To —— let's discuss

Date: Jan 10, 1979

TO: John Couch

From: John Arkley JOA

Subject: Software Protection for Applesoft/DOS OEM Vendors

In reviewing Randy Wigginton's memo, dated Dec 13, 1979, I have noted the following suggestions have been offered by Randy for use by OEM vendors.

1. "programs in memory" can be "protected" in memory by:
   a. Only allow programs to run in the machine if the Auto-Start Rom is the only monitor in a machine.
   b. Put part of the "program" in an area of memory that is stepped on by the reset function of either monitor.

2. "programs on diskette" can be "protected" by:

   a. Modify the "core routines" (and therefore the "formatter") to change the data that identifies where soft-sectors are on the diskette.
   b. Modify the "RWTS routines" to support a "hidden" directory track to evade the function of the FID copy program.
   c. Delete a track or a sector per track from the disk to defeat the standard copy program.

Item 1a can only be done by having intimate knowelege of and the ability to reassemble specialized or patched versions of DOS, ie DOS LISTING & SOURCE.

Item 1b can only be done for the small subset of programs that are written all in Assembly and never have to reload programs after initial boot, and certianly is not a relevant suggestions for a APPLESOFT ONLY software package with multiple modules (the typical case!).

All of the suggestions under 2 require very high degree of knowelege about the most complex aspect of our Disk Controller and its READ/WRITE "core" routines and the things that are built directly on these routines. The source listings for this code have never been made easily available and the time and skill required to modify these things is very high.

To date a relatively small handful of vendors, Software Arts, Personal Software, Muse, High Technology, have figured out most of these methods FOR APPLE without much assistance, accept the early copies of the RWTS routines and the "formatter", which are now out of date and known to contain bugs.

Since Dec 1, I have sent out 10 sets of my "Locked Run Only Dos". I have had 7 requests for "higher" protection than the above. I have sent out 1 copy of RWTS and had 8 requests for DOS Source listings. now available for

2 $5000

In summary, only a very small subset of the OEM vendors have the skills and information to be able to apply any of Randy's suggestions. Even Software Arts, who has the skills and has done it before, wants the listings for the 16-Sector DOS RWTS and Core routines so that they don't have to waste their time 'disassembling' the meaning of the object code so such modifications can be made for DOS 3.3 security.

Unless Apple makes at least partial source listings for RWTS, Core and DOS 3.X available to vendors it is not likely most of them would be able to change there diskettes and programs to be reasonably secure. Even this would require directions on how and where to make the necessary changes and explainations of legal sets of choices that can be made without fear of messing other things up.

I could put together a list of "how to patch the DOS" instructions that would allow vendors to create diskettes that are not "DOS 3.X" readable or COPYable using the suggestions Randy has made, but there would be little reason, if this were done, not to just mark up RWTS, and Core source listing and send that to vendors who need it. (The suggestions Randy has made about diskette protection are the basis for our own SSAFE methods which we intend to use for apple software, although our methods go much further than those suggestions.)

If you approve of the idea of a Tech Note on "Patching the DOS, RWTS, & Core Routines" I will put this together and submit it for technical review by Randy and Policy review by you and marketing.

OK - I need to review
JL

CC: Mike Kane
: Dennis Reiger

To: John Couch

From: John Arkley

Date: July 11, 1979

Subject: Run ONLY protection method for Applesoft II-DOS 3.2

A facility that has never been published by system software, exists within DOS 3.2. This facility is related to APPLESOFT II's RUN ONLY mode, and provides a means of marking APPLESOFT programs as 'RUN ONLY' in the disk directory. This new file type shows in a catalog display as a normal Applesoft file, but it will return a 'FILE TYPE MISMATCH ERROR' if any attempt to LOAD, RUN or SAVE the file is made using normal DOS 3.2. I have created a RUN ONLY DOS 3.2.1 by patching 4 instructions and changing the DOS command vector table. This results in a DOS that will not run normal APPLESOFT II programs and all the DOS commands that are not needed for a running system, all attempt to perform a RUN command if they are used.

The result is a environment that satifies a constant request for some way to protect programs that are being sold for business purposes, against casual modifications by the purchaser and thus creating maintainence problems for the vendors. This put a mimimal hurdle between an typical 'user' who is not an APPLE II expert and the software. This is NOT a 'protection scheme' but simply a means of LOCKING software and forcing those who want to modify a package for their own use to follow the instructions that could say 'we will sell you an UNLOCKED disk and remove you from our support list'.

The disadvantage of not doing something at this level is that every vendor who is not working thru APPLE is already selling a modified DOS with some protection scheme of his own and the resulting MESS is going to proliferate as more vendors do this on their own. If we set up a pair of diskettes that will allow vendors to created 'LOCKED' masters and sell this as a service, thru Technical Support, we would standardize this 'LOCKING' scheme and regain some control over what is happening to DOS in the field. ~~complete~~ Doing This is

I think inaction on our part is ~~[illegible]~~ unnecessary ~~and~~ ~~simple enough to put into~~ effect that we should do this. This approach could be taken for all the about to be released business software without changing any of the existing APPLESOFT programs.

To: JOHN COUCH
From: John Arkley

Greg FYI

Date: July 20, 1979

Subject: Levels of Software Protection

There are now in existance four levels of 'protection' that could be used by APPLE to provide various degrees of security for software on the APPLE II.

Protection Method	Level of Security	Effort to Implement
RUN ONLY 'LOCKED' DOS 3.2.1	Prevents access by the non-assembler programmer, and even then is difficult for inexperienced APPLE user. User can not LOAD,SAVE, or LIST APPLESOFT programs even using a standard DOS instead of the RUN ONLY DOS.	11 bytes of patches in DOS 3.2.1 to make it a RUN ONLY DOS. I have already done this.
RUN ONLY 'LOCKED' UNDUPLICATABLE DOS 3.2.1	This would add to the above the inability to easily copy the diskette with our COPY program as it is now.	Delete all the address marks from track 3 of the diskette to cause COPY to get I/O errors Standard COPY followed by a TRACK 3 ZAPPER or an addition to COPY.
MODIFIED DISKETTE FORMAT UNCOPYABLE (ala MICROCHESS)	This makes it very difficult to copy even for the avid APPLE hobbist, assembler type. It is difficult to have any DATA files on this diskette.	Special complex MASTER DUPLICATION program is required for productn. User must exchange a bad diskette with us

NOME(in memory) and on diskette protection	This provides protection by essentially running in another interperter and system that is a derivative of APPLESOFT/DOS with some encryptation scheme Even this level will be broken by the hobbist in the end The main idea is to make the code non transportable back to APPLESOFT / DOS 3.2	Difficult to create copies and maintain programs since the 'sold' is like an object module. Return for the effort is questionable and having to maintain another system costs.

MacDonald

Inter Office Memo

Date: August 30, 1979

To: Distribution

From: Randy Wigginton RW

Subject: SSAFE - Software Security from Apples Friends and Enemies

In order for Apple to fully enter into the professional software business, we must have a method for making programs "secure". This much is intuitively obvious; the difficulty comes in defining "secure". As a means of defining security, the following levels are set forth to serve as a guideline;

Level 1. Totally secure. Absolutely no method of stealing the software. 100% effective.

Level 2. Almost totally secure. Piratable only by the most dedicated enthusiast. 99.8% effective.

Level 3. Very Secure. Breakable by hardware hacks with a respectable amount of effort to the point of being able to examine programs. 97% effective.

Level 4. Fairly secure. Breakable by software types with a sophisticated knowledge of the Apple. 94% effective.

Level 5. Not very secure. Your minimum "bare bones" protection scheme, similar to the Microchess cassette tape they did. 80% effective.

Note that the ideal, level 1, is achievable only through disallowing any access of any kind to the software and the computer. Not very practical in our circumstances.

The next best, level 2, is achievable through sophisticated hardware schemes. I don't believe this is what we need or want.

The next two levels, 3 and 4, are the ones we should aim for. Here is where the questions start arising:

Do we want any form of hardware modification?

How much effort and manpower do we wish to put out, and required to break it?

What is the criteria for a successful protection scheme? (i.e., should Steve Wozniak, Dick Huston, and Andy Hertzfeld each take over 1 hour to break it?)

What do software houses want in the way of security?

What kind of programs are to be protected? (Integer Basic, Apple II, Pascal, Basic III, Assembler)

For Basic programs, should we "Semi-compile" them down?

Is any type of system configuration required?

What are users willing to pay for protected programs?

What are software houses willing to pay to protect their programs?

Any and all inputs would be appreciated in a timely fashion.

Distribution:	Executive Staff
	Engineering Staff
	Software Staff
	System Software
	Phil Roybal
	Dennis Rieger
	Will Houde
	Guil Banks
	Taylor Pohlman
	Greg Smith
	Cliff Huston
	Steve Wozniak

Whitney

Inter Office Memo

Date: August 30, 1979

To: Distribution

From: To Randy Wigginton RW from Whitney

Subject: SSAFE - Software Security from Apples Friends and Enemies

In order for Apple to fully enter into the professional software business, we must have a method for making programs "secure". This much is intuitively obvious; the difficulty comes in defining "secure". As a means of defining security, the following levels are set forth to serve as a guideline;

Level 1. Totally secure. Absolutely no method of stealing the software. 100% effective.

Level 2. Almost totally secure. Piratable only by the most dedicated enthusiast. 99.8% effective.

Level 3. Very Secure. Breakable by hardware hacks with a respectable amount of effort to the point of being able to examine programs. 97% effective.

Level 4. Fairly secure. Breakable by software types with a sophisticated knowledge of the Apple. 94% effective.

Level 5. Not very secure. Your minimum "bare bones" protection scheme, similar to the Microchess cassette tape they did. 80% effective.

Note that the ideal, level 1, is achievable only through disallowing any access of any kind to the software and the computer. Not very practical in our circumstances.

The next best, level 2, is achievable through sophisticated hardware schemes. I don't believe this is what we need or want.

The next two levels, 3 and 4, are the ones we should aim for. Here is where the questions start arising:

Do we want any form of hardware modification? Not on Apple II, Sara, Lisa ok

How much effort and manpower do we wish to put out and required to break it? Randy for 4 mo.

What is the criteria for a successful protection scheme? (i.e., should Steve Wozniak, Dick Huston, and Andy Hertzfeld each take over 1 hour to break it?) Should take 2-3 hours

What do software houses want in the way of security?
All would agree level 5 better than nothing - Level 4 is OK.

What ~~kind~~ of programs are primarily to be protected? (Integer Basic, Apple II, [✓ Pascal], [✓ Basic III], Assembler)

For Basic programs, should we "Semi-compile" them down? NO

Is any type of system configuration required? NO

What are users willing to pay for protected programs? depends on program

What are software houses willing to pay to protect their programs? a lot!!

Any and all inputs would be appreciated in a timely fashion.

Distribution: Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Rieger
Will Houde
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

SOFTWARE PROJECT AUTHORIZATION

PROJECT NAME	SSAFE	PROJECT #	E-78
PROJECT LEADER	Randy Wigginton	DATE	September 5, 1979
OTHER PERSONNEL		SIGN OFF: SECTION/MGR	[illegible]
		SOFTWARE VICE-PRES.	[illegible]
		ENGINEERING VICE-PRES.	[illegible]
		NEW PRODUCT DEV.	[illegible]

PROJECT OBJECTIVES

(Purpose and scope of work, desired specifications, critical areas, relationship to other developments, etc.)

Purpose of project is to protect any desired piece of software. Specifications will be written by project leader, subject to approval. Security methods will be adaptable to SARA

OPERATING ENVIRONMENT

(Cassette, diskette, memory requirements, etc.)

Disk will be required. Other environmental requirements are unknown.

MANUAL REQUIREMENTS

Shouldn't be any.

EQUIPMENT/SPACE REQUIREMENTS

Nothing special.

PROJECT COST ESTIMATE

A. People:
1. Software Person-months 4
2. Manual/PUBS-Person-months -0-
3. New Product Review-Alpha & Beta Testing 1 week

over..........

PROJECT COST ESTIMATE (continued)

B. Project Material (Detail)

None

C. Consulting, Data Processing, Other

Probably None

Project Schedule

Milestone	Original Date	Last Month	Current Plan
Engineering Investigation Report	10/10/79		
Engineering External Reference Spec	TBS		
NPR Prelim. Review Report	TBS		
NPR Test Plan	TBS		
PUBS. Begin Manual Design	TBS		
Engineering Internal Spec	TBS		
Engineering Coding Complete	TBS		
PUBS. Release - NPR Draft	TBS		
Engineering-Product to NPR (Alpha)	TBS		
NPR Product Testing Complete	TBS		
Marketing - Product Marketing Plan	TBS		
Beta Test Complete	TBS		
ECO To Production	TBS		

Other: (Be Specific)

MacDonald

Inter Office Memo

Date: September 18, 1979

To: Distribution

From: Randy Wigginton R.W.

Subject: SSAFE

There will be a meeting Monday, September 24, at 4:00 p.m. in the Engineering Conference Room to discuss problems of software security and the goals of the SSAFE project. On Friday, September 21, I will send out a memo stating what I believe to be reasonable objectives for the project, including draw backs, potentials, and possible time frames.

Please return all feed back to me by Thursday, September 20, regarding the SSAFE project.

Distribution: Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Rieger
Will Houde
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

MacDonald

te: September 21, 1979

To: Distribution

From: Randy Wigginton R.W.

Subject: SSAFE

The following memorandum will be the topic of discussion at the meeting on Monday September 24, 1979 in the Engineering Conference Room.

