

RECON/Guard Confidence Tests

(dtd 3 January 1983)

Attachment A

Introduction

The purpose of this document is to identify the objectives, methods, and acceptable results of the RECON Guard program acceptance test phase.

The tests outlined here will be conducted by government personnel. Many of the tests will utilize a test Trap-Door program. This Trap-Door program will be generated by OCR/IAB and is specified in this document.

RECON Guard System Acceptance Tests

(Overview)

Objective:

The objective of the Government acceptance tests is the determination of the Guard System's ability to perform its security functions in the presence of normal and abnormal perturbation of the expected/standard operating environment.

Security Vulnerability Determination:

- . Operational tests
 - incorrect key insertion/removal
 - induced errors on on-line and update Guard I/O modules (i.e., can record incorrectly written on output tape of update Guard be inadvertently released?)
 - establish ideal setting for variable Guard System parameter
- . Guard System outage tests
 - random stop/start during record transmission
 - other induced Guard System errors
- . Covert signalling channel tests attempt to transmit data by:
 - modulation of handshake sequence
 - modulation of error messages
 - modulation of legitimate traffic (intact RECON records)
 - alteration of record content
- . Composite tests

All tests will be conducted by or in the presence of designated Government personnel.

Each test or set of tests will be documented prior to the Guard System test subphase. This documentation will include the following:

- . objective
- . method
- . acceptable results

System Outage Testing/Operational Testing

Objective:

The objective of these tests will be to determine the reliability and stability of the Guard System under normal and abnormal/hostile conditions.

Method:

The operational tests will be ongoing during the entire acceptance testing phase. These operational tests will include such actions as would be encountered during operation of the Guard System. These actions will include the following:

1. Random unplugging of On-Line and Up-Date Guard.
2. Random switching/removal of AGB's
 - a. during operation periods
 - b. between operation periods
3. Disruption of Security Officer Interface system
 - a. by random/intermittent power-down
 - b. incorrect keyboard entries
 - c. etc.
4. Inducement of tape errors for Up-Date Guard.
5. Other disruptive events.

Successful Results:

The Guard System should continue its security function under all circumstances. Regardless of the disruptive tests applied, the Guard System should not permit any unauthorized release of information.

Hardware Component Verification Test

Objective:

The objective of this test is to ensure, as thoroughly as possible, that no clandestine hardware (i.e., tap devices) is included in the Guard System hardware set. This hardware set includes the On-Line Guard SBC's, the Up-Date Guard SBC's, all AGB's, and Security Officer Interface hardware.

Method:

The method applied for this test will be visual inspection. More elaborate methods (i.e., IR) will be possible only if an appropriate system is available. Inspection of microprocessor chips will entail magnification and comparison (visually) with proper chip layout description.

Successful Result:

A one-to-one comparison of SBC components and, where possible, chip layout inspection for computer-on-chip components will be deemed acceptable.

Handshake Sequence Modulation

Objective:

The objective of this test will be the transmission of clandestine information by selective interrupt of HAS-Guard and RECON-Guard software communications sequence in a codable repetitive manner. The communication sequence will be interrupted (error condition) in such a way as to encode the test character string.

Method:

The handshake interrupt will be attempted both from the 370 (RECON System) operator console and (if practical) from small embedded in RECON trap door software routines. Modulation of error interrupts will use the following techniques:

1. time lapse between error interrupts
2. device interrupt error type
3. # interrupts/time period
4. use of CANCEL and STATUS acknowledge

The receiving station for this test will be on the outbound (HAS) side of the On-Line Guard device.

Acceptable Results:

The On-Line Guard device should abort all transmission (execute a channel shutdown), prior to the receipt of the full test sequence at the outbound receiving station.

Error Message Modulation

Objective:

The objective of this test is to use error messages to transmit clandestine information through the On-Line Guard device to the COINS network.

Method:

The Trap-Door program will be used to generate and transmit error messages to the COINS network. These error messages will be transmitted via the operator control mode of the Trap-Door program. A predetermined test character sequence will be the object of this method.

Acceptable Results:

The On-Line Guard device should terminate channel activity prior to output of the full test character sequence.

Alteration of Records

Objective:

The objective of this test will be the transmission of clandestine information through the On-Line Guard device by overlaying such information on legitimate outbound RECON records.

Method:

The Trap-Door program will insert predetermined character strings into authenticated RECON records. These records will then be transmitted to the On-Line Guard device for output to the COINS network.

Acceptable Results:

The On-Line Guard device should detect all such unauthorized alterations of the legitimate RECON records. The On-Line Guard should then block transmission of the altered records and inform the audit and alarm devices.

Legitimate Traffic Modulation

Objective:

The objective of this test will be the transmission of a designated character string by the use of properly authenticated records. This test will attempt to open a covert signaling channel with legitimate message traffic.

Method:

Transmit 26 properly authenticated, "correct" records to HAS. Then, as part of the same response retransmit the 25th, 15th, 21st, 8th, 1st, 22nd, 5th, 2nd, 5th, 5th, 14th, 8th, 1st, 4th, then end of response. If possible, the first 26 records should be the shortest available.

Successful Result:

The Guard will immediately recognize a direct channel attack and shut down on the receipt of the 27th record. (Actually, if it did, it would be in error of some kind.)

Also, allowable response set limit, or retransmission limit should be exceeded and cause Guard cutoff.

RECON Test (Trap-Door) Program

Objective:

The objective of the RECON acceptance test Trap-Door program will be the introduction of security relevant errors and perturbation into the Guard-RECON-HAS function. The Trap-Door program should allow the 370 system (RECON host) operator to control error condition generation from the control console. Also, the Trap-Door program should have the capability to automatically generate RECON system errors.

Function:

1. Change contents of RECON record by the following methods:
 - a. single bit change in RECON record
 - b. overlay of character string in RECON record text
2. Transmit selected error messages by:
 - a. insertion in response record set
 - b. direction from operator console
3. Transmit response record set
 - a. response record set should include altered records
 - b. response record set transmission should be under console operator control
4. Automatic execution mode
 - a. program should automatically loop through any combination of above functions
 - b. selected functions will be determined by operator at initiation of automatic execution mode
5. Simulate RECON system outages
 - a. transmit CANCEL messages under operator control
 - b. transmit CANCEL messages via automatic loop

- c. transmit status acknowledgement messages under operator control
- d. transmit status acknowledgement via automatic loop

The RECON test program should have a control structure which permits execution of all five of the above functions by two separate modes. These control modes are:

1. Automatic function execution mode
2. Operator-controlled function execution mode

Automatic Function execution mode will be RECON test program initiation of operator-selected functions in a loop method. Execution will be stopped by a preset timer, loop parameter, or operator intervention. The operational scenario will be as follows:

- a. operator parameter entries
 - (1) functions to be executed
 - (a) order of execution
 - (b) random order of execution
 - (2) stop parameters
 - (a) time
 - (b) # of execution loops
 - (c) infinite loop (stopped by operator intervention)

Operator-controlled function execution mode will be RECON test program execution of specific operator-selected functions. The operational scenario will be as follows:

- a. operator parameter entries
 - (1) functions to be executed
 - (2) # of executions

COMMUNITY RECON ERROR GENERATOR

This paper outlines the specifications of an error generating program developed as part of an effort to test the ability of the RECON GUARD device to detect and flag RECON records which have been altered since the assignment of a 'CHECKSUM' (by the UPDATE GUARD device). The program operates in an IBM batch environment and is written in the IBM ASSEMBLER language. The purpose of the program is to intercept and alter a COMMUNITY RECON output file prior to being received by an ONLINE GUARD device for channeling back to the COMMUNITY RECON user.

The program has been designed to be inserted into the COMMUNITY RECON job stream after the final output data set has been created but before sending the output to the ONLINE GUARD device. Through the use of Control card input the program can alter any or all of five (5) output records. The records that can be affected are:

- 1) the FIRST-CARD
- 2) the 1st RECON record
- 3) the MIDDLE RECON record
- 4) the LAST RECON record
- 5) the LAST-CARD

The program accepts any of 7 Control card formats in any order. The number of Control cards has been limited (arbitrarily) to 20. Each type of format can be used any number of times (limited to a total of 20 Control cards) and the same record can be altered with several different Control cards. The 7 general 'commands' are:

- 1) USERID -- change the userid on the FIRST-CARD
- 2) AGENCY -- change the originating agency on the FIRST-CARD
- 3) REQNUM -- change the 4 digit request number on the FIRST-CARD and LAST-CARD
- 4) BIT -- flip a single bit in the RECON record
- 5) BYTE -- alter a character in RECON record to EBCDIC 'FF'
- 6) STRING -- Add, Replace or Delete a string (up to 20 characters)
- 7) ORDER -- Place copies of specified RECON records from the set after the last legitimate RECON record but before the LAST-CARD.

UNCLASSIFIED

For the purpose of RECON record manipulations , 23 subfields were selected to provide the capability for altering different parts of the RECON records. These subfields were assigned 2-6 character acronyms for use on the Control Cards. These 'field specifications' give the ability to select particular fields. Of particular interest for this test are the D1, D2 and CW fields which affect dissemination and the CS field which affects the CHECKSUM itself. The complete list of allowable field acronyms follows.

DF	--	document format
DNSB1	--	" " number subfield 1
DNSB2	--	" " " " 2
DNSB3A	--	" " " " 3
DNSB3B	--	" " " " 3
ITEM	--	" " " "
PAGE	--	" " " "
DT	--	" type
CL	--	classification
D1	--	dissemination 1
D2	--	dissemination 2
CW	--	code word
PD	--	publication date
EX	--	exemption category
TD	--	to date
PR	--	processing date
MG	--	management field
SN	--	series periodic set
SC	--	subject code in subject periodic set
CY	--	country code in " " " "
KW	--	keywords
TI	--	title
CS	--	checksum

UNCLASSIFIED

The format of the Control cards is as follows.

