00 00 01 53 63 ScID..........Sc
0050: 49 44 00 01 00 00 00 5A 00 00 01 88 18 6C 71 EE ID.....Z.....lq.
0060: 00 01 C6 0E 00 04 EE 45 E6 FB 75 FE 9B 94 81 A7 ....E..u.....
0070: 9B BC 04 80 68 5B 53 65 63 75 72 49 44 23 3A 20 ....h[SecurID#:
0080: 30 30 30 30 30 30 31 30 30 30 31 31 00 000000100011.


sn000000100011.prc
0000: 53 65 63 75 72 49 44 20 4B 65 79 20 46 69 6C 65 SecurID Key File
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0020: 00 01 00 01 B3 04 5F 00 B3 04 5F 00 00 00 00 00 ......_...._.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 41 75 74 68 ....Auth
0040: 53 63 49 44 00 00 00 00 00 00 00 00 00 01 53 63 ScID..........Sc
0050: 49 44 00 01 00 00 00 58 01 88 18 6C 00 00 00 00 ID.....X...l....
0060: 00 00 00 00 3C D6 B4 04 9F 56 9A 0A 8F 32 46 AE ....<....V...2F.
0070: 87 39 CA B2 53 65 63 75 72 49 44 23 3A 20 30 30 .9..SecurID#: 00
0080: 30 30 30 30 31 30 30 30 31 31 00 0000100011.

RSA SecurID for the Palm Computing Platform

Technical Specs



Current Shipping Version:	RSA SecurID for the Palm Computing Platform® v1.0
Desktop Requirements:	Windows 95 Windows NT Workstation 4
Required Components:	RSA SecurID for the Palm Computing Platform (1 per user) *Palm Device Requirements*: • PalmPilot™ Professional or Palm III™ organizer • Palm OS v2.0, v2.1, or v3.0 • 40K of available memory RSA ACE/Server v3.3.1 or v4.0 RSA SecurID for Palm Administration Program Available serial port for Palm docking cradle
Pricing and Availability:	The RSA SecurID for the Palm Computing Platform® is available through RSA Security sales channels.

• Download a demo of the RSA SecurID for the Palm Computing Platform ... Click Now!

RSA SecurID Authenticators:

• RSA SecurID Hardware Token
• RSA SecurID Software Token
• RSA SecurID 1100 Smart Card
• RSA SecurID for the Palm Computing Platform

RSA SecurID for the Palm Computing Platform

• Free RSA SecurID Token Trial [Downloads]
• RSA Conference 2000 Expands to Europe [Conference]