When speaking of "protecting software", one usually means both protecting software from competitors, and protecting software from unauthorized use and copying. This is what SSAFE is going to attempt to do. Other types of protection, for example from theft, destruction, obsolescense, etc. are not included in this project. Another type of protection that is extremely valuable but also not covered under this project is that of data protection from unauthorized perusers and carousers.

COSTS AND BENEFITS OF SSAFE

The only cost to Apple is that of initial programming effort and ocumentation, in addition to any extra time in the production stage (DYSAN copying), which I expect to be minimal. The benefits, which are many-fold, are:
 * increased sales of software (since I can no longer copy my friend's diskette)
 * Elimination of user modified programs (my copy of XYZ no longer works and all I changed was...)

 * Encouragement to professional software houses to write programs that can be sold without fear of piracy. This may even result in increased system sales due to extra software available.

 * Better estimates of how many people are using any given program, since we will know exactly how many copies are in the field

COMMENTS ON FEEDBACK

The comments returned on my memo of August 30 were sparse but useful.Here is a summary of what the comments were, in addition to my comments.

1. Software houses are willing to pay 5-10% of their income on a program to have it protected.
   Note that this fee could also include licensing fees for use of Apple's S, etc.

2. There should be no form of hardware modification necessary.
   Although this is what Radio Shack is doing, I agree.

3. The types of programs to be protected consist of Applesoft II, ① PASCAL, Assembler, and Basic III.

4. No specific system configuration should be required.
   This is fine, except that programs can be made slightly more secure by requiring that the user have an auto-start monitor ROM.

5. Users aren't willing to pay anything for protected programs.
   This was an amazingly misunderstood question. What I was asking was, "What price should this software that has been protected sell for, in order that users will still buy it instead of living without it?"

6. A feature that would be nice would be that if a program were either broken or copied by someone, there would be a way of proving in court they had deliberate intent to violate the copyright.
   This is going to be very difficult, but possible.

7. Diskettes should be serialized so that if someone starts copying it and giving it out, we can track down the original leak. — HIGH PRODUCTION COST?

Time Frames and Possibilities:

By October 31; A method for protecting Applesoft and Assembly language programs to slightly below level 4. Time for breakage by a software expert (Dick Huston, Andy Hertzfeld, Randy Wigginton) should be approximately 1 hour for inspection of the programs; many hours for copying the disk.

By Mid-January; A method for protecting Applesoft, PASCAL, Assembly and Basic III programs to a level 3 or slightly below.

It is still too early to really promise anything at this point, but the above should be fairly close.

Note that protecting PASCAL and Aseembly programs is both easier and more secure. In a Basic program, once the user breaks into examining memory, he can see the tokens, which can be fairly easily decoded, and thus stolen. In light of this I propose a Basic compiler for BASIC III programs (Applesoft programs could be converted to BASIC III first) to a near-assembly language level. Most probably, compiled programs could run in a 48K machine, even though they were developed on a language card system. This option needs to be investigated thoroughly.

Distribution:

Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Rieger
Will Houde
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

9/24/79

SSAFE

* Want to focus on A II protection mechanism.
  But protection scheme is also applicable to Sara & Lisa,
  and mechanism may be different.

* Must be able to backup protected diskettes

* Proposal (Randy's)   Diskette protection

$$\begin{tikzcd} \text{MASTER} \arrow[r] & \boxed{\text{SPECIAL MASTER} + \text{SPECIAL COPY}} \arrow[r] & \text{PRODUCT} \end{tikzcd}$$

Prepared by Apple
Specific to each vendor
License agreement

Program Protection

Pascal
Basic III
Assembler
Applesoft

[illegible]
11/28/78

APPLE ENGINEERING LAB PROJECT REPORT

PROJECT	SSAFE	MONTH	9-79
PROJECT NO.	E-78	PROJECT LEADER	Randy Wigginton [illegible]
		OTHER STAFF	

OBJECTIVE:

Purpose of project is to protect any desired piece of software. Security measures will hopefully be appicable to Sara.

MONTHLY PROGRESS/STATUS:

Formulation of ideas and gathering input.

OBJECTIVES FOR COMING MONTH:

Publishing of a project objectives.

CRITICAL DEPENDENCIES:

None

SCHEDULE

MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
Investigation Report	10/10/79		10/10/79

MacDonald

Inter Office Memo

Date: October 5, 1979

To: Distribution

From: Randy Wigginton [illegible]

Subject: SSAFE

The following memorandum outlines what I believe to be the objectives for the SSAFE project. There will be a meeting 3:00 p.m. Wednesday, October 10, in the Engineering Conference Room.

OBJECTIVES:

TRAPs RESET

1. For protection of programs while residing in memory it will be required any user running a protected program have only auto boot monitor R.O.M.s. Assembly and Pascal programs will have no further protection. For those who make some type of hardware modification in order to get past the autostart R.O.M.s, figuring out the Pascal and Assembly language programs should be protection enough. For Applesoft programs a primitive encryption scheme will be used, that won't be very difficult to break. (20 minutes by a software expert). If more protection than this is desired, it will require substantial effort on my part.

2. For protection of programs on diskettes:

   ASM ?

   Pascal and Applesoft II programs will be protected and unreadable. Data files and the directory will be unprotected. Pascal has a file transfer program, and we have one internally to transfer files on DOS 3.2 diskettes. These programs would be usable for backing up files, but would be useless on the protected programs themselves. Obviously the copy program would be useless, and any general software-only copy program would be impossible. Note that Woz's hardware-assisted copy program will also be defeated by scheme 'b' described below.

3. I propose two levels of protection for diskettes:

   a. The "bare-bones" package. A software house would send an unprotected diskette with the desired programs to be protected. We would then file the program, charge a fixed amount, then return a copying program for their diskette along with a "trap-door" master. Using the supplied copy program, they may duplicate the master, creating diskettes that are protected. Apple would then disclaim any responsibility for the program or the protection.

b. For programs that Apple wished to protect, we would have a special copy program, along with a 'production procedure" for duplicating the diskette. Diskettes would be unduplicatable via any general purpose copy program including Woz's hardare-assisted copy. Additonally, diskettes could be serialized, so that in case someone did make copies they would all have the same serial number.

Protection scheme (a) would be a level 4; breakable only by sophisticated software types. Protection scheme (b) would be levels breakable only via a "brute force" scheme - basically tracing through the boot procedure - a very, very long process.

<u>Time Schedules:</u>

Protection scheme (a) will be ready by mid November. Protection scheme (b) will require approximately 5 man days each of Al Hoffman's and Rick Auricchio's time for assistance, sometime in November, and could be ready before the end of the year.

Distributution: Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Reiger
Will Houde
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

MacDonald

Inter Office Memo

Date: October 10, 1979

To: Distribution

From: Randy Wigginton

Subject: SSAFE Meeting Change

The SSAFE Meeting scheduled for 3:00PM today has been CHANGED to tomorrow, October 11, at 3:00PM in the Engineering Conference Room.

Distribution:
Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Reiger
Will Houde
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

10/11/79

Sara

Diskettes applicable

Memory - use "reset disable feature"

Auto Start ROM problems (field feedback)

- loose: single step
  trace
  mult. + div.
- like: cursor controls

Language	Memory	Disk
A II		
ASM	auto boot ROM	
PASCAL	"	program files only
APPLESOFT	encryption	"
BASIC III		
INT. BASIC		
FORTRAN		
COBOL		

SARA

ASM

PASCAL

BASIC III

FORTRAN

COBOL

11/28/78

APPLE ENGINEERING LAB PROJECT REPORT

PROJECT	SSAFE	MONTH	October
PROJECT NO.	E-78	PROJECT LEADER	R. Wigginton [illegible]
		OTHER STAFF	

OBJECTIVE:

Purpose of project is to protect any desired piece of software. Security measures will hopefully be applicable to Sara.

MONTHLY PROGRESS/STATUS:

Publishing of project objectives; protection of scrubbing bubbles and microchess.

OBJECTIVES FOR COMING MONTH:

General protection scheme for programs, plus beginning work on protection scheme 'b' - the high security protection.

CRITICAL DEPENDENCIES:

None

SCHEDULE

MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
Investigation Report	10/10/79	10/10/79	Completed 10/5
General Protection Level 4	11/26/79		11/26/79

Inter Office Memo

Date: 1 November, 1979

To: Dennis Rieger

From: Joe Shelton

Subject: SSAFE

The following is a compilation of comments from people involved in the SSAFE project. What other information should we (I) be looking for?

The following open issues and schedules were formulated in discussion with Jack MacDonald.

1. Randy Wiggington is currently working on a protection scheme for "scrubbing bubbles". It is a one time "fix". A decision needs to be made regarding future protection for internal release software. Jack sees this as needing Marketing input and direction.

2. The protection scheme to be marketed as a product to software houses will be available for "testing" in mid-November.

3. The Proprietory protection scheme (to protect Apple products) has no current schedule for completion.

Taylor Pohlman had the following comments:

1. The OEM's will not commit resources to developing software if it can be readily copied. In order to entice OEMs to produce better quality software we must provide good protection schemes.

2. What the OEM's need and what they think they need aren't necessarily the same. We should provide them with what they think they need. They think they need totally protected software, what they actually need is high level protection. See below.

3. Whether or not the diskettes are actually copyable or not isn't most important. Someone will probably be able to break any protection scheme. By protecting the software to some level and then copywriting it, the OEM raises a flag that the software rights are important. Then anyone copying it can be prosecuted.

4. We should not release the (16 sector) Read/Write Track Sector

externally; and we should try to ensure that the lab is made aware of the need for security. Taylor feels the lab has leaks.

Randy Wiggington's comments:

1. Scrubbing Bubbles has been protected however there is a bug in the program so it won't boot on a basics diskette. The bug is being addressed by using a hardware logic tester.

2. Both Scrubbing Bubbles and Micro Chess use the same protection scheme. They will probably be the last programs to use that particular scheme because neither needs to write to the diskette.

Question - Can we reasonably do like VisiCalc? Even on a one drive system they boot with the program diskette and then use a data diskette. JS

3. Future diskettes will need a different type of protection because parts of the diskette will have to be protected and parts can't be because they will have to be written on.

4. There probably isn't a need to have an internal release protection scheme in addition to the OEM and Proprietory protections schemes.

5. The OEM protection version will be ready to test in about two weeks. There are potentially 255 versions of OEM protection.

6. There are approximately 30 different protection schemes for protecting proprietory software. The current version of the protection allows Woz's hardware assisted procedure to copy the diskettes. Randy feels that if he has a version that Woz can't copy then it is as protected as possible.

Question - If only someone with Woz's expertise can copy software protected with the current scheme, might it be protection enough? See Taylor's comment #3. JS

7. The Proprietory scheme will include a copy program that will place hidden serial numbers in the code, allowing tracing the purchaser of any programs that actually get copied.

Wil Houde's comments:

1. He sees a need for the same two levels of protection - OEM and Proprietory.

2. The new diagnostic diskette will be protected combining Dick Huston and Guil Banks' procedures.

3. Wil can administer the OEM protection without additional resources.

4. A product encoded by Apple should not be able to be construed by the public as an endorsement by Apple.

5. OEM protection should have a price of at least "a few hundred dollars".

Note: Bill Atkinson has a procedure that might copy VisiCalc. Woz hasn't been able to.

Inter Office Memo

Date: 2 November, 1979

To: Dennis Rieger

From: Joe Shelton

Subject: SSAFE Assessment

The following is an accessment of the information relative to the SSAFE project.

OEM Protection

In order to encourage OEMs to produce more and better software, we should provide a protection scheme that will make their programs and diskettes secure from copying or "loading and saving". Currently many OEMs are unwilling to make a commitment to develop sophisticated software without an ability to protect it from being freely copied.

OEM software protection would be sold as a service to software houses to protect their software. We would take their diskette and protect it. We would provide the OEM with a copy of the diskette and a copy program.

The fee we would charge would be in the $500 to $1000 range to attract serious OEMs only. There is concern that any OEM using our protection scheme would not be able to use it as Apple's endorsement of his product. This scheme will be ready for testing in mid-November.

There are approximately 255 schemes (codes) that can be used for protection of OEM software.

Proprietory Protection

A Proprietory protection scheme will be used on any future products marketed by Apple. There are about 30 different schemes to accomplish this. The copy program will place hidden serial numbers in the code as a further deterrent and to allow tracking any copied software to the purchaser.

Scrubbing Bubbles and Micro Chess use a different protection scheme that protects against writing to the diskette. Any program that requires writing to the diskette cannot use this scheme.

There is no current completion schedule for the Proprietory scheme.

Wil Houde can handle the sales of this service through the service department without additional resources.

Other Issues

The 16 sector RWTS should not be released externally. The release of the 13 sector version makes protection harder.

Scrubbing Bubbles has been protected however the program has a bug so it won't boot on a basics diskette.

Inter Office Memo

Date: 6 November, 1979

To: Dennis Rieger

From: Joe Shelton JS

Subject: SSAFE - differences in security and levels of safety

Randy Wiggington is reticent to give out information unless asked exactly what is desired. I don't know all the questions!

OEM protection -

There are approximately 65,000 ways to protect OEM software. They will be able to be broken by a knowledgeable software specialist who can "read the nibbles". They won't be able to be "load and saved"; they will be chained within a diskette. They will be designed to only run with an auto-start ROM, however they can be made to run with the old ROM.

Proprietory protection -

There will be approximately 30 methods to protect this software. The methods used will be very complex (compared to OEM protection) and will take an expert with sophisticated hardware assistance to break the software.

These won't be able to be "loaded and saved" either. They are also chained within a diskette. Programs using this protection will also require an auto-start ROM.