COMMAND	COLUMNS	CONTENTS
USERID	1-6	USERID
	7	-
	8-22	new userid
AGENCY	1-6	AGENCY
	7	-
	8-11	new agency
REQNUM	1-6	REQNUM
	7	-
	8-11	new request number
BIT	1-6	BIT
	7	F or M or L (first, middle or last)
	8-13	field specification (acronym for which part of the RECON record to alter)
BYTE	1-6	BYTE
	7	F or M or L
	8-13	field specification
STRING	1-6	STRING
	7	F or M or L
	8-13	field specification
	14	A or D or R (add,delete or replace)
	15-34	string to add,replace with or delete (in case of delete the contents of the string are ignored -- the # of characters in the string will be the number of characters deleted from the RECON record)
	--OR--	
	BLANK	(set field to blanks)
	--OR--	
	ZERO	(set field to binary zeros)
ORDER	1-5	ORDER
	7	-
	8-80	continuous sequence of 2 digit numbers

UNCLASSIFIED

(max 36) representing an ordering of RECON records from the output set to be placed after the legitimate records but prior to the LAST-CARD.

(ex. 20051920 -- copy the 20th RECON record in the set and place it after the last RECON record in the set. Follow this with the 5th, 19th and 20th records (spelling the word TEST).)

Theoretically the program can handle any file size passed in. For practical purposes however, an input file size limit of 1000 records should be observed. The program will output the altered records to a data set picked up by the spanned record formatter Utility. It will also output a hardcopy count of processed control cards and any error messages that were generated due to improper Control cards.

UNCLASSIFIED

Notes on Legitimate Traffic Modulation

STAT



29 January 1983

Introduction

Legitimate Traffic Modulation is one of the most formidable security problems encountered when data bases containing multi-level security and compartmented information is exposed (connected) to untrusted network access. At first glance, an LTM type clandestine attack would appear to offer an unlimited, uncontrollable bandwidth, since LTM involves the transmission of legal output but in a way as to encode information.

All covert signalling/timing channels can be fully exploited if tradeoffs on normal system/network operations and throughput are prohibited. The following discussion will suggest an approach to severely limiting the LTM available bandwidth, while defining the operational constraints on the target system.

This note contains a brief discussion of Legitimate Traffic Modulation (LTM). The techniques could be employed for any new remotely accessed data bases where security is of prime concern. With some enhancement of the Guard system the initial scenarios could be employed for existing data bases. The sequence proof (encrypted data base) scenarios could not be applied to existing systems without extensive modification.

The sequence proof scenarios would require a verified, off-line, data base encryption peripheral. The Guard Device prototype design could possibly serve this function with minor modification. Properly accessed encrypted data bases may be a solution to LTM.

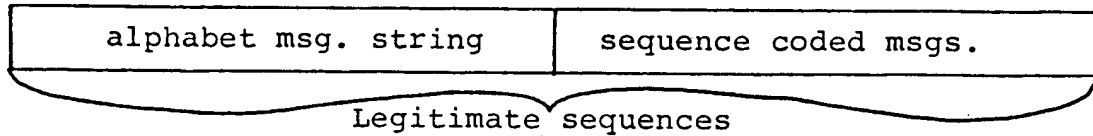
(3) patterns to be detected

- (a) alphabet strings: with unique msg. types
- (b) trailer strings: all previously encountered msg. types
- (c) buffer deleted when (above) clandestine code probabilities are encountered

4. Buffered response set will be scrambled

- a. order of response msgs. will be changed randomly
- b. order of response msgs. will be changed by pre-determined algorithms
- c. this appears to be most effective LTM countermeasure
 - (1) ease of implementation
 - (2) effectiveness
 - (3) can be standard output sequence function not requiring code ident (as in (3) above)

Legitimate Traffic Modulation (LTM)



A. Attack requirements:

1. sequence must be pre-ordered and known by receiver
(attack method is sequence integrity dependent)

B. Defenses:

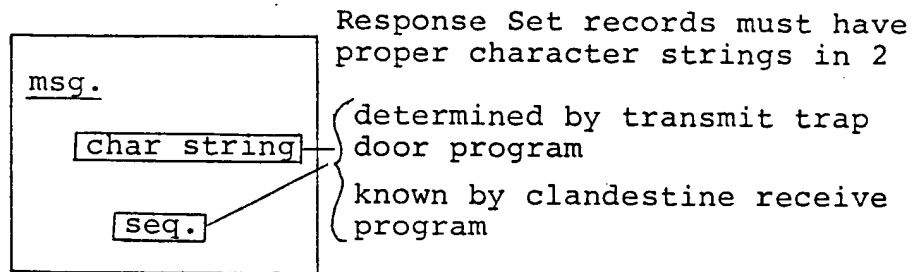
1. limit length of response sequence
(reduce channel bandwidth)
2. LTM detection
 - a. property of LTM attack
 - (1) must have nonunique msg. types in trailer portion of response set, or
 - (2) must have known spacings of unique msg. types in trailer portion
3. detector function
 - a. the principal detector function will be code detection
 - (1) various methods exist for this
 - b. method
 - (1) entire response msg set must be buffered prior to output
 - (2) alphabet string detection
 - (a) each 1 to n string of msgs. will be examined with the resultant trailer sequence

(Sequence Proof) LTM Channel

A. Trap door program function

From the set of responses

1. scan response set for records with
 - a. appropriate characters in predetermined string field
 - b. appropriate characters in predetermined sequence character field
2. transmit related records (this activates LTM channel)



3. receive program decodes clandestine message from the received response set

B. Constraints on defensive measures

1. Must not disrupt communication link throughput rate
2. Must not decrease response set received

I. Solution

A. Requirements for attack:

1. trap door program must know contents of records which will be transmitted
2. location of data and sequence information fields and/or pointers must be known to transmitter and receiver
3. trap door program must select, from response set, those records having proper characters in the predetermined data and sequence fields

B. General solution:

1. encrypted data base with header and trailer information stripped on output (trap door cannot know information released)
2. limit response set size

C. Solution (using Guard concept) when query capability is required:

1. (a) queries could search on header information only (header would contain key words for query search). Thus severely restricting the amount of characters usable by a trap door program
- (b) or incoming query (search parameters) encrypted with appropriate key (or single key if 1 key scenario is used)
- (c) thus query can only search/hit on appropriate records (if multi-key scenario is used)

2. Scenario:

- Enter:
1. auth and save store encryption results and auth (at update device)
 2. now no way to tell what output will be from internal trap door. Thus trap door cannot select sequence or character string of response set for output because it cannot know what will be output

- Exit:
1. decrypt at (online device)
 2. authenticate as before
 3. scramble response set
 4. Guard determines destination and strips headers and trailers (only encrypted text is passed after decryption by Guard)
 5. Guard checks contents of response set (i.e., response set must contain more than 1 record, and response set must contain no identical records)

Conclusions

As discussed in part C.2, above, an encrypted data base will prevent a trap door program from knowing which data is transmitted. This is because encryption/decryption of text is done off line (outside of operating space of the data base or the target system). No plain text data arriving at the Guard is transmitted (headers are stripped). No encryption data is transmitted (encrypted text changed to plain text at Guard before transmission). Thus the Guard also functions as an encryption device. Although not applicable to the present RECON system, new data bases employing this scenario with a Guard Device concept may be protectable in this manner.

This communication suggests an approach to controlling a covert signalling channel utilizing LTM. The proper use of encryption and response record sequence sampling on the bandwidth of a LTM type attack can be severely limited. It is hoped that the above discussion can stimulate dialog leading to more complete solutions to the LTM problem.

OS/ISSG
Draft Evaluation Report

Attachment B

CONFIDENTIAL

~~CONFIDENTIAL~~

ROUTING AND RECORD SHEET

SUBJECT: (Optional)

Testing of the RECON GUARD Prototype

FROM:

NO.

DATE

5 JUL 1984

TO: (Officer designation, room number, and building)

DATE

OFFICER'S INITIALS

COMMENTS (Number each comment to show from whom to whom. Draw a line across column after each comment.)

RECEIVED FORWARDED

1. RECON Project Officer
PATG/ISR/ORD
726 Ames Building

7/11

Cwk

Chuck,

Attached is the RECON GUARD Prototype Test Report, which I showed you last week. I've sent a copy of this to [redacted] as a draft.

I intend to submit this same memo to go from the Director of Security to [redacted] and Claire Rice, therefore, please consider this official between you and me but not the official OS response.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

~~CONFIDENTIAL~~

29 JUN 1984

MEMORANDUM FOR: RECON Project Officer
Processing and Analysis Technology Group,
Information Systems Research Division,
Office of Research and Development

25X1 FROM:

[Redacted]
Industrial Systems Branch,
Information Systems Security Group, OS

25X1 SUBJECT:

Testing of the RECON GUARD Prototype [Redacted]

1. Between 14 May and 11 June 1984, Information Systems Security Group performed testing of the RECON GUARD prototype hardware which was developed by Sytek, Incorporated, under contract to the Office of Research and Development. The objective of this Project was to demonstrate the feasibility of the GUARD-Device concept and its applicability to the problem of connecting classified Agency data bases to external networks. Preliminary examination of the test results, indicate that the GUARD performs this function. I recommend, therefore, that the GUARD Device be certified as having passed the tests described in attachment A. [Redacted]

2. Testing was performed using the Guard Test System (GTS) which emulated a Host Access System (HAS). Use of the GTS allowed for easier manipulation and verification of the data than if using a live COINS HAS. Additionally, testing was performed using only a 2,000 record test RECON data base. It is most important to note that extensive testing with a live network and a live data base must take place before a GUARD-Device can hold the trusted position of protecting classified Agency data from unauthorized exposure to an external network. [Redacted]

3. It is also noted that the GUARD cannot control what happens to data before data enters RECON via the Update Guard, while data is scored in the RECON data base, or after data is released from RECON via the On-line Guard. [Redacted]

4. The boundary of effect of the GUARD system is defined to be these entry and exit points. There are currently no known Trojan Horses or trap-doors in the MVS, JESS or RECON systems. Since RECON GUARD, as it is presently configured does not address the Trojan Horse issue, the GUARD system must be assessed on this basis. [Redacted]

~~CONFIDENTIAL~~

CONFIDENTIAL

5. Testing was intended to address the trustworthiness of the GUARD as a gateway to allow the release of authorized data and to prevent the release of unauthorized data. It did not, for instance, consider issues such as throughput.

25X1

6. All currently planned testing has been accomplished.

25X1
25X1

Attachments:

- A. RECON GUARD Test Plan (Attached)
- B. RECON GUARD Test Results (Pending)

APPROVED:

25X1

Chief, Information Systems Security Group

5 July 1984
Date

CONFIDENTIAL

~~CONFIDENTIAL~~

RECON GUARD TEST PLAN

The tests outlined in the following pages were performed based upon several assumptions. The GUARD must be operated within these constraints for the test results to remain valid. These assumptions are as follows:

- ° One AGB¹ is dedicated to each site for this test (i.e., State, DIA, and NSA).
- ° Individuals at each site are cleared system high and identified to the network; terminals are physically located in system high approved areas.
- ° Sufficient physical and personnel security standards for system high processing are afforded to the GUARD hardware and the Security Officer Interface Device (SOID).
- ° The GUARD cannot control what happens to data before data enters RECON via the Update Guard, while data is stored in the RECON data base, or after data is released from RECON via the Online Guard. The boundary of effect of the GUARD system is defined to be these entry and exit points.
- ° Testing was performed using identical secret keys for the multiple sites.
- ° Testing was performed using the Guard Test System (GTS) which emulated a Host Access System (HAS). Use of the GTS allowed for easier manipulation and verification of the data than if using a live COINS HAS.
- ° A Trap-Door program was utilized to allow for limited data manipulation of RECON records. Releasable RECON records could probably be created within the RECON data base given the knowledge of the appropriate secret key, sufficient time, systems knowledge, and accessibility.
- ° Testing was performed using only a 2,000 record test RECON Data base.
- ° There are currently no known Trojan Horses or trap-doors in the MVS, JES3 or RECON systems. Since the GUARD as it is presently configured, does not address the Trojan Horse issue, the GUARD system must be assessed on this basis.

¹ See the glossary of definitions on the following page.

~~CONFIDENTIAL~~

The following definitions will be used throughout this document:

- ˆ A record or a RECON record is defined to be a collection of data which meets the format requirements for entrance to RECON, or such a collection of data after it has been marked for distribution to specified communities.
- ˆ A secret key or key and initial value consists of a cryptographic type of numeric key value and initial value used by the DES algorithm to encrypt or decrypt a given data stream. Both the secret key and the category expression reside in an erasable programmable read only memory (EPROM) that is located on one or more Authentication Generator Board (AGB).
- ˆ An authenticator is a digital signature attached to a record. It is either a blank field called a null authenticator or it is a field of the record containing a value calculated for the record with an algorithm such as DES using the record itself and a secret key. Authenticators are used to control release and distribution of RECON records.
- ˆ A category can be identified for a record by specific fields of the record. A boolean function called a category expression is associated with each releasable data set. If the category expression when applied to a given record evaluates to "TRUE," then the authenticator is computed with the secret key associated with that category expression.
- ˆ A record is a candidate for release by the GUARD or the GUARD is authorized to release a record if and only if there is a category expression and secret key within the GUARD which is associated with the record.

EXAMINE CODING OF SOFTWARE

Review and confirm that Theorems and Corollaries are valid in the "GUARD System Verification Report," dated 3 May 1984.

Review and confirm "vmulti pl" master, dated 11 May 1984.

Review and confirm "vupislv pl" (outer block of code that runs on the Update Guard Input Slave Board), dated 11 May 1984.

Review and confirm "vupdate pl" master, dated 11 May 1984.

Review and confirm "vassembly al", dated 11 May 1984.

Review and confirm "vagbmod al", dated 4 January 1984.

Review and confirm "vagb pl" (outer block of code that runs on the AGB), dated 4 January 1984.

Evaluate security relevant material from "Guard System Operation and Maintenance" report, dated 16 April 1984.

Evaluate security relevant material from "RECON IV Users Manual," dated April 1980.

CONFIDENTIAL

VERIFY FUNCTIONAL OPERATION OF GUARD

1. Develop 20 test queries.
2. Run the test queries directly through the RECON Test Data Base without interfacing with the GUARD.
3. Run the test queries through the GUARD System utilizing all possible AGB's and sites (nine runs per query).
4. Compare the results of the test runs to ensure functional operation of the GTS and ensure that improper release or spillage has not taken place.

SECURITY VULNERABILITY DETERMINATION

Operational Tests

1. Incorrect EPROM insertion/removal.
2. Induced errors on the Update Guard (UDG).
 - a. Write to an output tape where the records from the input tape exceed the length of the output tape.
 - b. Invalid record test requiring improperly formatted input tapes.
 - 1) A record too short.
 - 2) A record longer than the UDG buffer (2048 bytes).
 - 3) A record with an incorrect checksum.
 - 4) A record that already has an authenticator.
 - 5) A record with a category expression unknown to the UDG.
 - 6) A record that goes off the end of the input tape.
 - 7) A record with incorrect tape parity.

CONFIDENTIAL

- c. Configuration test illustrating UDG's reaction to changes in slave board configuration and placement.
 - 1) Remove output slave processor board.
 - 2) Start system with drives off load-point and offline.
 - 3) Start system with drives powered off.
- d. Manipulation of alarm system and auditing device connection.
 - 1) Unplug the alarm cable before starting UDG.
 - 2) Unplug the audit cable before starting UDG.
 - 3) Unplug the alarm cable while UDG is running.
 - 4) Unplug the audit cable while UDG is running.
- e. Abnormal AGB data EPROM contents.
 - 1) Improper checksum.
 - 2) Corrupted secret key.
 - 3) Corrupted initial value.
 - 4) Corrupted stock plain text.
 - 5) Corrupted cyphertext.
- f. Abnormal tape contents.
 - 1) A tape without IBM labels.
 - 2) A tape with improper IBM labels.
- g. Abnormal tape procedures.
 - 1) Attempt to combine multiple input tapes given a single output tape.
 - 2) Take drives offline.
 - 3) Alter tape position by hand.
 - 4) Put write ring on input tape.
 - 5) Remove write ring from output tape.

5
CONFIDENTIAL

3. Induced errors on Online Guard (OLG).
 - a. Manipulation of alarm system and auditing device connection.
 - 1) Unplug the alarm cable before starting OLG.
 - 2) Unplug the audit cable before starting OLG.
 - 3) Unplug the alarm cable while OLG is running.
 - 4) Unplug the audit cable while OLG is running.
 - b. Random start, stop and power disruption.

GUARD System Outage Tests

1. Random start/stop during record transmission.
 - a. Power off the protocol converter during record transmission.
 - b. Reset GUARD during record transmission.
 - c. Power off GUARD during record transmission.
2. Random unplugging of Online Guard.
 - a. During record transmission.
 - b. After system configuration but prior to query submission.
3. Random switching/removal of AGB's.
 - a. Removal of AGB during record transmission.
 - b. Insertion of new AGB after system configuration but prior to query submission.
 - c. Insertion of new AGB during record transmission.
4. Disruption of Security Officer Interface Device.
 - a. By random/intermittent power down.
 - b. Incorrect keyboard entries.
 - c. Preparation of EPROM on a EPROM burner which is not a part of the SOID.
5. Inducement of tape errors for Update Guard.
 - a. Tape too short for specified records.
 - b. Random manipulation of alarm, audit device, etc.
 - c. Invalid data.

CONFIDENTIAL

Manipulation of AGB Boards

1. No AGB.
2. Single AGB with multiple category expressions.
3. Single AGB with multiple category expressions and secret keys.
4. Multiple AGB's with same category expression and secret key.
5. Multiple AGB's with different category expressions.
6. Multiple AGB's with same secret key.
7. Multiple AGB's with different secret keys.
8. Random manipulation of AGB's from one site and office/ organization identification from another site.
9. General manipulation of AGB sites, secret keys and release categories.

Trap - Door Program

1. Manipulate data without modifying authenticator.
2. Modify only authenticator.
3. Modify both authenticator and data.
4. Attempt to manipulate data in such a way to achieve same checksum.
5. Attempt to manipulate data in such a way to achieve same authenticator.
6. Modify data, attempt to release, then return data to original condition.
7. Modify data and/or authenticator, determine and append new checksum.
8. Modify data and/or authenticator, determine and append new authenticator.
9. Reauthenticate records in test RECON.
10. Alteration of record content.

CONFIDENTIAL

CONFIDENTIAL

Covert Signalling Channel Tests Attempt to Transmit Data:

1. Modulation of handshake sequence.
 - a. Time lapse between error responses.
 - b. Device Interrupt error.
 - c. Number of interrupts/time period.
 - d. Use of CANCEL and STATUS acknowledge requests.
2. Modulation of error messages through manipulation of records via the trap door program.
3. Modulation of legitimate traffic (intact RECON records).