Comparative safety -

The following is Randy's ranking of their relative safety, with 10 being completely protected and 0 being no protection.

```
0---------------------------5------------------8---------10
                           OEM               APPLE
```

11/16/79

Jack —

I broke Visicalc's scheme. Elapsed time = 1hr 15 mins

I would rate it slightly better than the Scrubbing bubbles & Microchess schemes, but not as good as our scheme (either for vendors or internal~~ly~~ programs).

Randy

SSAFE

* Does RWTS availability jeopardize security? Why?

* If AII+ AND w/AS card (or vice versa) and trace out to monitor on card, then change switch on card to old monitor, then hit RESET. Now in monitor!

11/29/79

Marketing + Eng.

- Concern regarding bringing product in-house to protect; unclear what Apple's ~~to~~ liability is.

- Hardware assistance in protection mechanism is acceptable if more secure.

Rev. 11/13/79

APPLE ENGINEERING LAB PROJECT REPORT

PROJECT SSAFE MONTH November, 1979

PROJECT NO. E-78 PROJECT LEADER Randy Wigginton RW [illegible]

OTHER STAFF

OBJECTIVES FOR CURRENT MONTH:

General protection scheme for programs, plus beginning work on protection scheme 'b' - the high security protection.

MONTHLY PROGRESS/STATUS:

Scrubbing Bubbles and Microchess have been released to Production, including verification program. Protection scheme is approximately 60-70% finished; will continue work in any case until Marketing decides otherwise.

OBJECTIVES FOR COMING MONTH:

To redefine project from a Marketing point of view and to re-evaluate objectives of project; i.e., what level of protection is necessary.

CRITICAL DEPENDENCIES:

Marketing must decide what they want; Engineering progress will proceed.

SCHEDULE

MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
Unknown; depent upon Marketing decisions			
Copy with serial numbers (Per John Couch)	12/10/79	--	12/10/79

Jack

Inter Office Memo

Date: December 11, 1979

To: Dennis Rieger, Joe Shelton, Jack MacDonald, John Couch

From: Randy Wigginton R.W

Subject: SSAFE

It has become apparent that the end result of SSAFE may not be what anyone really wants. Therefore, there will be a meeting Thursday, December 13th at 10:30 a.m. to 12:00 noon in the Executive Board Room to discuss the following questions:

1. Does Apple wish to form a protection service?

   The original idea of the SSAFE project would be that a software house would send us a diskette to be protected then we would return a master diskette and a copy program, from which the software house could create their own protected diskettes. However, this raises many possible problems; for example, the software house may be expecting a higher level of protection than they are actually receiving, and might hold Apple responsible if their program is pirated.

2. What kind of protection is desired and what are users willing to pay?

   The current SSAFE project does not protect memory at all; in order to do this, more hardware is going to be necessary (ala TRS-80 style). This weakness has already been exploited; an unprotected version of Scrubbing Bubbles has been obtained this way, with very little effort on the part of the thief.

CONCLUSIONS:

After careful consideration of what people seem to want, I propose that I finish with the current SSAFE project and we give it to software houses, with the recommendation it be used for programs with a retail price of $100.00 or less. Apple could use the same general scheme for all inexpensive programs.

As for protection of programs where a high level of protection is necessary, I recommend that SSAFE not deal with this area, but rather form another project that would require additional hardware to run protected programs.

cc: Executive Staff
Engineering Staff
Software Staff
System Software
Phil Roybal
Dennis Rieger
Will Doude
Guil Banks
Taylor Pohlman
Greg Smith
Cliff Huston
Steve Wozniak

Date: December 13, 1979

To: John Couch

From: Randy Wigginton

Re: Good (Apple) vs. Bad (Users)

When dealing with the question of program security on the Apple II, one must really face two quite distinct areas of difficulty:
1. Security of programs in memory, and
2. Security of programs on diskette.

The first question deals with preventing users from simply hitting the Reset key and saving the contents of memory to cassette tape or even diskette. The second is the problem of preventing a user from running the standard Apple COPY program on the protected diskette, or even using FID, the FIle Development program that allows individual file copying. This document shall propose various ideas that can be used to protect against undesired modification/inspection/duplication by users. We shall deal with the problem of memory protection first.

There are two basic methods for protecting programs in memory from user examination/modification. The first method is very simple; never allow the user control of the machine, through use of the Autostart Monitor ROM. Only allow programs to run if the user has an Autostart ROM, and nothing else, in the machine. In this case, be sure to toggle $C080/$C081, the Applesoft II Firmware card control locations, to make sure there isn't a standard monitor Rom on the card, thus allowing the user to flip the switch on the card and hit RESET.

The second, and best, method of protecting programs in memory is to put part of the program where it will be destroyed even if the user gains control of the machine. The best place that it is sure to be wiped out during a Reset operation is the text screen ($400-$7FF). However, if the program generates text output, then a special character output routine needs to be written which outputs to the second screen area($800-$8FF). Other good locations to use are the input buffer ($200), and zero page locations that are destroyed by the monitor during the reset operation ($31-$33, $3C-$3F, other miscellaneous places).

Protection of Diskettes:

There are two entities that need to be protected against: the standard Apple COPY program, and FID, the file copy program.

The easiest way to protect against both of these programs is to modify the core routines so that a standard DOS will not be able to read the diskette, and likewise the special DOS will not be able to read standard diskettes. If no backup capability is desired or necessary, this is a very simple, very effective method that is also quite easy to implement.

Another interesting but quite easy way to protect programs, but still keep them hidden from the CATALOG function and the FID program, is to modify the RWTS routine such that when a read is requested from track 17 (the directory track), and certain key values are in certain key locations, RWTS actually reads from an entirely different track which contains a "hidden" directory. This directory will be accessed only if a program knows what values to poke into which locations, and thus access other programs on the diskette. Note that the key locations should be reset to some neutral values as soon as possible after loading the new program to insure against the user somehow gaining control of the machine and examining the secret directory track, even though she would be trying to read track 17.

To protect against the standard COPY program, the easiest way is to simply "bomb" one of the tracks on the diskette by seeking the head to the desired track, then turning on the write head. Another simple method that is somewhat more effective is to modify the formatter such that one sector on each track is improperly formatted. This prevents a user from restarting the COPY program in the middle, for example, to copy only tracks 5-35.

If either of the above methods is used, the protector must take great care that the master bit map of used sectors reflects which sectors really shouldn't be used, either because they are formatted incorrectly or because they contain programs that the user directory does not know about.

12/13/79

SSAFE

- Probably will have an AP note on suggested protection mechanisms; this scheme is oriented towards garage shop vendors

- Probably don't want to bring software into Apple for protection
  - potential liability
  - discloses proprietary software to Apple

- Randy has only one proprietary scheme
  - DOS 3.3 12/25
  - Pascal Jan-mid
  - Sara ?

- level 1 - hand out to requestors via arkley
  - 2 - Apple proprietary (Randy's)
  - 3 - ~~[illegible]~~ super protection scheme

- Randy will write memo on level 1 schemes apple computer inc.

To: From:

Subject: Date: 12/20/79

Randy:

What needs to be done to your protection scheme in order for it to work under DOS 3.3?

Jack

Very little – about 2 days modification – only change is in core routines

10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

Rev. 11/13/79

APPLE ENGINEERING LAB PROJECT REPORT

PROJECT SSAFE MONTH December, 1979

PROJECT NO. E-78 PROJECT LEADER Randy Wigginton [illegible]

OTHER STAFF

OBJECTIVES FOR CURRENT MONTH:

To redefine project from a Marketing point of view and to re-evaluate objectives of project; i.e., what level of protection is necessary?

MONTHLY PROGRESS/STATUS:

Project objective was defined; ideas for protection were turned over to the technical support group. Copy program with serial numbers compelted.

OBJECTIVES FOR COMING MONTH:

Finish protection scheme for both Pascal and DOS 3.2 programs, and investigate hardware protection.

CRITICAL DEPENDENCIES:

May need help from Al Hoffman regarding Pascal protection.

SCHEDULE

MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
Copy with Serial Numbers	12/10	12/10	Complete
DOS 3.2 Protection	12/21	12/21	12/21
Pascal Protection		1/15	1/15

Comments added to Couch's copy

Jack,
Comments?
Joe

Date: 14 January, 1980

To: Distribution

From: Joe Shelton

Subject: SSAFE AND OEM SOFTWARE SECURITY PRODUCT PLAN

SSAFE AND OEM SOFTWARE SECURITY PRODUCT PLAN

1. PRODUCT DESCRIPTION

1.1 Overview

This product plan defines a scheme for software protection of proprietary products (SSAFE) and also a method for disseminating software protection information to OEMs. This dual level of protection will result in limiting the proliferation of pirated software and encourage OEMs to produce additional better quality software.

1.2 PROPRIETARY Protection (SSAFE)

The security provided by SSAFE will be used on all new products shipped after 1 February, 1980. It will be better than the security scheme on VisiCalc.

There are two criteria that this scheme should meet. The first is that there will be a large number of "codes" that can and will be changed periodically. This will prevent anyone from breaking one "code" and then having the key to all others. Each product could have its own "code".

Secondly, any product that is copied (short of returning the diskette to standard DOS) should produce a diskette with the same protection. This will substantially eliminate the proliferation of most copied software by eliminating the binary expansion effect (1 copy becomes 2 which become 4, 8, 16, etc.).

SSAFE has the capability to protect both diskettes and individual files. This will allow the protection of a complete diskette (as in the case of a game) or file and program protection (to allow writing to the diskette, e.g. Apple Writer).

Any changes to DOS or system software will be accomplished so as to have minimum impact on secured products already in the field.

1.3 OEM Protection

OEMs (that meet criteria yet to be established) will be provided with information on different protection schemes. Engineering is working on developing these schemes. This will allow the OEM to obtain a minimal level of protection through their implementation of the information provided. This information will not provide a high level of security because it probably won't completely protect memory. The relative level of safety will be made known to interested OEMs. There will be a large number of codes that will permit different protection for each product.

We will inform the OEMs that Apple will continue to be sensitive to these schemes in future DOS and systems software. Additionally, we will take steps (through licensing agreement or otherwise) to ensure that there will be no warranty, implied or otherwise.

2. PRODUCT CONTRIBUTIONS/RISKS

2.1 Business Objectives

There are three objectives that will be accomplished by using a proprietary protection scheme and providing protection information to OEMs.

The first will be to encourage OEMs to design and produce more and better software. A usable protection scheme will encourage OEMs (both those already programming Micros and others that might be interested) to produce additional software because their products will be protected from piracy.

The second objective is to limit the proliferation of pirated Apple proprietary software and thus sell more software. This is the same principle that applies to the OEMs, more people will buy our software because less will be available through piracy.

And third, as a result of the first two objectives, to increase Apple Corporate profit through the sales of additional systems (due to the addition of quality software) and proprietary software.

2.2 Market Contribution

The availability of software protection, both through OEMs and Apple proprietary products, will increase the amount of quality software products available in the market.

2.3 Business/Manufacturing Risks

2.3.1 Marketing/Support Risks

The only risks entailed are based upon future changes to systems software. SSAFE may not work with future systems software and the future installed base of protected products may not work with a new version of systems software.

3. PRODUCT CONFIGURATION

3.1 Software/Hardware Configuration

The proprietary scheme (SSAFE) will be usable on any 13 sector basic or 16 sector Pascal diskettes and will protect either complete diskettes or specific files on the diskette.

The OEM schemes capabilities have yet to be determined.

4. PRICING

The only potential charge to OEMs would be a nominal charge that would cover costs incurred by Apple.

5. PROFITIBILITY

There is no estimate of profitability directly attached to this product.

6. MERCHANDISING PLAN

6.1 Distribution Channels and Communications Plan

OEMs will be made aware of the availibility of this information through Software Engineering, Technical Support, the Hot Line, Marketing and any other departments that interface with OEMs.The information on protection procedures will be made available to interested and qualified OEMs through Technical Support (John Arkley).

6.2 Availability for New Products.

DOS 3.2	- 1 February, 1980
DOS 3.3	- 8 February, 1980
Pascal	- 29 February, 1980

7. SUPPORT

7.1 Support

Engineering support will be through Systems Software and Technical Support.

8. OPEN ISSUES

8.1 There is a need for a sophisticated software protection that will probably require hardware assistance. This level of protection will be necessary for both SARA and LISA software. A more sophisticated protection scheme would also be worthwhile for Apple II. A project needs to be opened immediately to address these needs.

RECEIVED FEB 6

✓ Bruce

✓ Randy

DISTRIBUTION

Carl Carlson
G. Carter
J. Couch
S. Jobs
A.C. Markkula
M. Scott
A. Sousan
J. Vennard
T. Whitney
K. Zerbe
P. Fry
A. Oppenheimer
J. Richardson
D. Bryson
T. Hawkins
D. Rieger
P. Roybal
P. Wyman
Mike Kane
J. MacDonald
K. Rothmuller

Date: 31 January, 1980

To: Distribution

From: Joe Shelton

Subject: SSAFE AND OEM SOFTWARE SECURITY PRODUCT PLAN

SSAFE AND OEM SOFTWARE SECURITY PRODUCT PLAN

1. PRODUCT DESCRIPTION

1.1 Overview

This product plan defines a scheme for software protection of proprietary products (SSAFE) and also a method for disseminating software protection information to OEMs. This dual level of protection will result in limiting the proliferation of pirated software and encourage OEMs to produce additional better quality software.

1.2 PROPRIETARY Protection (SSAFE)

There are two criteria that this scheme should meet. The first is that there will be a large number of "codes" that can and will be changed periodically. This will prevent anyone from breaking one "code" and then having the key to all others. Each product could have its own "code". It will be better than the security scheme on VisiCalc. — i.e., Alá Woz

Secondly, any product that is copied (short of returning the diskette to standard DOS) should produce a diskette with the same protection. This will substantially eliminate the proliferation of most copied software by eliminating the binary expansion effect (1 copy becomes 2 which become 4, 8, 16, etc.). ?

SSAFE has the capability to protect both diskettes and individual files. This will allow the protection of a complete diskette (as in the case of a game) or file and program protection (to allow writing to the diskette, e.g. Apple Writer).
Each version of SSAFE (DOS 3.2.1, 3.3, or PASCAL) is dependent on the underlying operation system. As the operating system is changed, SSAFE will have to be changed also.

One interesting note. The SSAFE DOS will have the "SAVE" DOS command missing. This will prevent users from saving protected programs. With this command removed, the Apple will potentially take larger programs in memory. — false

1.3 OEM Protection

OEMs (that meet criteria yet to be established) will be provided with information on different protection schemes. The Technical Support Group is working on developing these schemes. This will allow the OEM to obtain a minimal level of protection through their implementation of the information provided. This information will not provide a high level of security because it probably won't completely protect memory. The relative level of safety will be made known to interested OEMs.

We will inform the OEMs that Apple will continue to be sensitive to these schemes in future DOS and systems software. Additionally, we will take steps to ensure that there will be no warranty, implied or otherwise.

2. PRODUCT CONTRIBUTIONS/RISKS

2.1 Business Objectives

There are three objectives that will be accomplished by using a proprietary protection scheme and providing protection information to OEMs.

The first will be to encourage OEMs to design and produce more and better software. A usable protection scheme will encourage OEMs (both those already programming Micros and others that might be interested) to produce additional software because their products will be protected from piracy.

The second objective is to limit the proliferation of pirated Apple proprietary software and thus sell more software. This is the same principle that applies to the OEMs, more people will buy our software because less will be available through piracy.

And third, as a result of the first two objectives, to increase Apple Corporate profit through the sales of additional systems (due to the addition of quality software) and proprietary software.

2.2 Market Contribution

The availability of software protection, both through OEMs and Apple proprietary products, will increase the amount of quality software products available in the market.

2.3 Business/Manufacturing Risks

2.3.1 Marketing/Support Risks

SSAFE may not work with future systems software and the future installed base of protected products may not work with a new version of systems software.

In addition, support will be difficult for a number of reasons. With SSAFE protection, Hotline software changes or updates cannot be made. The diskettes must be physically returned to either the dealer or Apple. (This may be an advantage because it helps guarantee that no one changes the production software.

This also means that Apple must determine a way to handle replacement of diskettes that have to be either updated or replaced because the user can no longer make his own back-up copy.

A third problem is that, in essence, Apple will be sending out modified DOS.

This means that Apple will now have to support more than one DOS at a time. - No! This D.O.S. will require No support.

OEM's will need a "cookbook" to outline schemes and may require even more hand holding. ?

3. PRODUCT CONFIGURATION

3.1 Software/Hardware Configuration

The proprietary scheme (SSAFE) will be usable on any 16 sector DOS or Pascal diskettes and will protect either complete diskettes or specific files on the diskette.

An Auto-Start ROM is the only special hardware required to run SSAFE protected software.

The OEM schemes capabilities have yet to be determined. Systems Software (Randy Wiggin~~g~~ton) and OEM Support (John Arkley) are working on acceptable schemes.

4. PRICING

The only potential charge to OEMs would be a nominal charge that would cover costs incurred by Apple. ?

5. PROFITABILITY

There is no estimate of profitability directly attached to this product, but there will be the increased costs from additional engineering support. The purpose behind this project is to eliminate the proliferation of pirated software and thus increase the sale of Apple software; increase the interest of OEM's to write good application software; and thus, because of the increasing availability of quality software, increase the sales of systems.

6. MERCHANDISING PLAN

6.1 Distribution Channels and Communications Plan

OEMs will be made aware of the availability of this information through Software Engineering, Technical Support, the Hot Line, Marketing and any other departments that interface with OEMs.The information on protection procedures will be made available to interested and qualified OEMs through Technical Support (John Arkley).

6.2 Availability for New Products.

DOS 3.2	- Currently Available
DOS 3.3	- 30 January, 1980
Pascal	- 1 March, 1980

7. SUPPORT

7.1 Support

Engineering support will be through Systems Software and Technical Support. See section 2.3.1.

8. OPEN ISSUES

8.1 There is a need for a sophisticated software protection that will probably require hardware assistance. This level of protection will be necessary for both SARA and LISA software. A more sophisticated protection scheme would also be worthwhile for Apple II. Due to lack of resources, Engineering has NO further plans to continue this project. Prior to shipping software on Sara (and Lisa), we must have a protection scheme available. If this project is not part of the current Sara (Lisa) software effort, a project should be scheduled.

8.2 Apple must determine a policy and method for handling updating and returning of different products.

Rev. 11/13/79

APPLE ENGINEERING LAB PROJECT REPORT

PROJECT SSAFE MONTH January, 1980

PROJECT NO. E-78 PROJECT LEADER Randy Wigginton [illegible]

OTHER STAFF

OBJECTIVES FOR CURRENT MONTH:

Finish protection scheme for both Pascal and DOS 3.2 programs, and investigate hardware protection.

MONTHLY PROGRESS/STATUS:

SSAFE for DOS 3.2 finished with last remaining bugs being ironed out. Pascal protection not yet begun.

OBJECTIVES FOR COMING MONTH:

DOS 3.3 protection is next, followed by Pascal. Will finish DOS 3.3.

CRITICAL DEPENDENCIES:

None

	SCHEDULE		
MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
DOS 3.3 Protection			1/28/80
Pascal Protection	1/15/80	1/15/80	***

*** Temporarily delayed until after Sara Basic is Alpha released.

Inter Office Memo

Date:	February 15, 1980
To:	Distribution
From:	Jim Jatczynski [illegible]
Subject:	Protection of SARA Software

A meeting has been scheduled for Wednesday, February 20, in our Lazaneo Conference Room from 2:00 to 3:30.

The purpose of this meeting is to open the discussion of SARA Software Protection by gathering ideas from the attendees.

Distribution: Bruce Daniels
Al Hoffman
Randy Wigginton
Dick Huston
Tom Root

Attend if Interested: Donn Denman
Bob Etheredge

cc: Jack MacDonald
John Couch

①stop copying or ②stop loading / execution

① need back-up capability
need to copy out onto hard disk
need to modify OS to support scheme
∴ not promising area

② tie piece of software to a piece of hardware
add ROM card; each application requires hardware ROM key
software could check key periodically, in several places
ideally would like a plug in "key" (which may contain a ROM)
(AR) w/ each piece of software Jim will talk to Wendel
ROM could even contain applications code!

– Can use serial number to tie software diskette (or files) to system but it is a very restrictive mechanism.

Inter Office Memo

Date: February 25, 1980

To: Jack MacDonald

From: Jim Jatczynski JJ

Subject: SARA SOFTWARE PROTECTION

Attached is a summary of my investigation of SARA Software Protection.

cc: Route - General Distribution

Investigation of SARA Software Protection
Report 1 (25 February 1980)

Jim Jatczynski

PURPOSE

This report presents results of an initial attempt to characterize and solve the SARA software protection problem. It proposes two practical solutions.

SUMMARY

Effective software protection insures that use of a software entity is restricted to individuals who have purchased it. In particular, a protected program is executable only by a purchaser, and a protected data file is accessible only by a purchaser.

Two standard solutions to the protection problem are copy protection and execution protection. Copy protection should not be seriously considered as a general solution to the protection problem because it places too many restrictions on the user and has pervasive impact on system software. On the other hand, execution protection has inherent flexibilities that allow implementors to select an appropriate level of user restriction and limit the software development impact to only those software entities that require protection.

Two forms of execution protection are feasible for SARA. Execution authorization using the machine serial number is simple and effective but too restrictive to be used generally. Execution authorization using an uncopyable electronic key contained in a plug-in module is a powerful general solution. SARA software protection should be based primarily on this plug-in key method.

PROBLEM STATEMENT

As developers and sellers of software, Apple and other vendors face a costly bootlegging problem: anyone with suitable equipment can copy and sell the medium containing a valuable software product, generally at a much lower price than the developer's price. An effective means of software protection is needed to minimize the loss of revenue due to bootlegging.

Effective protection insures that use of software is restricted to individuals who have purchased it from an authorized vendor, or to agents of these individuals. More precisely, this means that the authorized vendors must have control over the number of usable copies of the software, but not necessarily over exactly who uses the copies or on which of many SARAs they are used.

GENERAL SOLUTIONS

Successful software bootlegging requires the ability to copy the software and then to execute the copied software (see Figure 1).

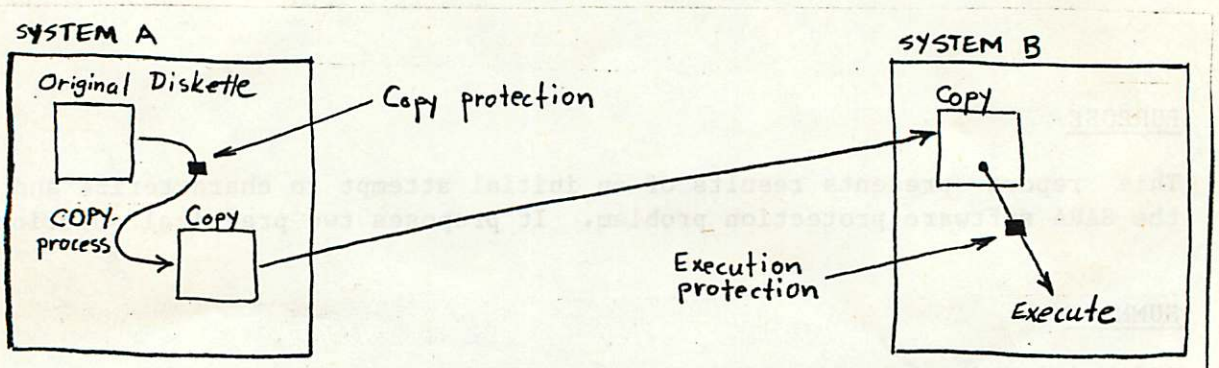


Figure 1. Successful Bootlegging Process

Bootlegging can be thwarted by introduction of adequate roadblocks in either the copy or execution process.

ATTRIBUTES OF ACCEPTABLE SOLUTIONS

The following attributes are desirable in any solution to the software protection problem:

User convenience

[1.1] avoidance of the need to involve the user in elaborate rituals in order to use the software

[1.2] ability to execute one's own copy of a software product on any SARA

[1.3] ability to make backup copies of one's copy of a software product

Manufacturing and distribution cost

[2.1] no differentiation required in manufacturing (i.e. all copies of a given software product are identical)

[2.2] no manufacturing or dealer intervention required to initiate the protection scheme

Solutions described in the following sections will be judged against these and additional criteria.

COPY PROTECTION SOLUTION

The concept of copy protection leads to several techniques, all intended to preclude creation of usable copies of original software. None of the examined schemes are deemed generally suitable for software protection because they have some or all of the following disadvantages:

[1] The user cannot make backup copies of purchased diskettes.

[2] The user cannot copy the application program from a diskette to his own medium (e.g. a hard disk).

[3] Most schemes require special data encoding or distortion that makes protected disks incompatible with unprotected disks. At best, a small set of programs must be written to deal with protected disks, and at worst, a customized operating system must be provided with the protected application.

Because of these disadvantages, execution protection is a more suitable general solution to the protection problem.

EXECUTION PROTECTION USING SARA SERIAL NUMBER

Each SARA has a unique (possibly modulo 65536) software readable serial number that can be used to implement various execution authorization schemes for protected applications.

One scheme would work as follows:

[1] When an individual purchases a SARA, the serial number is registered along with the individual's name.

[2] Thereafter, when the individual purchases protected software, the software vendor inserts the serial number at appropriate locations on the diskette.

[3] The protected application contains code to read the serial number of the machine it is executing on and compare it with the serial number written on the diskette. In case of a mismatch, the application program makes itself unexecutable.

Some refinements are needed to provide an acceptable level of protection: 1) the serial number should be encrypted before it is written on the diskette, 2) the protected application program should check repeatedly for serial number match, and checking code should be repeated at several locations in the program.

This scheme has several disadvantages: 1) serial number registration is a costly and error prone process, 2) the protected application is tied to exactly one machine, 3) differentiation is required during software manufacturing since each diskette must be customized with a serial number.

A refinement of the above technique eliminates problems 1) and 3). In the refined method, all protected application diskettes are initially identical, and certain locations contain data indicating that the diskette has never been used. When the application is first run, it checks these locations, and, because it finds that the disk has never been used, it reads the SARA serial number, encrypts it, and replaces the initial value with the encrypted serial number. On subsequent runs, the program finds that the special locations contain a non-initial value and therefore performs a serial number comparison.

Even with this refinement, the protected application is tied to exactly one machine. Therefore, this method is not generally applicable, but it may be used to protect programs such as SOS that can be tied to a single machine.

EXECUTION PROTECTION USING AN ELECTRONIC KEY

The method discussed in this section offers the greatest potential of being an acceptable general solution to the protection problem.

This method uses a lock and key implemented with the following hardware and software components:

[1] A software protection module that plugs into the SARA (possibly into a peripheral slot). The purpose of this module is to provide the interface between protected applications running on SARA and key modules that are part of each protected application package. Thus, the protection module must contain an externally accessible connector into which key modules can be plugged.