8
CONFIDENTIAL

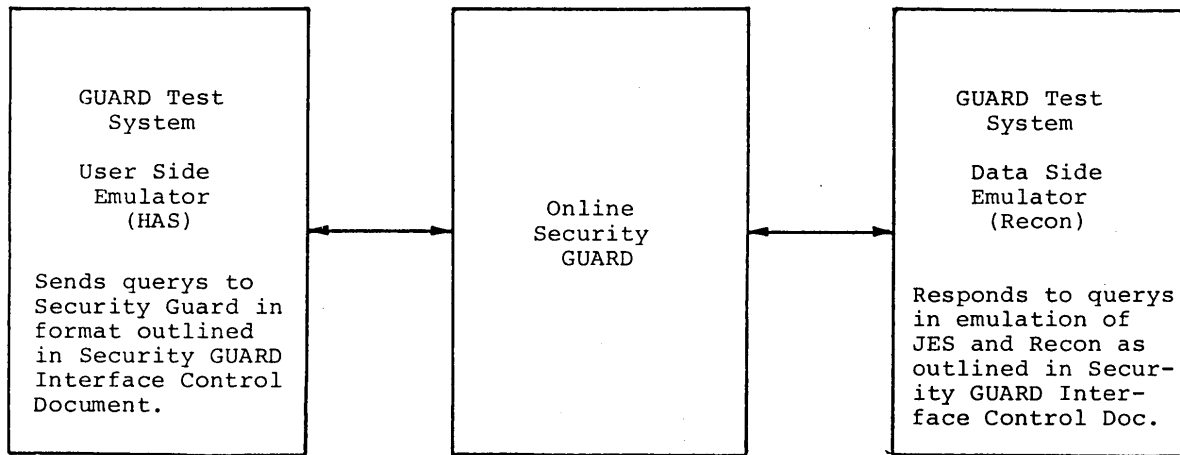
Configuration Diagrams

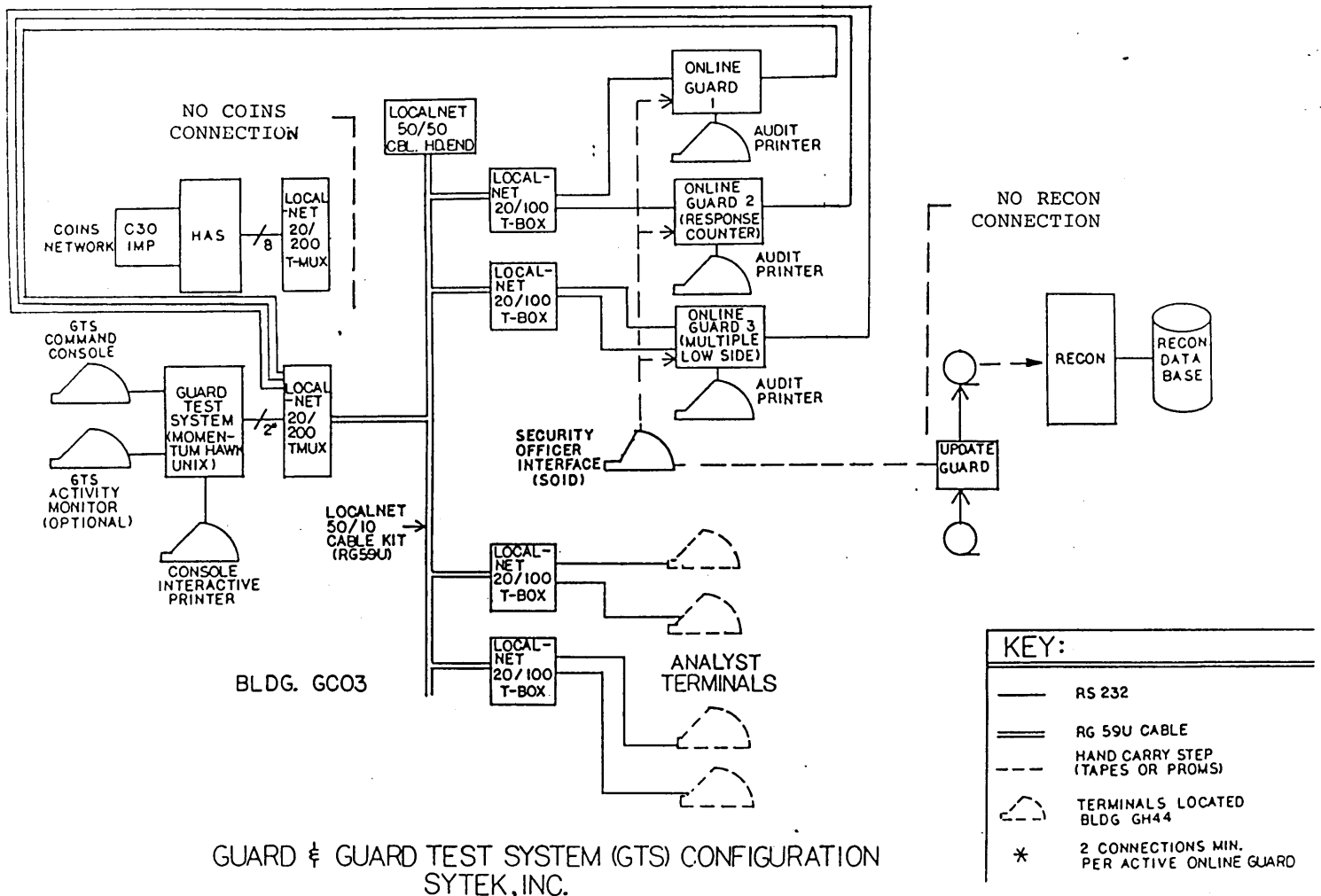
**Guard Test System
(GTS)**

Comprising:

- (a) Momentum Haw RECON/HAS Emulator
- (b) LocalNet System Components

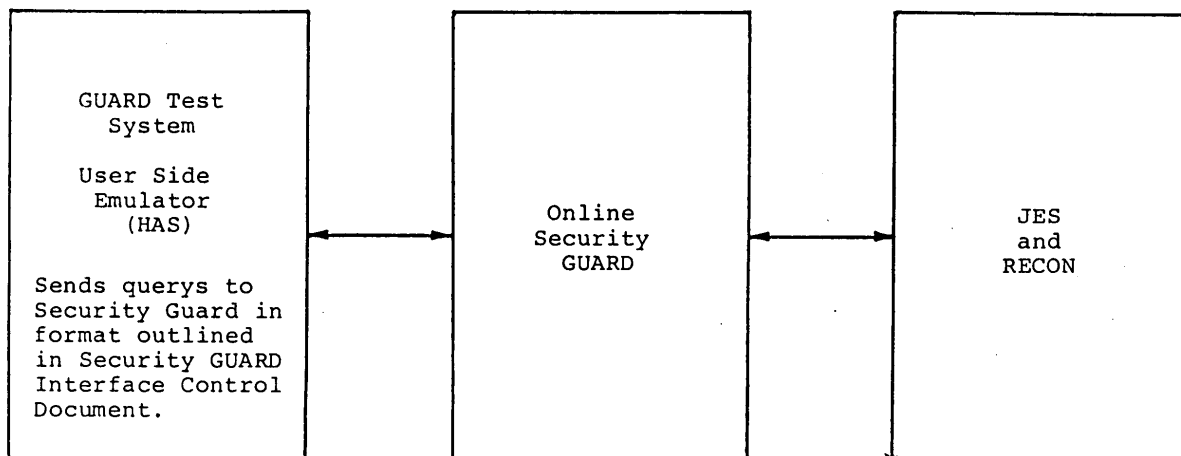
Attachment C



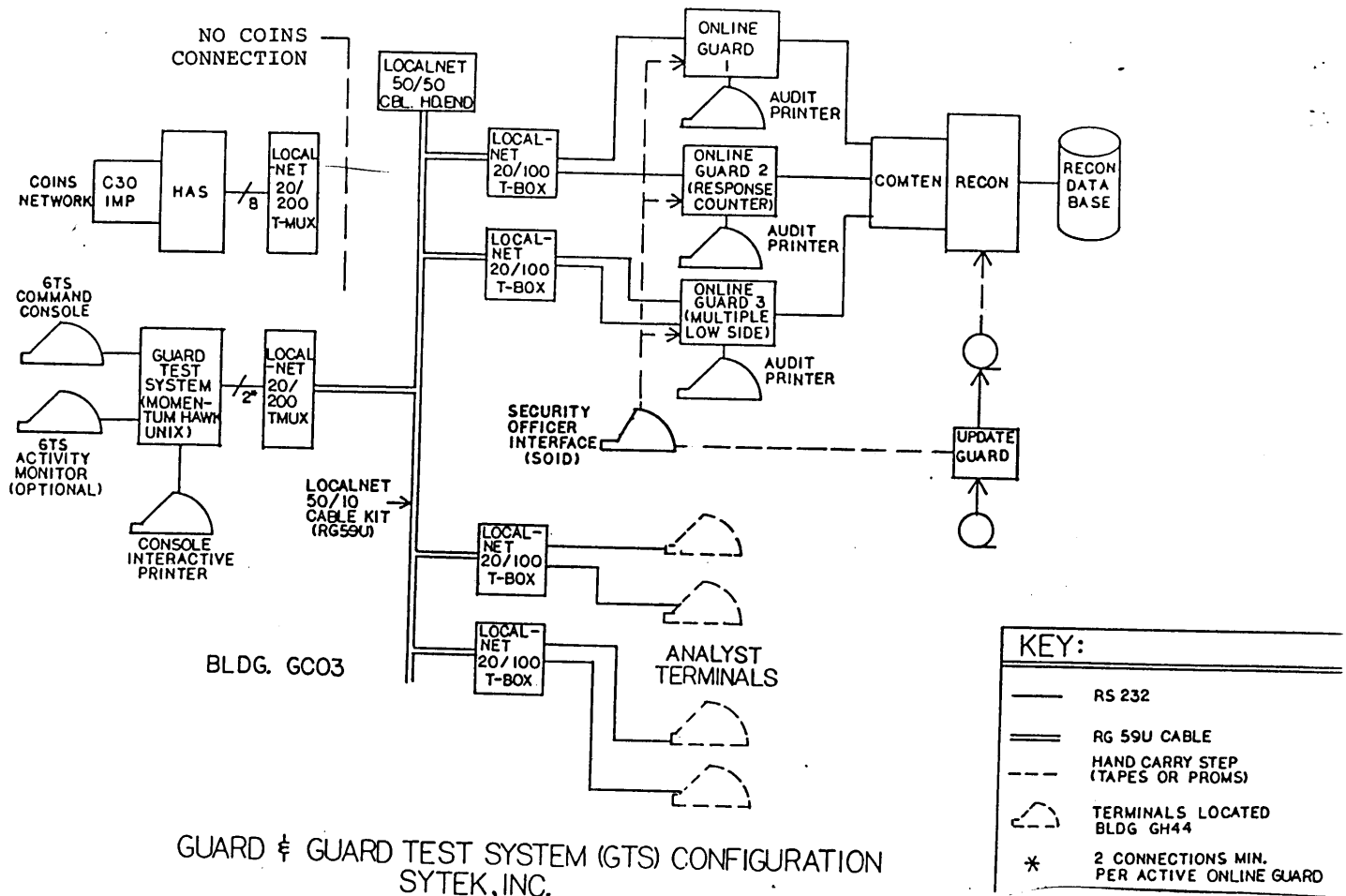


STAND ALONE - Figure 1

Declassified in Part - Sanitized Copy Approved for Release 2012/08/21 : CIA-RDP95-00972R000100100012-7