[2] A key module, one of which is provided in each application package. In order to execute an application, its key module must be plugged into the executing SARA's software protection module.

[3] Some form of authorization software included in the application program. This software uses the software protection module to access information contained in the plugged-in key module in order to determine whether or not the application should be allowed to run.

Software Protection Module

This module is a simple port that key modules are plugged into. It is a standard module compatible with all protected applications. Thus, a user must purchase and install the module only when he purchases his first protected application. Design of this module is straightforward except for the connector that the key modules plug into. The experience of other companies with plug-in software modules should be a useful guide to the reliability of such a connector.

If it is possible to do so, we should avoid using a peripheral slot for the software protection module. In any case, the key module plug must be easily accessible to the user.

SARA ONLY HAS 4!

Key Module

Design criteria for the key module include the following:

[1] It must be inexpensive so that it can be used even with relatively low-cost application programs.

[2] It should be compact so that a user can carry several from one location to another. MAGNETIC STRIP ON A CARD? (EASY TO FORGE?)

[3] The connector must withstand a large number of insertions.

[4] The module should be capable of containing at least 256 bytes of information.

[5] It should contain a means of preventing access to the information unless a proper sequence of bits has been sent to it (possibly a state machine). ?

Authorization Software

The protected application software protects itself in the sense that it either authorizes or denies use of itself by checking information in the key module. Several authorization schemes are possible; two likely candidates are:

[1] Place a "secret code" in the key module and in the application code. The authorization software checks the codes against one another. The "secret code" must be suitably hidden in the application code, and the authorization software should be hidden and/or repeated several times in order to complicate software modifications intended to bypass it.

[2] Place several crucial subroutines in the key module. Execute them directly from the module if that is possible; otherwise copy them to main memory before starting execution of the application.

Since the application program protects itself, the scheme it uses can be made arbitrarily complex. In any case, it is very important to 1) maintain secrecy of the information in the key module and 2) hide or obscure the authorization software portion of the application program.

Advantages

This method of software protection has several important advantages:

[1] Any protected application may be run on any machine that has a software protection module as long as the application's key module is plugged in.

[2] There is no restriction on copying of application diskettes.

[3] No manufacturing differentiation or dealer intervention is required to implement the scheme.

[4] The exact means of protection is left up to the application vendor who may specify both the contents of the key module and the authorization code.

<u>Disadvantages</u>

[1] One-time purchase of the software protection module is required.

[2] A key module is a required part of every protected application package.

[3] If no other means can be found to connect the software protection module to the SARA, it will be necessary to use a peripheral slot.

<u>DATA PROTECTION</u>

Vendors may sell diskettes that contain valuable data rather than valuable application software. The key protection scheme can be used to limit access to this data as follows:

[1] Encrypt the data that is placed on the distribution medium.

[2] Place the encryption key in the key module that is sold with the data and programs that access it.

<u>IMPLEMENTATION NOTES</u>

A conversation with Wendell Sander brought up the following implementation issues:

[1] Key modules must be made extremely difficult to copy. Possible ways of doing this include 1) using semi-custom chips that include both the ROM and state machine, 2) potting the entire circuit in plastic, 3) using hybrid technology.

[2] Only a few options are available for connecting the software protection module to SARA: 1) peripheral slot, 2) game I/O, 3) Trendcom port, 4) RS 232 port. Only 1) and 2) seem reasonable.

[3] Use of a semi-custom chip in the key module involves a mask charge of approximately $2000 for each protected application. Therefore the software must be sold in sufficient volume to justify the mask charge.

[4] Manufacturing cost of the software protection module is probably about $15.

[5] Manufacturing cost of the key modules is probably about $15.

2/26/80

Randy - SSAFE

Mainboard	ROMs
ROM card	"s

AII has orig. IB ROM + monitor ROM

AII+ AS " auto boot ROM

ROM card: AS ROM, uses main board ROM
IB ", used std. monitor ROM on card (not mainboard)

Problem:

On ROM card, flip switch + hit RESET, goes into monitor
- either main board ROM
- or card "

Can be beaten: — for memory access, not disk file protection

1. Pull out ROM card, disable software selection pin on chip. Then locks
2. Put ROM card in slot other than ∅, may also require mod to card (Avericchio discovered this)

On IB ROM card, can remove solder bridge to disable ROM on card

2/27/80

Bruce

1. Put part of prog in key
   - use state mach as 1 or more doors to contents of ROM

2. Encrypt prog
   - ROM contains decryption key

- Would like set of sockets on a single peripheral card, one for Pascal, FTN, Visicalc
Install card once, install each ROM once.
System searches card for a ROM which responds properly (like looking thru a box of keys)

Std

BOOT
Prom loads TØ, SØ into P3; transfers control to top of page 3. Last two bytes are A,N

(w/ PROM)
This boot routine then loads TO, SØ → N @ A (RWTS, load routines). Then transfer to A+256 (loader prog.)
Now can drag in rest of DOS.

µCHESS (encrypted by D. Huston)

Encrypted TO, S1-SN. ~~If try to load via boot~~ ~~PROM~~ Page 3 and 13 sect. boot PROM are required to load TØ, S1-SN. Tied to 13 sector boot PROM

VISICALC

~~3 copies~~ ASAP NS

Inter Office Memo

Jack / Tim
Whats the next step?
i.e who has the ball?
[illegible]

Date: February 29, 1980

To: Distribution

From: Jim Jatczynski [illegible]

Subject: SARA SOFTWARE PROTECTION

Revision B of my report on SARA Software Protection is attached. Changes from revision A are indicated by a bar in the right margin.

Please review the report and return any comments to me by March 12, 1980.

I will be setting up a meeting to discuss the proposed protection scheme some time before March 12.

Distribution: John Couch
Jack MacDonald
Bruce Daniels
System Software (Route)
Richard Zimmerman
Wendell Sander
Dennis Rieger
Don Bryson

Inter Office Memo

Date: February 29, 1980

To: Distribution

From: Jim Jatczynski [illegible]

Subject: SARA SOFTWARE PROTECTION

Revision B of my report on SARA Software Protection is attached. Changes from revision A are indicated by a bar in the right margin.

Please review the report and return any comments to me by March 12, 1980.

I will be setting up a meeting to discuss the proposed protection scheme some time before March 12.

Distribution: John Couch
Jack MacDonald
Bruce Daniels
System Software (Route)
Richard Zimmerman
Wendell Sander
Dennis Rieger
Don Bryson

Investigation of SARA Software Protection

Report 1

Revision A 25-Feb-1980
Revision B 29-Feb-1980

Jim Jatczynski

PURPOSE

This report presents results of an initial attempt to characterize and solve the SARA software protection problem. It proposes two practical solutions.

SUMMARY

Effective software protection insures that use of a software entity is restricted to individuals who have purchased it. In particular, a protected program is executable only by a purchaser, and a protected data file is accessible only by a purchaser.

Two standard solutions to the protection problem are copy protection and execution protection. Copy protection should not be seriously considered as a general solution to the protection problem because it places too many restrictions on the user and has pervasive impact on system software. On the other hand, execution protection has inherent flexibilities that allow implementors to select an appropriate level of user restriction and limit the software development impact to only those software entities that require protection.

Two forms of execution protection are feasible for SARA. Execution authorization using the machine serial number is simple and effective but too restrictive to be used generally. Execution authorization using an uncopyable electronic key contained in a plug-in module is a powerful general solution. SARA software protection should be based primarily on this plug-in key method.

PROBLEM STATEMENT

As developers and sellers of software, Apple and other vendors face a costly bootlegging problem: anyone with suitable equipment can copy and sell the medium containing a valuable software product, generally at a much lower price than the developer's price. An effective means of software protection is needed to minimize the loss of revenue due to bootlegging.

Effective protection insures that use of software is restricted to individuals who have purchased it from an authorized vendor, or to agents of these individuals. More precisely, this means that the authorized vendors must have control over the number of usable copies of the software, but not necessarily over exactly who uses the copies or on which of many SARAs they are used.

GENERAL SOLUTIONS

Successful software bootlegging requires the ability to <u>copy</u> the software and then to <u>execute</u> the copied software (see Figure 1).

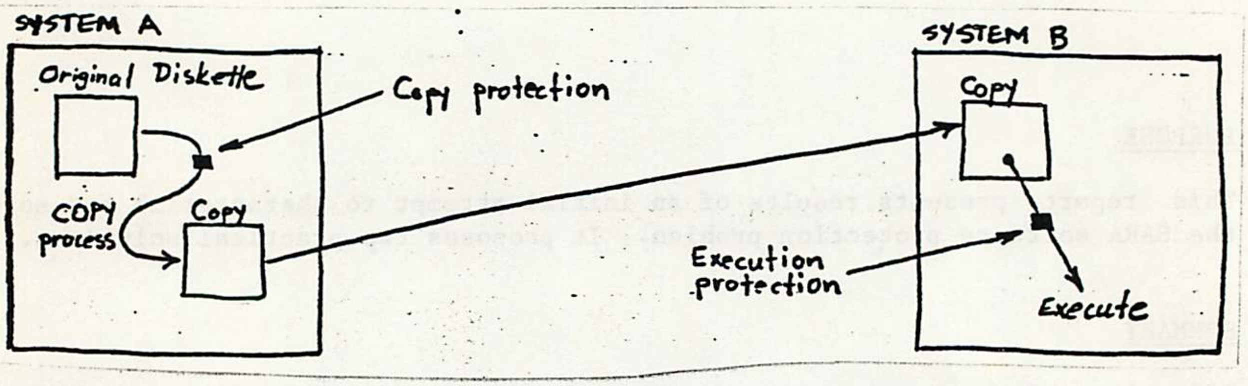


Figure 1. Successful Bootlegging Process

Bootlegging can be thwarted by introduction of adequate roadblocks in either the copy or execution process.

ATTRIBUTES OF ACCEPTABLE SOLUTIONS

The following attributes are desirable in any solution to the software protection problem:

User convenience

[1.1] avoidance of the need to involve the user in elaborate rituals in order to use the software

[1.2] ability to execute one's own copy of a software product on any SARA

[1.3] ability to make backup copies of one's copy of a software product

Manufacturing and distribution cost

[2.1] no differentiation required in manufacturing (i.e. all copies of a given software product are identical)

[2.2] no manufacturing or dealer intervention required to initiate the protection scheme

Solutions described in the following sections will be judged against these and additional criteria.

COPY PROTECTION SOLUTION

The concept of copy protection leads to several techniques, all intended to preclude creation of usable copies of original software. None of the examined schemes are deemed generally suitable for software protection because they have some or all of the following disadvantages:

[1] The user cannot make backup copies of purchased diskettes.

[2] The user cannot copy the application program from a diskette to his own medium (e.g. a hard disk).

[3] Most schemes require special data encoding or distortion that makes protected disks incompatible with unprotected disks. At best, a small set of programs must be written to deal with protected disks, and at worst, a customized operating system must be provided with the protected application.

Because of these disadvantages, execution protection is a more suitable general solution to the protection problem.

EXECUTION PROTECTION USING SARA SERIAL NUMBER

Each SARA has a unique (possibly modulo 65536) software readable serial number that can be used to implement various execution authorization schemes for protected applications.

One scheme would work as follows:

[1] When an individual purchases a SARA, the serial number is registered along with the individual's name.

[2] Thereafter, when the individual purchases protected software, the software vendor inserts the serial number at appropriate locations on the diskette.

[3] The protected application contains code to read the serial number of the machine it is executing on and compare it with the serial number written on the diskette. In case of a mismatch, the application program makes itself unexecutable.

Some refinements are needed to provide an acceptable level of protection: 1) the serial number should be encrypted before it is written on the diskette, 2) the protected application program should check repeatedly for serial number match, and checking code should be repeated at several locations in the program.

This scheme has several disadvantages: 1) serial number registration is a costly and error prone process, 2) the protected application is tied to exactly one machine, 3) differentiation is required during software manufacturing since each diskette must be customized with a serial number.

A refinement of the above technique eliminates problems 1) and 3). In the refined method, all protected application diskettes are initially identical, and certain locations contain data indicating that the diskette has never been used. When the application is first run, it checks these locations, and, because it finds that the disk has never been used, it reads the SARA serial number, encrypts it, and replaces the initial value with the encrypted serial number. On subsequent runs, the program finds that the special locations contain a non-initial value and therefore performs a serial number comparison.

Even with this refinement, the protected application is tied to exactly one machine. More damaging, however, is the fact that it is very easy to bulk copy previously unused diskettes. Therefore, this method is not generally applicable, but it may be used to protect programs such as SOS that can be tied to a single machine. For effective protection, dealer initialization of diskettes would be required.

EXECUTION PROTECTION USING AN ELECTRONIC KEY

The method discussed in this section offers the greatest potential of being an acceptable general solution to the protection problem.

This method uses a lock and key implemented with the following hardware and software components:

[1] A software protection module that plugs into the SARA (possibly into a peripheral slot). The purpose of this module is to provide the interface between protected applications running on SARA and key modules that are part of each protected application package. Thus, the protection module must contain an externally accessible connector into which key modules can be plugged.

[2] A key module, one of which is provided in each application package. In order to execute an application, its key module must be plugged into the executing SARA's software protection module.

[3] Some form of authorization software included in the application program. This software uses the software protection module to access information contained in the plugged-in key module in order to determine whether or not the application should be allowed to run.

Software Protection Module

This module is a simple port that key modules are plugged into. It is a standard module compatible with all protected applications. Thus, a user must purchase and install the module only when he purchases his first protected application. Design of this module is straightforward except for the connector that the key modules plug into. The experience of other companies with plug-in software modules should be a useful guide to the reliability of such a connector.