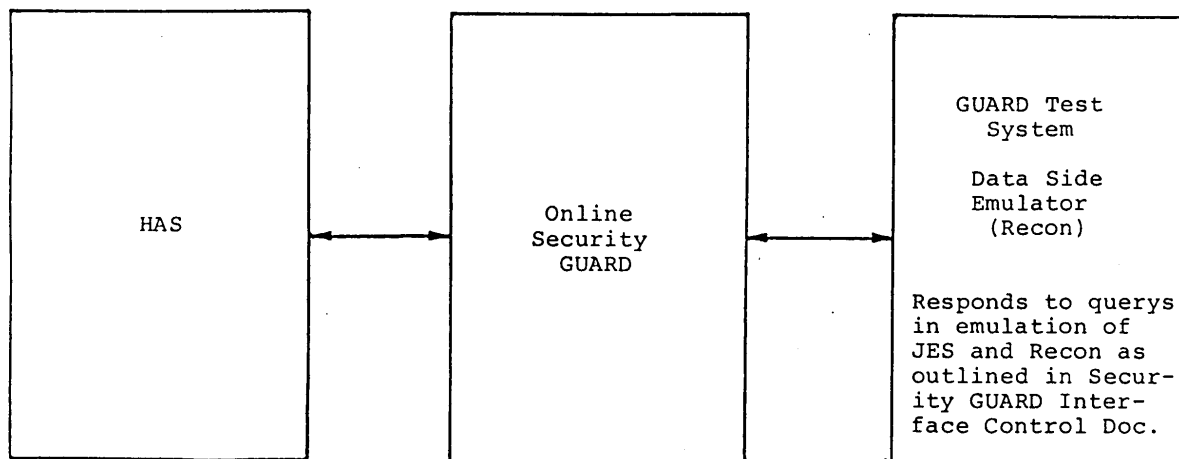


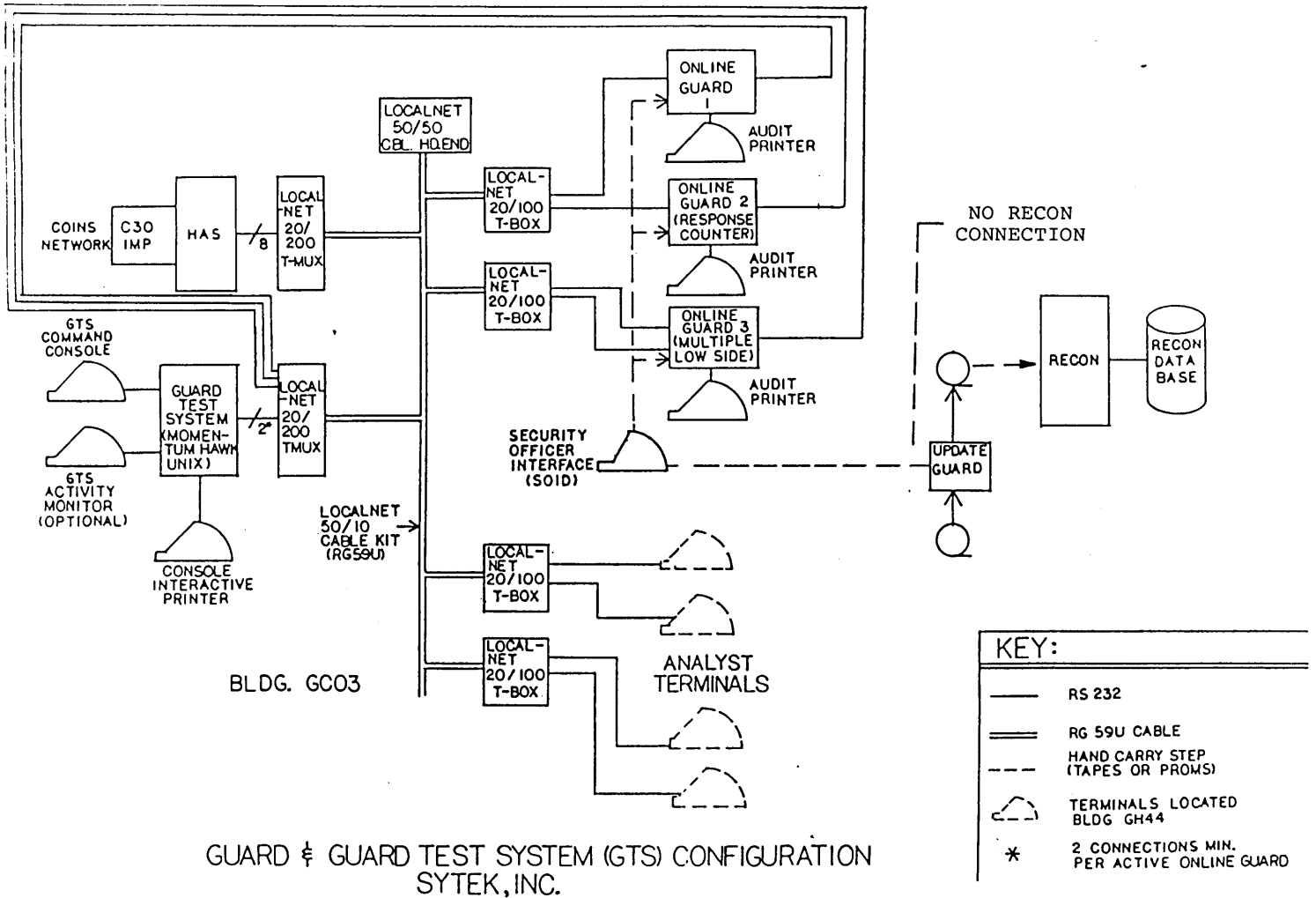
Declassified in Part - Sanitized Copy Approved for Release 2012/08/21 : CIA-RDP95-00972R000100100012-7



GUARD & GUARD TEST SYSTEM (GTS) CONFIGURATION SYTEK, INC.

RECON System Connected, GTS Simulates COINS - Figure 2





COINS Network Connected, GTS Simulates RECON - Figure 3

LocalNet™

**Broadband
Network
Technology**



Linking today with tomorrow.

As a manager with responsibility for data communications decisions, you are faced with three key issues when you evaluate a local area network. The first is growth. Your network must have the capacity to handle your current needs, plus whatever the future may bring—including video and voice transmission, links to other local networks, and gateways to long-distance networks.

The second issue you must face is your sizable investment in existing equipment. The network you select must be compatible with *all* of it.

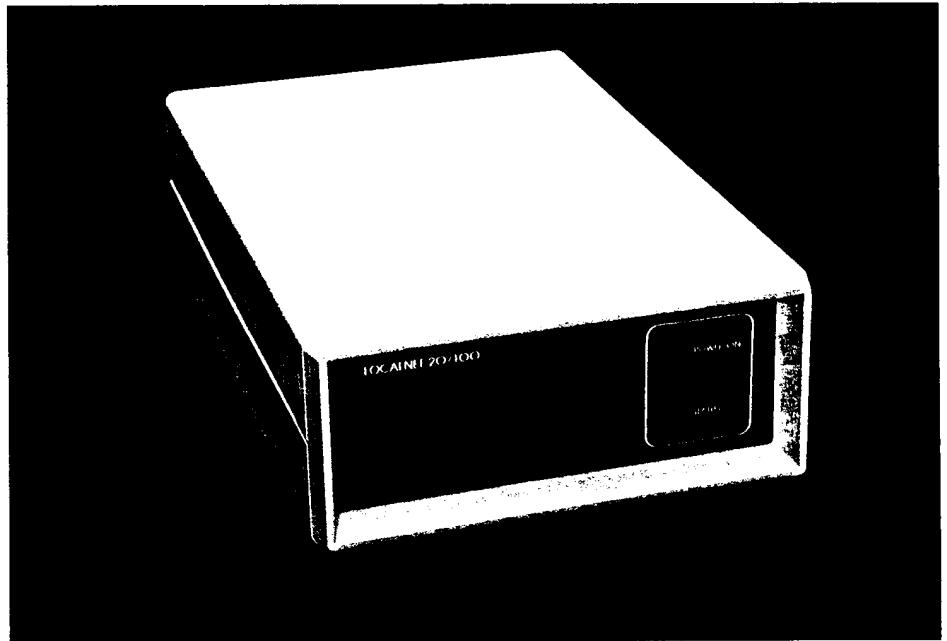
The third issue is the certainty of change. Your network must be able to accommodate whatever the future may bring—whether it be new vendors, new transmission standards, or even new information technologies.

In addition to these fundamental issues, there are other very important considerations—the physical means by which information is transmitted, for example. Systems which use telephone cable or point-to-point coaxial cable can create wiring nightmares that become an enormous barrier to growth—and sometimes actually make growth impossible.

Cost control is another consideration. For some network technologies, the incremental cost of expansion is very high. With leased lines, for example, you have no way to avoid costly rate increases, or even predict them!



LocalNet is totally vendor-independent, so all the devices you own can communicate with one another.



A PCU associated with each LocalNet node handles all intelligent data communications functions.

The LocalNet™ Solution

The Broadband Approach. With LocalNet™, Sytek has created a technology that resolves all these issues, and provides you with a spectrum of cost-effective data communication services—for today *and* tomorrow.

LocalNet is a broadband local area network that can simultaneously accommodate communication between over 20,000 separate user devices (computers, terminals, digital sensors, etc.), all on a single coaxial cable of the type used by the cable television (CATV) industry. Moreover, when LocalNet is operating at its maximum (25.4 Mb/sec), *74% of the cable's capacity is still available for other services, such as video and voice.*

The LocalNet transmission technique is compatible with midsplit, subsplit and dual cable installations; and you can locate nodes anywhere within a radius of 50 kilometers.

LocalNet is totally vendor-independent, which means that *all* the computers, terminals or other devices you now own can communicate with each other; it also means you have complete freedom to select new vendors and new equipment in the future.

LocalNet supports a broad spectrum of devices with widely varying data rates,

from low-speed terminals with serial interfaces (synchronous or asynchronous) to high-speed host computers with parallel interfaces.

Practical Considerations. LocalNet makes planning the distribution of data in a building, college campus, or industrial complex as straightforward as planning the distribution of electricity. Since the coaxial taps into the network are industry-standard and inexpensive, they can be pre-installed to accommodate whatever requirements the future may bring.

Once in place, LocalNet is physically rugged and reliable. Since it uses standard CATV hardware, LocalNet provides proven reliability—and its topology is such that damage to one outlet or cable has no effect on the rest of the network.

Technology

Distributed Intelligence. In LocalNet, the intelligence resides within the network. A Packet Communication Unit (PCU) is associated with each user device or group of devices. The PCU "packetizes" digital input from a user device, and converts it to analog (radio frequency) output for the network. It also handles all intelligent data communications functions, such as network access, formatting, addressing,

and error checking. Since network control or management by a central computer is not required, LocalNet can be implemented without diminishing the capacity of existing computers.

LocalNet PCUs provide a broad range of user services and functions, such as protocol translation and virtual terminal support. LocalNet PCUs implement the upper six layers of the Open Systems Interconnection Reference Model, currently being standardized by the International Standards Organization (ISO). This architectural layering provides for ease of expansion, ease of support, and the standardization of external interfaces.

The primary techniques used by LocalNet to support multiple data streams on a single CATV cable are packet switching, CSMA/CD (Carrier Sense Multiple Access with Collision Detection) and FDM (Frequency Division Multiplexing).

Packet switching allows many PCUs to exist on the same cable, and only consume bandwidth while actually sending data. CSMA/CD is a "listen while talk" contention management technique which allows multiple nodes to efficiently share one frequency channel. FDM permits multiple frequencies on the same cable.

The Products. LocalNet products are divided into three families: The LocalNet 20 family provides economical low speed (128 kb/sec) networking for devices with serial interfaces, both synchronous and asynchronous. The LocalNet 40 family is designed for relatively high speed (2Mb/sec) devices with parallel interfaces.

LocalNet 50 products implement optional higher level functions such as access control, monitoring, failure isolation, and security. LocalNet 50 products also handle inter-network linking functions and long-distance gateways. Products from all three families can co-reside on the same cable, along with a broad spectrum of other services.

The Company

About Sytek. Sytek offers a wide variety of network-related services, including the specification of local area networks, system architecture design, and prime contracting for turnkey local area network installations. In essence, Sytek's level of involvement can vary according to your individual requirements.

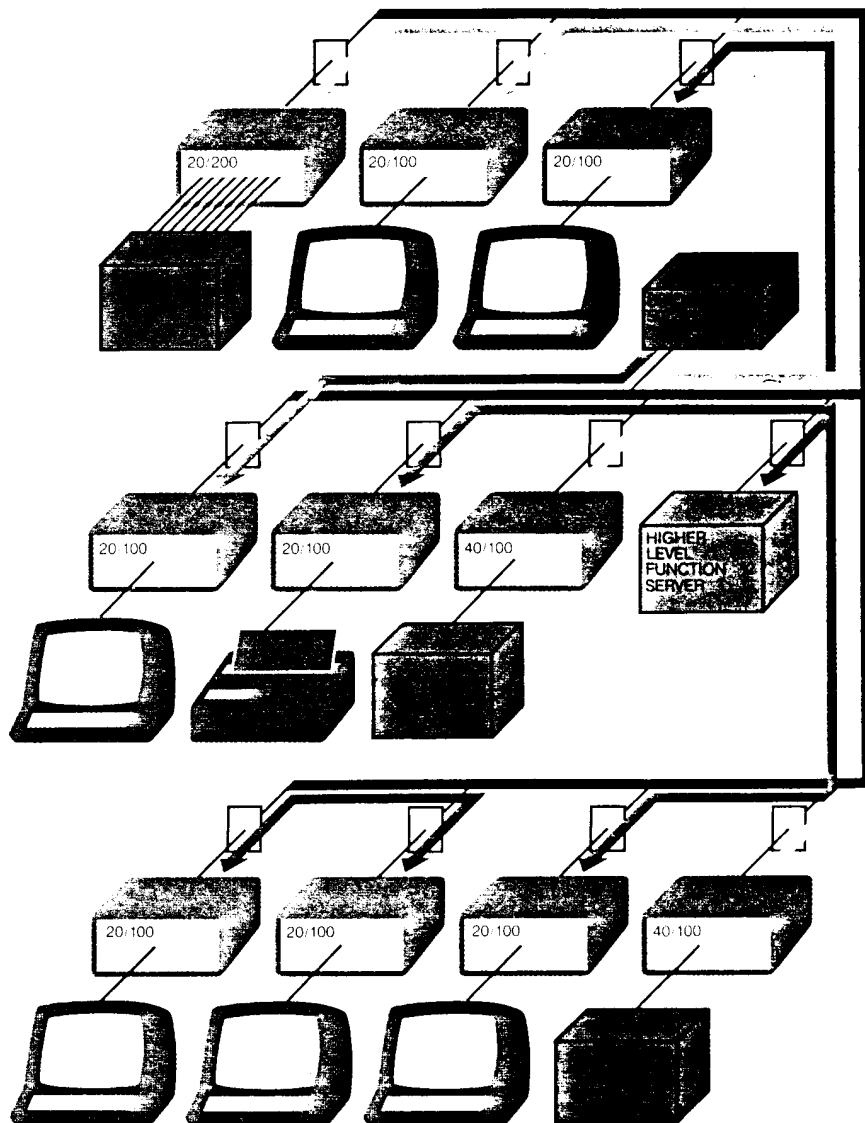
Sytek is affiliated with General Instrument Corporation, the world's largest supplier of CATV equipment. Sytek's

With LocalNet, multiple nodes share one frequency channel, and multiple channels share the same coaxial cable. A bridge permits interchannel communication.

LocalNet products are serviced at over 250 locations nationwide through General Instrument's Data System Group.

Towards an Intelligent Networking Environment. LocalNet can become the foundation for a total intelligent networking environment, which includes such func-

tions as data communications, security, energy management, voice, and telecommunications, as well as video. With LocalNet, you can plan for installation of equipment that hasn't been dreamed of today. No other local area network gives you more freedom in managing today, and preparing for tomorrow.



Sytek, Inc.
1225 Charleston Road
Mountain View, CA 94043
(415) 966-7330

0000-0001 4/83
© Sytek, Inc. 1983
Printed in U.S.A.

**Additional Applications
for
Trusted Domain Machine (TDM)**

Attachment D

MULTI-SYSTEM DATA TERMINAL DESIGN

USING AUTHENTICATORS

R.K. Bauer

1. INTRODUCTION

Often Government personnel require access to multiple EDP systems, each classified at a different level. Use of separate terminal equipment for each system is inefficient, expensive and often leads to an explosion of terminals and interconnecting cables. Use of a single terminal, while more convenient, poses the danger of information spillage. Classified information may still be present within the terminal when it is connected to an unclassified EDP host after a classified work session. This problem is of concern for terminals which include internal storage and especially for those which incorporate permanent storage media such as disks, tapes and nonvolatile memory. Currently these terminals must be "sanitized" after use in a classified environment in order to completely expunge any classified information residue. Such sanitization procedures are inconvenient, make demands upon the operator and require trust in their correct function.

Rather than ban the use of storage terminals, a solution is sought which not only allows their use, but exploits their storage features. It is convenient for such terminals to perform a data transfer function. Information selected from an unclassified system is stored within the terminal and later transferred onto a classified system to aid work performed there. Selective information transfer to systems at the same or higher classification level is a distinct advantage and represents no security risk. However the reverse of this process represents information downgrade or compromise, and must be prevented.

This paper describes a solution approach based upon data marking with authenticators. The approach utilizes special marking and filtering devices and specialized terminals which isolate the keyboard from the display, local memory, storage and computation facilities. The design makes extensive use of hardware security features and is far less complicated (and more trustworthy) than solutions employing multilevel secure operating systems in the terminal.

2. DATA MARKING AND FILTERING

Markers and filters are interposed between all EDP equipment and the communications medium which they use for intercommunication. The markers indelibly mark data packet classification level prior to their entry into the communications medium. The filters effectively block high level data which their host may not view. The communications medium underlying the markers and filters may be point-to-point cabling, a local area network, or even a satellite link. Its exact nature is unimportant, only that it is certified to carry the highest level of classified information presented by any attached host EDP system. The markers and filters must also be protected at this level.

Figure 1 shows two EDP hosts operating at different security levels and a terminal which may access either host. Data which leaves either EDP host is labeled with its classification level by the attached marker. To ensure that the label and data will not be altered, it is sealed with an authenticator. This guarantees to the receiving filter that the packet and its level are authentic and have not been altered or otherwise manipulated during transit or storage. The authenticator is a cryptographic tag computed over the entire data packet contents and its level, based upon a secret key. The process is analogous to encryption, but rather than producing scrambled data, produces the authenticator. The authenticator can be validated by any filter having the correct key. Any single bit alteration in the data, level or key will produce a drastically different authenticator. Algorithms suitable for encryption are easily used for this purpose. DES in cipher block chaining mode is used by the RECON Guard, although stronger algorithms are as easily employed.