If it is possible to do so, we should avoid using a peripheral slot for the software protection module. In any case, the key module plug must be easily accessible to the user.

Key Module

Design criteria for the key module include the following:

[1] It must be inexpensive so that it can be used even with relatively low-cost application programs.

[2] It should be compact so that a user can carry several from one location to another.

[3] The connector must withstand a large number of insertions.

[4] If it is determined that the module might usefully contain information in ROM, at least 256 bytes of ROM should be present. If ROM is present, there must be a means of preventing access to it, for example, a state machine that must be driven through a complex homing sequence in order to enable ROM access.

[5] Possibly the simplest implementation of the key module would consist of only a state machine. The machine should be drivable into its initial state via a homing sequence. Subsequently, it should respond to a correct input sequence with its secret output sequence that is to be verified by the authorization software.

Other Design Criteria

It may be necessary to design the software protection module and key modules so that two or more key modules can be plugged in simultaneously. This would be necessary if two or more protected applications were run together, for example, a protected plotting package along with a protected database manager. Questions to consider include: 1) how many plugs are enough and 2) is there an alternative that will allow several protected applications to be serviced by one key module?

Portability of software and associated key modules is important, but effortless day to day portability is not required. It is more important to enable the user to plug one or more key modules into his home system and forget about them than to minimize the complexity of plugging and unplugging key modules.

Authorization Software

The protected application software protects itself in the sense that it either authorizes or denies use of itself by checking information in the key module. Several authorization schemes are possible; two likely candidates are:

[1] Place a "secret code" in the key module and in the application code. The authorization software checks the codes against one another. The "secret code" must be suitably hidden in the application code, and the authorization software should be hidden and/or repeated several times in order to complicate software modifications intended to bypass it. The secret code is read from the key module my driving the module′s state machine through a homing sequence and then through a key access sequence during which the secret key value is read.

[2] Place several crucial subroutines in the key module. Execute them directly from the module if that is possible; otherwise copy them to main memory before starting execution of the application.

Since the application program protects itself, the scheme it uses can be made arbitrarily complex. In any case, it is very important to 1) maintain secrecy of the information in the key module and 2) hide or obscure the authorization software portion of the application program.

Advantages

This method of software protection has several important advantages:

[1] Any protected application may be run on any machine that has a software protection module as long as the application′s key module is plugged in.

[2] There is no restriction on copying of application diskettes.

[3] No manufacturing differentiation or dealer intervention is required to implement the scheme.

[4] The exact means of protection is left up to the application vendor who may specify both the contents of the key module and the authorization code.

Disadvantages

[1] One-time purchase of the software protection module is required.

[2] A key module is a required part of every protected application package.

[3] If no other means can be found to connect the software protection module to the SARA, it will be necessary to use a peripheral slot.

DATA PROTECTION

Vendors may sell diskettes that contain valuable data rather than valuable application software. The key protection scheme can be used to limit access to this data as follows:

[1] Encrypt the data that is placed on the distribution medium.

[2] Place the encryption key in the key module that is sold with the data and programs that access it.

IMPLEMENTATION NOTES

A conversation with Wendell Sander brought up the following implementation issues:

[1] Key modules must be made extremely difficult to copy. Possible ways of doing this include 1) using semi-custom chips that include both the ROM and state machine, 2) potting the entire circuit in plastic, 3) using hybrid technology.

[2] Only a few options are available for connecting the software protection module to SARA: 1) peripheral slot, 2) game I/O, 3) Trendcom port, 4) RS 232 port. Only 1) and 2) seem reasonable.

[3] Use of a semi-custom chip in the key module involves a mask charge of approximately $2000 for each protected application. Therefore the software must be sold in sufficient volume to justify the mask charge.

[4] Manufacturing cost of the software protection module is probably about $15.

[5] Manufacturing cost of the key modules is probably about $15.

Rev. 11/13/79

✓

APPLE ENGINEERING LAB
PROJECT REPORT

PROJECT SSAFE MONTH February, 1980

PROJECT NO. E-78 PROJECT LEADER Randy Wigginton [illegible] [illegible]

OTHER STAFF

OBJECTIVES FOR CURRENT MONTH:

DOS 3.3 protection is next, followed by Pascal. Will Finish DOS 3.3

MONTHLY PROGRESS/STATUS:

DOS 3.2 and DOS 3.3 protection was completed.

OBJECTIVES FOR COMING MONTH:

Production support of SSAFE, do Q.A. verify programs for both SSAFE 3.2 and 3.3, also do Pascal protection.

CRITICAL DEPENDENCIES:

SCHEDULE

MILESTONE	ORIG. DATE	LAST MO.	CURRENT PLAN
Pascal Protection			3/14/80

Inter Office Memo

Date: March 6, 1980

To: Jim Jatezynski

From: Barry Yarkoni [illegible]

Subject: SARA Software Protection - Comments

DISTRIBUTION
============

J. Couch
J. McDonald
B. Daniels
R. Zimmerman
W. Sander
D. Rieger
D. Bryson
M. Kane
S. Jobs

1. It is not necessary for protection to be thorough or protect against experts. We are out to stop the geometric replication of software.

2. Including hardware, such as a ROM key with software is totally unacceptable from a cost point of view. This is a case of the cure being worse than the disease.

3. How about encryption, where the SARA serial # along with a password form the encryption key. This means that each SARA would have a unique encryption key for a given piece of software. This key could be provided to customers either by our dealers or through a "hot line."

We are not there yet. It is crucial that we get there soon...whatever it may be!

Inter Office Memo

Date:	March 14, 1980
To:	Distribution
From:	Jim Jatczynski [illegible]
Subject:	SARA SOFTWARE PROTECTION

In my memo dated February 29, 1980, I said I would set up a meeting to discuss protection before March 12. However, based on response to the report attached to that memo, I've decided to document additional issues and a newly proposed protection scheme before calling such a meeting.

Please review and comment on the attached report before March 21. I will determine a meeting date after I receive your feedback.

Distribution:
- John Couch
- Jack MacDonald
- Richard zimmmerman
- Dennis Rieger
- Don Bryson
- Wendell Sander
- Bruce Daniels

Investigation of SARA Software Protection

Report 2

Revision A (14 March 1980)

Jim Jatczynski

PURPOSE

Reviewers of Investigation of SARA Software Protection Report 1 (Revision B (28 February 1980)) have raised new issues and suggested an additional protection scheme. Report 2 presents these issues, reiterates the key-based protection scheme of Report 1, describes the newly proposed scheme, and presents advantages and disadvantages of both schemes. We cannot begin implementation until we resolve these issues and choose one of the two alternative protection methods. This report is intended to provide more input for the decision process.

SOFTWARE PROTECTION ISSUES

Goals of Software Protection

It is not necessary for the protection scheme to protect against experts. We intend only to stop the relatively casual geometric replication of software. That is, we need only provide a scheme that thwarts most but not all potential copiers.

Cost of Bootlegging Problem

Cost of the bootlegging problem to Apple and other vendors is unknown. In order to justify effort in this area, we need to assess the potential extent of lost revenue. It is particularly important to note that SARA is aimed at a market in which casual bootlegging seems significantly less likely than in the Apple II market. If most bootlegging is done by experts, the solutions proposed here will not prevent this loss of revenue.

Cost of the Protection Scheme

We cannot allow the cost of the protection scheme to exceed more than a very small percentage of the cost of each protected application program; a protection cost of less than 5% of the application cost seems desirable. Based on an estimated hardware solution cost of $15 to $30, only $300 to $600 software products would be candidates for protection. It is important to note that significant additional software development costs are required by both hardware-based and software-based protection methods.

CANDIDATE SOFTWARE PROTECTION SCHEMES

This section describes two primary candidate protection schemes and lists their advantages and disadvantages. The hardware key scheme has already been described in Report 1, so only a summary of the scheme will be given here.

Electronic Key Protection

This scheme has three components:

[1] A software protection module-- a single card connected to SARA that is used by all protected programs. It provides program access to information in key modules.

[2] A key module for each protected application. To run the application, the key module must be plugged into the software protection module.

[3] Authorization software "scattered" throughout the protected program. This software verifies the right of the user to execute the program by assuring itself of the presence of the appropriate key module.

The relationship of these three components is shown in Figure 1.

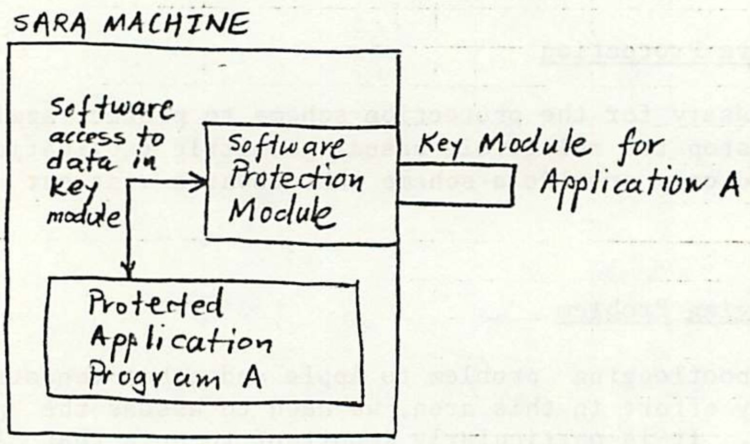

SARA MACHINE
Software access to data in key module
Software Protection Module
Key Module for Application A
Protected Application Program A

Figure 1. Electronic Key Protection

Advantages

[1] Any protected program may be run on any machine that has a software protection module as long as the correct key module is plugged in.

[2] No restriction on copying application diskettes for backup.

[3] No differentiation in manufacturing.

[4] No dealer or Apple intervention required to initiate the protection mechanism.

[5] Flexibility in implementation of authorization software by each application writer.

[6] Easily extended to data protection.

Disadvantages

[1] If no other means is found to connect the software protection module to the SARA, a peripheral slot will have to be used.

[2] Added cost to user of a software protection module.

[3] Added cost to user of a key module for each protected application.

[4] Added cost to Apple of developing the software protection and key modules, customizing the key module for each application, and writing the authorization software for each application.

[5] Inconvenience of plugging in the key modules (these can probably be designed so it is necessary to do this only once).

<u>Serial Number and Password Protection</u>

This newly proposed method uses the built-in serial number in conjunction with a dealer- or Apple-supplied password in order to decrypt software that is encrypted on the application diskette. The scheme works as follows:

[1] All application diskettes contain an identical encrypted version of the protected application:

encrypted program = f1 (key1, program)

COSTS TIME

[2] Each time the user runs the program, it is decrypted as it is loaded into memory:

VULNERABLE TO BEING COPIED WHEN IN MEMORY

program = f2 (key2, encrypted program)

[3] The protection mechanism computes key2 as

key2 = f3 (key1, password, machine serial number)

Password is computed by a dealer or Apple and is a function of the particular application and the machine serial number that must be supplied by the user in order to obtain the password when the software is purchased. Key1 must be known to the protection mechanism in the user's machine, and the machine serial number is built into each machine.

In summary, the protected software is encrypted and decryption requires knowledge of the machine serial number and an Apple-supplied password that is a function of the application and the machine serial number. Figure 2 illustrates the entire process.

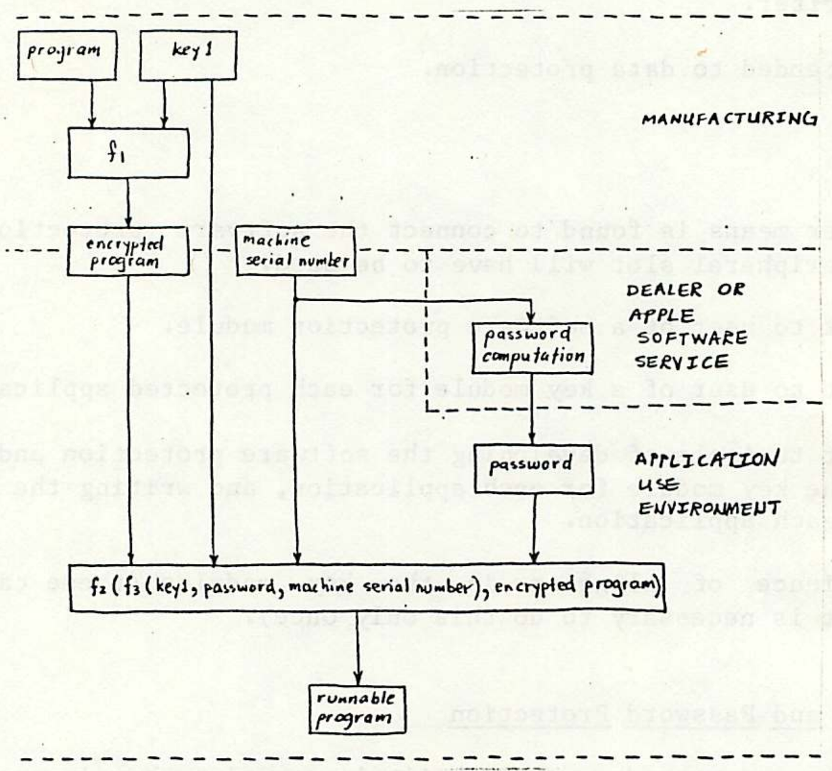

program
key 1
f_1
MANUFACTURING
encrypted program
machine serial number
DEALER OR APPLE SOFTWARE SERVICE
password computation
password
APPLICATION USE ENVIRONMENT
f_2 (f_3 (key1, password, machine serial number), encrypted program)
runnable program

Figure 2. Serial Number and Password Protection Process

Advantages

[1] No additional hardware required for each protected application.