Filters are able to recalculate the authenticator for data packets presented to them using the secret key and levels which it stores. It notes if it is authorized for the classification level indicated by the label in the data packet, recomputes the authenticator using the key and compares the result with the original authenticator appended on the data packet. If they match the data is passed, otherwise it is blocked. The level and authenticator on acceptable data packets can either be stripped or left intact by the filter as is convenient. If any of the following conditions exist, the data packet is not passed:

1. The filter is not authorized for the classification level indicated by the data packet.
2. The data in the data packet was altered
3. The classification level marking in the data packet was altered
4. The authenticator in the data packet was altered
5. The data packet authenticator is absent

Point 1 is the means by which viewing rights are controlled. Unlike the marker, the filter may accommodate several

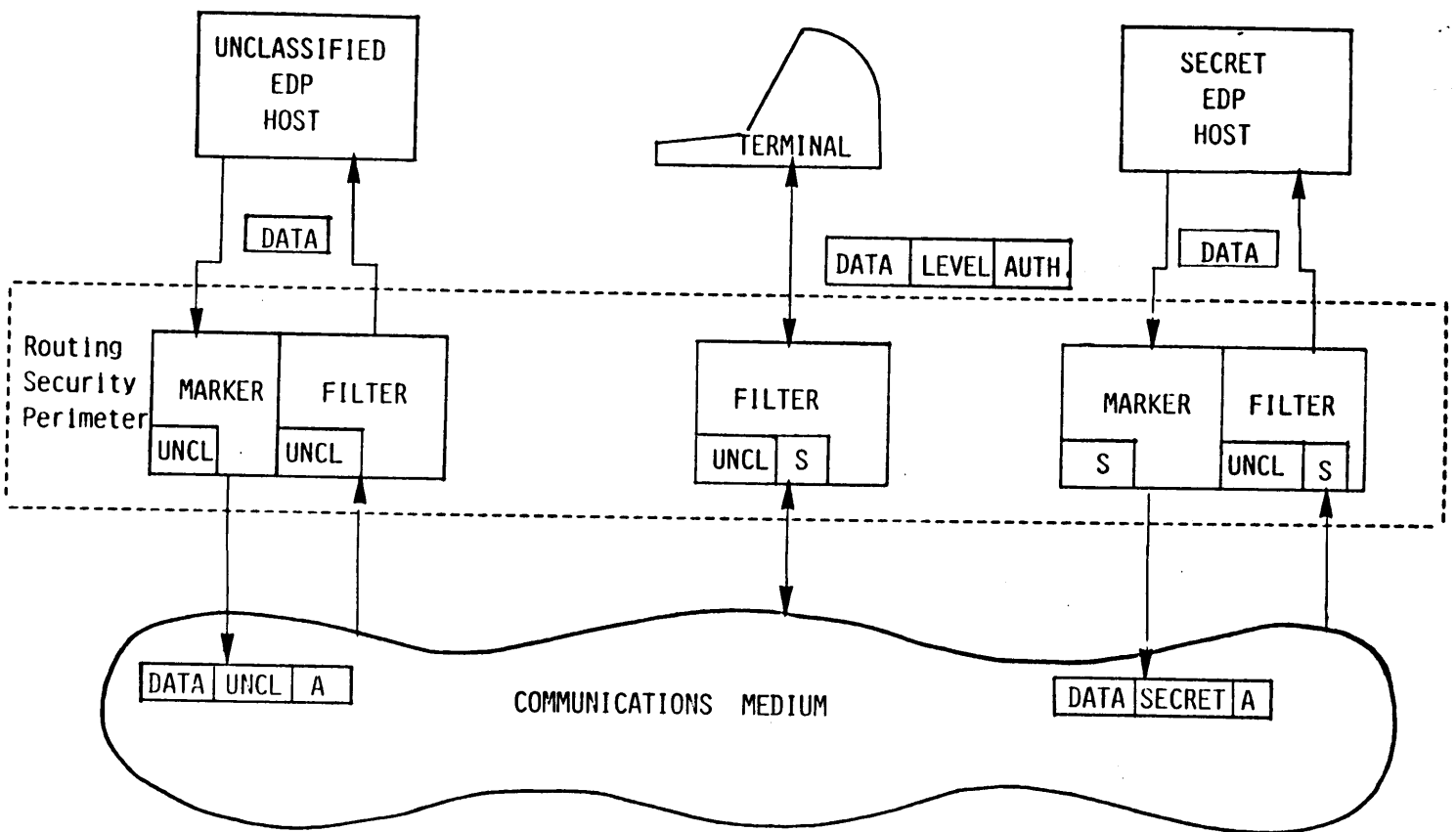


FIGURE 1: DATA PACKET MARKING & FILTERING

classification level/key pairs depending upon the EDP host's viewing rights. For example, a Top Secret host may have keys for Unclassified, Confidential, Secret and Top Secret allowing it to view data packets with any of these markings. Points 2-4 represent inadvertent or intentional tampering with the data packet during transmission or storage in the terminal. All such tampering is detected (barring cryptanalysis of the authenticator calculation algorithm or key compromise) and data packet delivery prevented. Point 5 covers loss of authenticator and misrouted information from the communications medium. It is also useful for designating unreleasable data packets.

We will now explain how data can be moved from the Unclassified EDP host to the Secret Host and how reverse movement is blocked. The terminal establishes a session with the Unclassified EDP host and attempts to move data into the terminal storage. This is successful. The marker marks the data Unclassified, and the terminal filter is authorized at that level. The terminal filter leaves the level and authenticator intact, protecting the data and level from undetectable alteration while within the terminal. The terminal now establishes a session with the Secret EDP host and attempts to forward the data. This also is successful. The terminal filter forwards the outbound packet without examination or modification. Any discrepancies are detected by the Secret host's filter upon receipt of the data packet. Alternately, the terminal may have a direct, outbound only connection to the communications medium for this purpose. The outbound only requirement stems from the need to filter all incoming data packets to the terminal.

Similarly Secret data can be moved, with level markings and authenticators intact, into the terminal. However, this data cannot be exported to the Unclassified system because it will be blocked by the Unclassified host filter which is not authorized at that level. If the terminal operator alters the level in the packet, the filter would block transmission as the authenticator would be incorrect. Similar manipulations are also detected by changes in authenticator.

3. TERMINAL DATA ENTRY

Unfortunately there is an additional complication not covered by Figure 1. This is the issue of new data entry at the terminal. The difficulty is in determining the proper level at which the terminal marker(s) should operate. Some terminal data entry is essential if for no other reason than to issue logon sequences for the various EDP hosts, formulate data base queries, enquire concerning query status etc. Obviously such information must be unclassified when directed to an unclassified system, but may contain sensitive information when queries are directed to a classified system. For some applications it may be possible to use "canned" logon sequences and queries with precalculated

authenticators. However this is an application specific solution, greatly restricts operator flexibility and invariably leads to future alteration. Another application specific approach uses markers which can recognize data at different classification levels. This rather unsatisfactory solution requires rigidly controlled formats, a great deal of trust in software correct function, and has little flexibility.

Most generalized approaches rely upon the operator to correctly label keyboard entered data. Approaches differ in the degree of operator support provided and in the level of operator trust. The approach illustrated in Figure 2 allows the operator to switch select the classification level of data entered, which has interesting sanitization implications.

The keyboard is memoryless and strongly isolated from the rest of the terminal. This eliminates the need for sanitization when the operator switches from a higher to lower classification marking. Through a user selectable control sequence, the keyboard may instruct the local computer or send control information onto the communications medium. The keyboard controller module is responsible for directing keystrokes to the local computer or marker as specified by the user. It also prevents transfer of data (classified or otherwise) from the local computer into the keyboard subsystem. Keystrokes destined for the marker and remote hosts are directed to the dedicated marker where they are labeled according to the operator selected switch position. These packets are sealed with an authenticator prior to transmission.

The terminal computer and display subsystem connects to the communications medium via its own filter which allows it to receive and forward data packets. Incoming data packets from the communications medium are screened to ensure they are viewable by the terminal. Incoming data packets passed by the filter have their level markings and authenticator intact. These marked packets may be stored, processed or displayed as specified by the operator via the keyboard, and otherwise intermixed with unmarked and unsealed data. Data directed from the display subsystem to a remote host is forwarded by the filter without examination or modification to the communications medium. However the display subsystem and filter have no provisions for marking packet level or appending authenticators. Therefore only previously marked and sealed data received by the terminal can be delivered to its destination. Only these packets will have correct authenticators and be passed by remote host filters.

Depending upon the communications interface, multiple simultaneous connections can be supported, even to EDP hosts running at different security levels. Since the operator must correctly label the security level of keyboard entered data, all displayed data must be clearly marked with its classification level. Visual segregation of multilevel data is readily accomplished via separate display windows, colors, or type fonts for each