[2] No restriction on copying application diskettes for backup.

[3] No differentiation in software diskette manufacturing.

[4] Easily extended to data protection.

[5] Uniform mechanism for all applications.

Disadvantages

[1] Applications are tied to a single machine since decryption depends on the machine serial number.

[2] Differentiation is required in hardware manufacturing--each machine must be given a unique software accessible serial number. ALREADY BEING DONE

[3] Dealer or Apple intervention is required to supply the password given the machine serial number.

[4] Added cost to Apple of developing the protection mechanism, installing the mechanism in each protected application, and providing passwords to users.

[5] DECRYPTION TIME EACH TIME FILE/PROGRAM IS ACCESSED.
[6] PROGRAM VULNERABLE WHEN IN MEMORY

Additional Considerations

[1] Exactly where is the decryption performed? Is each application responsible for decrypting and loading itself, or should we build a general mechanism into each of the language systems?

[2] Who should provide the passwords? Choices are: Apple, the dealers, the vendor of the protected application.

<u>REQUIRED</u> <u>DECISION</u>

The methods presented here represent two main classes of solutions to the protection problem: 1) hardware-software methods and 2) software-only methods. We need to make two decisions as soon as possible:

[1] Does the cost of the bootlegging problem justify the cost of any solution?

[2] If so, which of the two solutions (or some other solution) should we adopt?

How & when is password supplied? (Saved on diskette)

Say I wish to protect a multi-pass translator. If each overlay is encrypted, it will be a real pain to run it.

3/20/80

Randy

- Used Barth's scheme

got ~~[illegible]~~ 4 K words
but 30 sec / stmt to compile

DATE: March 24, 1980

TO: Distribution

FROM: Dennis Rieger

SUBJECT: SARA Protection

There will be a SARA Protection meeting scheduled for Wednesday, March 26, 1980.

Please meet in the Executive Board Room at 1:00 to 2:30.

Distribution: Don Bryson
John Couch
Jim Jatczynski
Jack MacDonald
Pat Marriott
Barry Yarkoni

/pc

3/25/80

Disadvantages of SSAFE

- (Wyman) A non-autoboot ROM anywhere in the system will cause SSAFE to abort.*

- It's been cracked

*Randy says this can be fixed. Will bypass check on motherboard ROM, however this will permit ~~integer basic~~ entering Integer Basic and pressing RESET

3/26/80

Sara Protection

- (GLANVILLE) Extra room at end of block that can be used.

AR Reaffirm that serial number is still in Sara! (See Zimmerman)

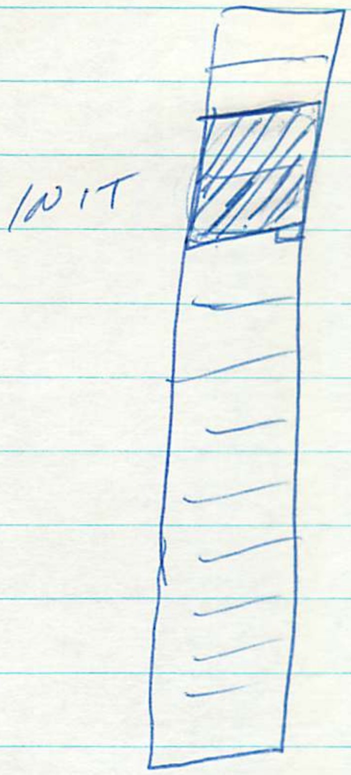
① - Will protect Visicalc simply (a la GLANVILLE/Wigginton for FTN) by 5/1
AR
- Need to automate 800 number scheme

- Will license Visicalc, WP

- Will look at long term solution (Eng, Mktg, Mfg)

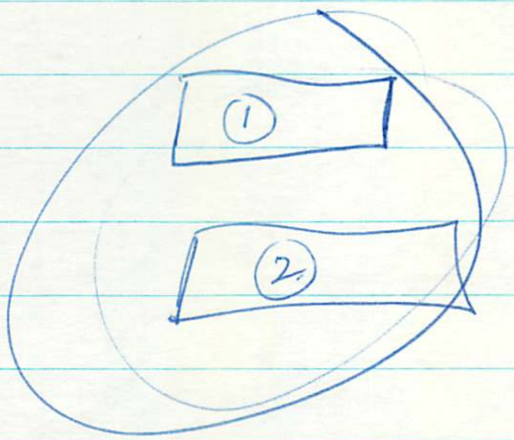
3/26/80

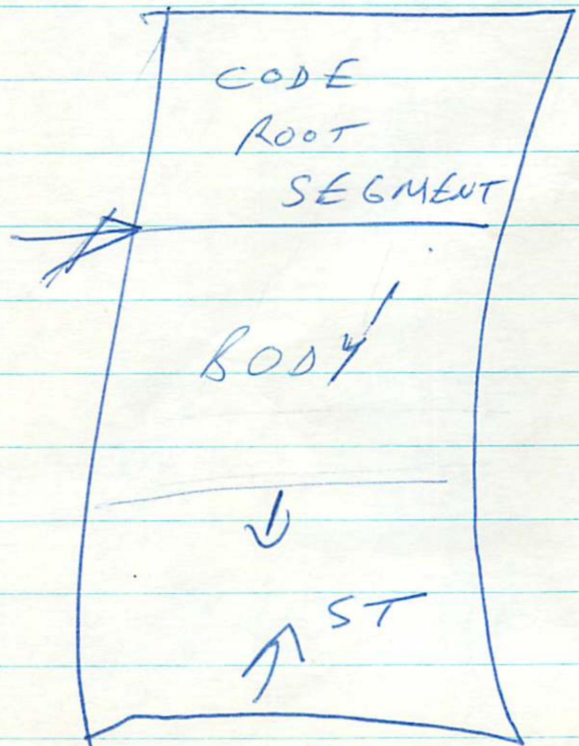
Glanville



.CODE

0 1





3/26/80

FTN Protection

- Segment procedure INIT is goobered on diskette such that filer will not be able to read it (e.g. checksum may be wrong).

- Before it is read, Randy's ASM routine #1 is called which changes some tables in the OS so INIT can be ~~called~~ loaded with no errors

- After INIT is loaded and executed, ASM routine #2 is called to reset OS tables

APPLE ENGINEERING LAB
PROJECT REPORT

PROJECT: SSAFE MONTH: MARCH 1980

PROJECT NO: E-78 PROJECT LEADER: RANDY WIGGINTON
OTHER STAFF:

LAST MONTH'S OBJECTIVES:

PERFORM FINAL PROTECTION ON FORTRAN AND PILOT AND WHATEVER ELSE NEEDED PROTECTING.

MONTHLY PROGRESS/STATUS:

FORTRAN protection was completed, with protection scheme being given go-ahead by Barth & Glanville. A method was discovered whereby FORTRAN will theoretically work on all future releases of the PASCAL system.

OBJECTIVES FOR COMING MONTH:

Document procedure to protect PASCAL programs, and finish documentation on BASIC ssafe mechanism. An investigation will be made into automating the PASCAL protection scheme, which currently requires approximately 7 man-hours of my time per program to be protected.

CRITICAL DEPENDENCIES:

MUST RECIEVE SOURCE TO ALL PROGRAMS TO BE PROTECTED.

SCHEDULE

MILESTONE DESCRIPTION	ORIG. DATE	LAST MONTH'S	CURRENT PLAN
-DEPENDS UPON DATE OF RECIEPT OF SOURCE FOR PROGRAMS-			

APPLE ENGINEERING LAB
PROJECT REPORT

PROJECT: SSAFE MONTH: MARCH 1980

PROJECT NO: E-78 PROJECT LEADER: RANDY WIGGINTON
OTHER STAFF:

LAST MONTH'S OBJECTIVES:

IMPLEMENT PASCAL PROTECTION UNDER THE RUN-TIME SYSTEM.

MONTHLY PROGRESS/STATUS:

PASCAL PROTECTION WAS COMPLETED. METHOD COMPLETED REQUIRES MODIFICATION OF THE SOURCE PROGRAM. THIS WAS DECIDED AS A REASONABLE PRICE TO PAY FOR PROTECTION. SINCE THE PASCAL OPERATING SYSTEM PASSES NO INFORMATION REGARDING WHO IS REQUESTING ACCESS TO A FILE(I.E., WHETHER THIS IS A UNITREAD/WRITE, A FILER OPERATION, A PROGRAM REQUEST, ETC.), THIS WAS THE ONLY FEASIBLE METHOD. THE METHOD CAN BE TRANSFERRED TO SARA WITH MODIFICATIONS TO THE OPERATING SYSTEM.

OBJECTIVES FOR COMING MONTH:

PERFORM FINAL PROTECTION ON FORTRAN AND PILOT AND WHATEVER ELSE NEEDS PROTECTING.

CRITICAL DEPENDENCIES:

MUST RECIEVE SOURCE TO ALL PROGRAMS TO BE PROTECTED. FOLLOWING THIS, PROTECTION WILL REQUIRE AN ADDITIONAL 7 MAN-HOURS PER DISKETTE OF MY TIME.

SCHEDULE

MILESTONE DESCRIPTION	ORIG. DATE	LAST MONTH'S	CURRENT PLAN
-DEPENDS UPON DATE OF RECIEPT OF SOURCE FOR PROGRAMS-			

APPLE ENGINEERING LAB
PROJECT REPORT

PROJECT: SSAFE

MONTH: MARCH 1980

PROJECT NO: E-78

PROJECT LEADER: RANDY WIGGINTON
OTHER STAFF:

LAST MONTH'S OBJECTIVES:

TO FINISH PROTECTION ON PASCAL.

MONTHLY PROGRESS/STATUS:

FORTRAN PROTECTION WAS COMPLETED AND FOUND TO BE INADEQUATE. A METHOD OF PASCAL PROTECTION WAS DISCOVERED.

OBJECTIVES FOR COMING MONTH:

IMPLEMENT PASCAL PROTECTION UNDER THE RUN-TIME SYSTEM.

CRITICAL DEPENDENCIES:
-NONE-

SCHEDULE

MILESTONE DESCRIPTION	ORIG. DATE	LAST MONTH'S	CURREN PLAN
PASCAL PROTECTION	1/15/80	***	4/15/8

*** PASCAL PROTECTION WAS DELAYED UNTIL AFTER COMPLETION OF SARA BASIC ALPHA RELEASE.

4/1/80

Pascal Protection

- no dynamic file protection
- files are read only; can't write to a protected file

- mod Pascal BIOS + OS

- ~~protects run-time programs as well as~~

- capable of protecting any file in either Pascal environment (run-time or regular)

To: Jack MacDonald

From: Randy Wigginton R.W

Date: April 9, 1980

Subject: Pascal Protection Scheme

The protection scheme for PASCAL is very nearly finished and this document is to explain the features and quirks of the scheme. None of the technical details are discussed here.

In order to protect a program, a simple modification to the source program to be protected must be made, followed by a re-compile. This is necessary so that a protected program may be run at any time on any system. The features of this include being able to boot on any system, then run the protected software, then returning to the top-level command line in the PASCAL system. Another advantage of this scheme is that a program, such as PFS, which uses all space in memory available, could be shipped on a run-time system diskette, but if a user owned a language card, s/he would have extra space available to the program. Note that Mike Kane will soon be sending out a memo stating that this is how the protection scheme must work--on any system at any time, whether the user has booted on an old SYSTEM.APPLE or a new one. This will be a pleasant surprise to everyone that we've already allowed for this. However, a disadvantage that should be pointed out is that when yet another release of the operating system comes out, protected software will only run on the 'older' operating systems--there is no way that my software can allow for all future revisions of the operating system and core routines.

The features of this scheme are:

-Protection of program-referenced data files. Note, for example, that this scheme will not allow protection of the SYSTEM.LIBRARY file(as desired for FORTRAN), because this file will be referenced by the PASCAL SYSTEM.LINKER, which is not a protected program. However, programs like the Tax-Planner which use data files may now protect those files.

-Files that are protected may be modified on the fly-i.e., the Tax Planner could make changes to some or all of it's files, yet they would remain protected. Note that files that are protected may grow in size, but only a pre-selected portion of the file will remain protected. This means that a typical user using the filer would still be unable to transfer the file, but a fairly sophisticated user could examine the sectors of the file that are not protected. In all likelihood this will not make any difference to anybody.

Note that if a user hits the reset key during a protected program's execution, s/he will have to re-boot the system. (System will hang upon Reset) This should be pointed out. Marketting just can't have everything.

The documentation on the technical aspects of both this protection scheme and the SSAFE scheme for DOS 3.2&3.3 should be done soon. Documentation has already been started, but will require at least another 3-4 man days to complete. Of course, I would expect the technical documentation to be kept somewhere fairly secure--protection schemes aren't much use when users know the method used. This documentation will be quite essential to the person after myself who assumes the responsibility of modifying & protecting PASCAL programs, since although the process will be fairly straightforward, it will never be brought to the level of SSAFE as far as simplicity goes.

Inter Office Memo

Date: April 25, 1980

To: Jack MacDonald

From: Jim Jatczynski [illegible]

Subject: Recommendation for Apple III Machine Readable Serial Number

Storage space for a machine readable serial number has been reserved in the Apple III ROM. This memo recommends standards for serial number assignment based on the assumption that future software protection schemes will make use of the serial number.

Recommendations

[1] The serial number field should be 32 bits wide.

[2] Each machine should have a unique serial number.

[3] The serial number should be "non-significant." That is, it should neither have meaning nor be broken up into meaningful fields.

[4] If more than one manufacturing station or manufacturer is used, care must be taken to avoid serial number duplication.

[5] An additional machine type field may be desirable, but this should not be part of the serial number. The machine type field will not be used for software protection.

```
              x
             xx
            xxx
           xxx
           xx
   xxxx    x    xxxx
  xxxxxxxxxxxxxxxxxx
 xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxx
 xxxxxxxxxxxxxxxxxxxx
  xxxxxxxxxxxxxxxxxx
   xxxxxxxxxxxxxxxx
    xxxxxxxxxxxxxx
     xxxxx  xxxxx
```

FROM: Pete Sinclair DATE: April 30, 1980 ?

TO: Dennis Rieger SUBJECT: Software Protection

Dennis Rieger	
Jim Jatczynski	
John Couch	Steve Jobs
Jack MacDonald	Pat Marriott
Bruce Daniels	Trip Hawkins
Dick Zimmerman	Taylor Pohlman
Wendell Sander	Rob Campbell
Don Bryson	Steve Wozniak
Barry Yarkoni	Mike Kane

Having just joined the team working on software protection schemes, I would like to take this opportunity to summarize my impressions of what I see happening in this area. Basically, I beleive that the course being taken is leading us toward a cure that is more painful than the disease. A number of conflicting goals and restrictions have been raised, the combination of such has tended to distort the primary issue. I think that we need to lay down a single set of goals and restrictions in order to properly evaluate proposed solutions.

Software Protection Goals

As I see the issue, we actually have three levels of complementary goals:

Primary Goal:	Prevent people from using copies of Apple software products that they have not rightfully purchased.
Secondary Goal:	Provide a method for Apple compatible software vendors which allows them to protect their

software from use by persons who have not rightfully purchased it.

Desireable Goal: Allow users to protect their applications and data from being pirated by outside individuals.

The solution chosen to satisfy these goals must adhere to a few limitations in order to meet Apple standards as well as satisfy user expectations. A few of the major restrictions are as follows:

1. Users should be able to execute protected software which they have rightfully purchased on any Apple system.

2. Users need to possess or have access to duplicates of their application diskettes in case their diskette is rendered unusable for any reason.

3. Protected software should be made no more difficult to use than was the original unprotected product.

4. The protection scheme should not add significanlty to the cost of the product (<10%).

While many more restrictions can be created, this basic set should cover most of the objections that might come up about any chosen solution.

The first restriction (use on multiple systems) does have two sides. If one assumes that purchasing a software product only gives the user the right to execute it on a single machine, then this restriction does not apply. I believe, however, that our marketplace and products demand that the software be executable on any system as long as an Apple produced application diskette is used. Multi-terminal system manufacturers typically restrict software to running on a single machine. But that software is accessable from any terminal on the machine. Since the Apple concept is to put computers at each professional, secretarial, and clerical worker´s desk, it is important that purchased software work on any machine that the purchasing user chooses to use at the time. As such, I believe that it is a fundamental mistake for us to restrict software execution to a single machine.

Proposed Solution

After reviewing the goals and restrictions above, I have come up with the following multi-part recommendation for software protection:

1. All application diskettes requiring protection should be made uncopyable or very difficult to copy by the average user (average user defined as a non-technical professional individual).

2. The above protection technique (or a similar process) should be made available to vendors and/or users in order to allow

them to render their diskettes uncopyable. An operating system utility to do this seems to make the most sense here.

3. If the user´s diskette becomes unreadable, then the user should be able to exchange the bad diskette for a new one at the dealer. Note that the user must turn in the bad original diskette in order to obtain a new one.

4. We should implement a 50% discount on multiple copies of software purchased by a user. This discount will discourage users from attempting to copy software since multiple copies will be more reasonably priced.

5. If a customer for some reason gives away or loses his or her application diskettes, then he or she must repurchase the software at the multiple copy discount price. This will encourage people to keep better track of their valuable software and not lend it out.

6. Whenever updating, a customer must trade in the old software volumes for the new ones.

7. Each software product should have a unique registration number associated with it. This number need only be stamped on the diskette and registration card, not encoded in the software. Whenever the user wants an update or support, he or she must state his or her name and number. Only if the name and number given match the registered name and number will the update or service be provided. Such a registration system will discourage customers from even thinking about circulating copies of their applications.

In summary, the plan has three parts: Uncopyable diskettes, multiple copy discounts, and unique registration numbers. I beleive that this plan will satisfy all of the goals and restrictions presented earlier. In addition, it can be implemented at much lower cost and with much less effort than can the hardware and/or software key or system type identification protection schemes discussed so far.

Conclusion

I believe that we must have some solution for the problem of software pirating. But, in reaching a solution, we must not lose sight of both our original goals and the true scope of the problem. I encourage you to present feedback on my recommendations, for I believe that together we can reach a solution that will satisfy all of our needs without overly taxing our resources or putting undue burdens on our legitimate customers.

Inter Office Memo

Date:	June 10, 1980
To:	Distribution
From:	Jim Jatczynski [illegible]
Subject:	VISICALC III Software Protection

Please attend a meeting to discuss the above subject on June 13, Friday, in the Diablo Conference Room, Bandley III, from 1:00 pm to 2:00 pm.

Distribution: Jack MacDonald
Bob Etheredge
Dennis Rieger
Pete Sinclair
Randy Wigginton

FYI: John Couch

1) Randy's scheme

2) Woz's new scheme

Inter Office Memo

Date: June 18, 1980

To: Distribution

From: Jim Jatczynski [illegible]

Subject: Summary of Visicalc III Software Protection Meeting (6/13/80)

The first part of the attached report summarizes the Visicalc III software protection discussion of June 13, 1980.

The second part is an implementation plan based on this discussion. Close cooperation between marketing and engineering will be required to carry out this plan.

There will be a meeting to discuss the above subject Friday, June 20, at 9:00 am in the Yosemite Room, Bandley III, next the cafeteria.

Please note that the contents of the report are Company Private.

Distribution:	John Couch
	Bob Etheredge
	Dick Huston
	Jack MacDonald
	Pete Sinclair
	Dennis Rieger
	Randy Wigginton
	Steve Wozniak

KANE

SUMMARY OF DISCUSSION

Contractual Obligation to Personal Software

Personal Software is to deliver a complete, executable copy of Visicalc /// to Apple. Apple then has a thirty-day acceptance period which may be extended if Apple rejects the product in its current form. Simultaneously, Apple is to carry out a contractual obligation to provide software protection by delivering to Personal Software a computer program capable of making copies of Visicalc that are reasonably protected from unauthorized copying. The contract allots Apple a "reasonable time" after delivery of Visicalc in order to fulfill its obligation to provide the protection method.

The phrase requiring delivery of a computer program to Personal Software implies that they wish to retain flexibility as to who actually produces copies of Visicalc. However, the nature of the selected protection method may make it desirable for Apple to produce copies for Personal Software. Therefore, we might have to modify the contract.

Protection Methods

We discussed two copy protection schemes, one developed by Randy Wiggington and the other by Steve Wozniak.

Randy's scheme works as follows:

[1] Modify selected diskette sectors so that a checksum error will occur if the normal disk read routine is used.

[2] Modify the application program so that it dynamically installs modifications in the disk read routine that allow it to read the altered disk sectors.

Standard copy routines are unable to copy protected diskettes because they use the normal disk read routine which is unable to read the modified sectors without returning an error indication.

WOZ's scheme is based on the observation that there are four unused bits in every disk sector. The standard disk write routines set these bits arbitrarily (to 0's), and the standard read routines ignore them. The scheme works as follows:

[1] Modify selected sectors on the protected diskette so that the four normally unused bits are set to a function f of the remaining bits in the sector.

[2] Modify the application so that during initialization (and at other times during execution, if desired) it access one or more of the modified sectors and assures itself that the four normally unused bits are correctly set to f(remaining bits). It the bits are not set properly, the application aborts itself or performs some other appropriate action.

Diskettes protected in this way can be copied using standard copy routines, but copied diskettes will not operate properly since the application will find that sectors that should have been modified are not. Production copies can be made with the standard 16-sector Dysan copy program.

WOZ proposed a second protection method, but informed me on June'16 that it does not work.

Selected Protection Method

We determined that WOZ's method is probably easier to implement in the time available and easier to maintain in the long run for the following reasons:

[1] No changes need be made to the low level disk routines which are a rather esoteric portion of the system.

[2] The application interface to the protection scheme can be implemented using one new SOS call.

[3] The protection scheme is alterable from one application to another by changing the function f.

[4] As long as the disk data format and the new SOS protection call remain the same, the method is relatively insensitive to other changes in the operating system and in the application.

IMPLEMENTATION PLAN

This section identifies the tasks that have to be performed in order to implement the selected protection method for Visicalc III. Dependencies and assumptions are stated where necessary, but no schedule is given.

Tasks:

[1] Add the following system call to SOS:

```
     CHECK_PROTECTION (input DEVICE NUMBER,
                       input SECTOR_NUMBER,
                       input KEY
                      )
```

DEVICE_NUMBER identifies the device containing the protected diskette.

SECTOR_NUMBER identifies the sector to be checked as required by WOZ's protection method.

KEY is a bit pattern that will be XORed with the data in the sector as part of the process of computing the function f of the sector's contents.

CHECK PROTECTION will read the specified sector from the specified device, XOR the raw data with the KEY (repeating the key as many times as necessary to XOR all of the data), XOR the resulting modified data in four-bit groups, and finally compare this result to the four normally unused bits. If the value of f matches the four unused bits, SOS returns normally. Otherwise, it returns an error.

We probably want to document this SOS call only in internal documents in order to avoid providing clues for breaking the method.

[2] Modify the Visicalc source to make SOS CHECK PROTECTION calls at appropriate points. In order to do this, we will need a Visicalc release diskette, the associated source listing, and any associated documentation. We will need to determine 1) which disk sectors should be modified, 2) what KEY value should be assigned, and 3) where SOS protection calls should be made.

[3] Translate the modified source into object form. Depending on the source language, we may need help from Personal Software to do this.

[4] Modify the selected sectors on the new object diskette in accordance with the selected key. This may require us to write a modified diskette write routine.

[5] Return the protected master diskette to Personal Software. Assuming they can make literal copies of the diskette, including the four normally unused bits in each modified sector, Apple need not participate further in the process. However, if they cannot make literal copies, we may need to modify the contract and sell our copying service to them, since the 16-sector Dysan copy routine can make the required copies.

Alternatively, we could proceed as follows:

[1'] Make the SOS changes described in [1].

[2'] Provide external documentation of the SOS CHECK_PROTECTION call to Personal Software. Require them to 1) select the disk sectors to be modified, 2) select the KEY, 3) install the SOS CHECK_PROTECTION calls as required, and 4) provide a modified object diskette, a list of sectors to be modified, and a KEY to Apple.

[3'] Apple modifies the selected sectors and returns the modified diskette to Personal Software.

[4'] The same considerations about making production copies apply.

The first method is more attuned to the sense of our contract in that Apple makes all modifications necessary to install protection. On the other hand, the second method is probably more efficient in that Personal Software modifies its own software. It also provides an additional measure of security for Personal Software in that only they know without an extensive search where the SOS protection calls are installed.

ACTION TO BE TAKEN

First, we need to determine which of the proposed implementation methods or combination thereof is acceptable to ourselves and Personal Software, particularly because neither adheres to the letter of the contract.,

Second, we need to schedule the implementation of the method we decide to use.

Finally, we need to do it.

Inter Office Memo

Date: June 20, 1980

To: Distribution

From: Tupper Snook

Subject: Results of Disk Protection Meeting, June 18

John Couch promised that:

- John Arkley will s-safe every DOS product in the first release of the Catalog.
- Randy Wigginton will provide protection on the level used for Apple Stellar Invaders for any Pascal programs appearing in the first Catalog.
- The dependancy of s-safe on the auto-boot ROM is a decision to be made by the Catalog group.
- There is no protection available to stop the user from stealing a program from memory.

TS:lvh TS

Distribution:

A. Agrella
J. Arkley
J. Couch
M. Kane
J. MacDonald
R. Wigginton

6/20/80

(AR) Sector read / write on 3.3 copies bits

Randy objected to Woz's scheme because

- 13 sector copy could be modified to 16 sector operation and could then copy protected diskettes; someone at Microsoft is doing this now.

- Pascal programs do not call ~~a~~ SOS directly. ~~cannot have~~ ~~[illegible]~~ If protection check were part of a SOS call, these Pascal applications could not make use of it.

- Prefers to have nibbles encrypted

6/25/80

16 sector products:
- 3.3 Master
- DOS Toolkit (back in NPR)
- Education II

(Woz) Change index initialization

- Broken
  - LOW LEVEL | - Sector at a time
  - LOW LEVEL | - Existing Pascal system; ~~copying~~ ~~to~~ calling Pascal core routines, making ASM program
  - HIGH LEVEL - Finding where key is checked + bypassing

(Huston) Won't be able to copy to Twiggy or Pippin
Economic justification wanted.

Example (Sinclair)
CP/M 40% bootleged w/o protection
After changing protection mechanism, reduced to less than 5%

Decision (Couch)
- Will make one byte change to DOS 3.3 and resubmit to NPR ~~[illegible]~~ for signoff ("is there

won't be an NPR")
- Toolkit will be fixed also since it is back in NPR for APA fixes
- Other products containing DOS 3.3, which are currently in some state of production, will be updated ASAP

1. What products include DOS 3.3?

2. Does Pascal COPY copy hidden bits?

Meeting Wed @ 11