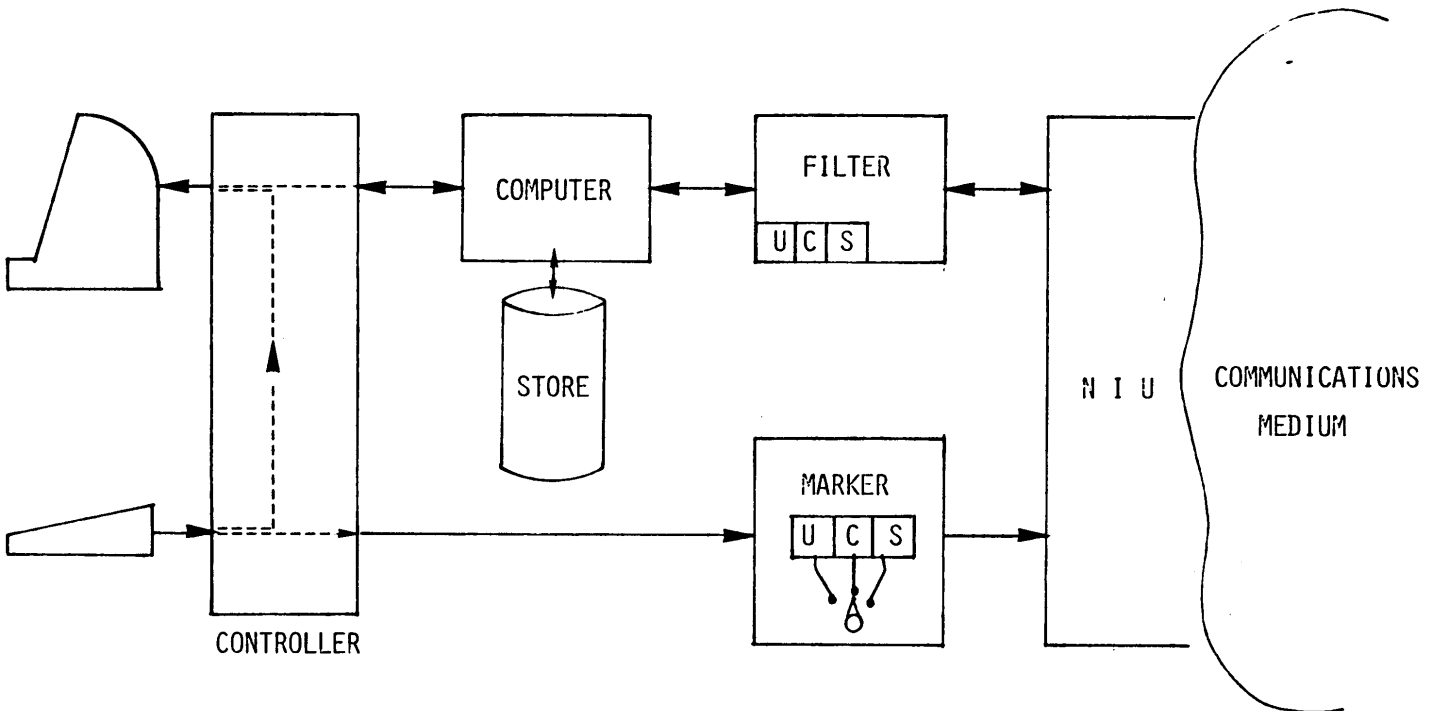


FIGURE 2: TERMINAL BLOCK DIAGRAM WITH KEYBOARD INPUT MARKED AT OPERATOR-SELECTED LEVEL

different level. Equally important is correct and complete display of classified information. Obviously, inaccurate display of data and its sensitivity level might confuse the operator, causing improper marking of keyboard entered data. This condition is true even when only single level is displayed, making display hardware and software security relevant.

4. IMPLEMENTATION CONSIDERATIONS

The previous sections raised substantial security requirements which are levied upon the terminal architecture and subsystems in addition to normal functional requirements. These security requirements affect terminal design as strongly as functional requirements and influence make/buy decisions. This section traces further the impact of security upon terminal implementation. Implementation impacts of terminal keyboard and display requirements are discussed followed by a discussion of the marker/filter implementation.

4.1 Terminal Considerations

A major requirement is suitable isolation of the keyboard and display subsystems. Recall that all outbound keyboard input is labeled and sealed according to the classification level switch position. Restricting this to keyboard input not only limits the rate at which data might be compromised but makes a clear impression as to the source of data so labeled. The best isolation is accomplished in hardware by completely separating the circuitry for the two subsystems. Suitable "one-way" communication between the two subsystems could be implemented by a specific hardware protocol. Verification of correctness is accomplished by design review and inspection of actual hardware circuit board traces.

Accurate and complete display of classified information is another important issue. The display software and hardware must be able to recognize data packet format, extract the appropriate classification level label, and accurately display all the data in the packet in the appropriate color, font or in the appropriate window. Isolation of this functionality from the general local computer is important to ease verification of these properties, giving rise to the special controller module shown in Figure 2. This controller hardware directly controls the physical aspects of the display and is programmable. The display control software must be verified to properly manipulate the hardware in response to commands to display classified data.

Most of the other terminal requirements center upon the support of and interface to the marker/filter subsystem. Because markers and filters must be protected to the same extent as the communications medium, it is desirable to physically co-locate them. In this approach the terminal must support communications to a marker/filter subsystem which is not necessarily located next to,

or within the terminal enclosure. These communications must provide a trusted path over which the terminal may set the marking level in the marker. Alternately the marker and terminal can be co-located. This eases the interface chores between the terminal and marker and allows switch selection of marking level. However the marker must be secured and will likely require a special enclosure.

4.2 Marker/Filter Subsystem Considerations

This subsystem is the heart of the approach and embodies little risk due to extensive experimentation conducted during the RECON Guard program. This program resulted in the implementation and security verification of a prototype marker/filter system for use in the RECON environment. Evaluation of its suitability for protecting sensitive RECON data base records while providing common network access to others has been successfully completed.

This prototype equipment is well adapted for use in this application. Each filter or marker is implemented by three separate processing environments, each supported by its own microprocessor. Two microprocessors independently interface to the protected data base machine and the network communications processor. The third microprocessor performs all security processing. The security processor is supported by an encryption peripheral which computes authenticators based upon the secret keys it stores. The hardware design ensures all high to low data transfer is mediated by the security processor. The simplicity of the security task, freedom from interface concerns and software verification ensure that the security processor software is correct and secure.

5. A. PRACTICAL APPLICATION

The ideas in the previous sections solve an immediate problem. Government intelligence analysts often draw upon a variety of EDP systems to prepare intelligence reports. While the analyst's primary EDP system may be highly classified, a wide range of less restricted supportive material may be available at other agencies which would be of tremendous value in report preparation. As an extreme example, dial up access to unclassified medical and/or newspaper data bases is useful. The goal is to allow information extraction from these other sources while maintaining complete confidence that classified data within the terminal is not leaked during the connection.

Figure 3 shows a configuration of networks, hosts, terminals, markers and filters which support this application. Local agency resources (Hosts A&B and terminals a&b) are interconnected via a local area network. The local agency network is connected to an interagency network, allowing connection to Host D at another agency and connection to a dial-out modem for accessing an

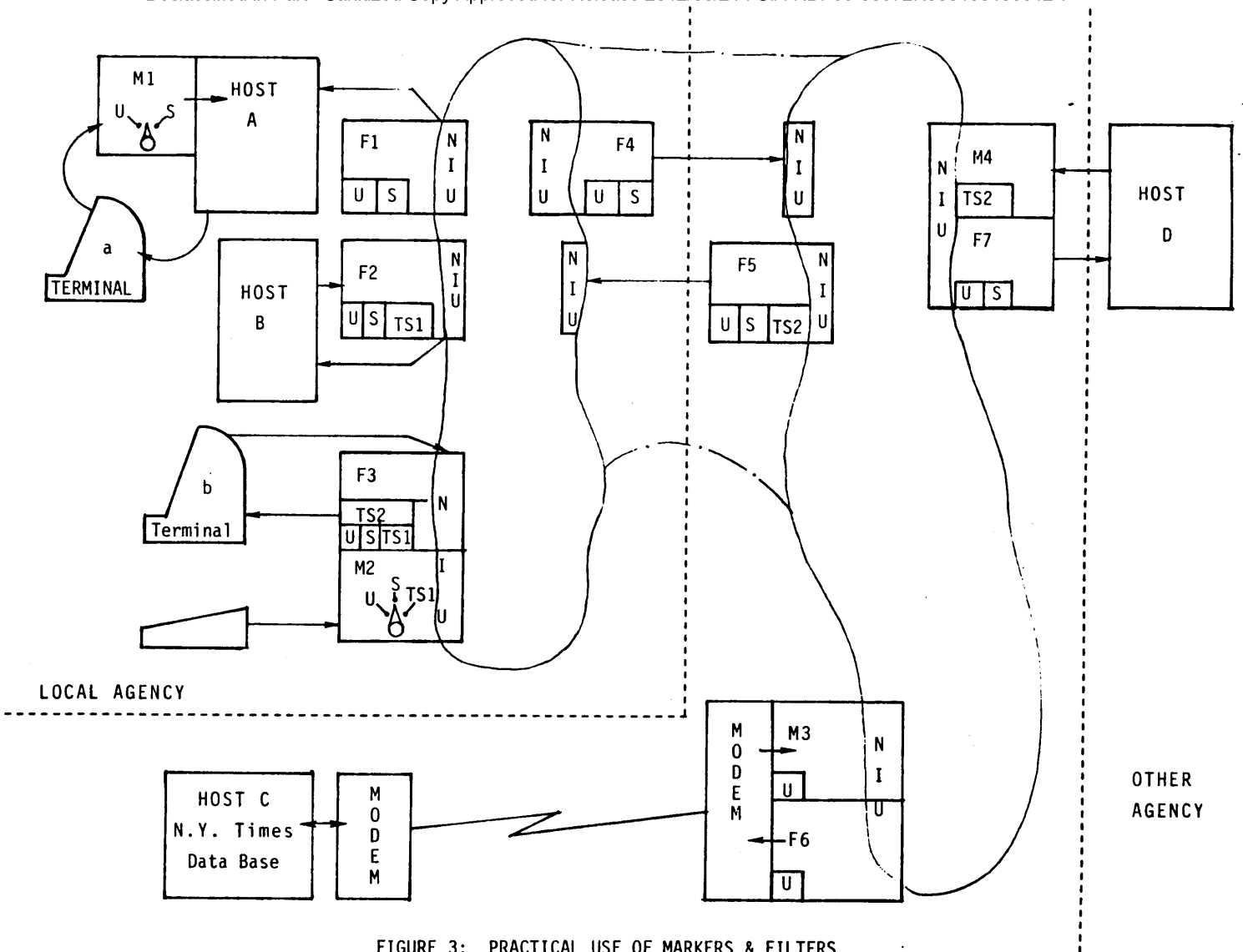


FIGURE 3: PRACTICAL USE OF MARKERS & FILTERS

unclassified data base.

Table I describes viewing, forwarding and creation rights for the different hosts and terminals in Figure 3. The first table column (MAY VIEW OR STORE) lists the level of packets which may be input and optionally stored by the host or terminal. In our example there are five categories of data packet: unclassified (U), secret (S), top secret category 1 (TS1), top secret category two (TS2), and unmarked, which is data with no level marking or authenticator.

Table I: Host and Terminal Capabilities for Figure 3

DEVICE	MAY VIEW OR STORE	MAY FORWARD	MAY CREATE
Term a	U,S,TS1,TS2, unmarked	-	U,S (M1)
Term b	U,S,TS1, TS2, (F3)	U,S,TS1,TS2	U,S,TS1 (M2)
Host A	U,S,TS1,TS2, unmarked	U,S, (F1)	-
Host B	U,S,TS1,TS2, unmarked	U,S,TS1 (F2)	-
Host C	U (F6)	-	U (M3)
Host D	U,S (F7)	-	TS2 (M4)

The second column details forwarding capabilities. Terminals and hosts with direct outbound connections to a network may release or forward any data they contain. Data packet forwarding by terminals and hosts whose outbound connections are mediated by a filter is limited to appropriately marked data packets. Finally the last column shows data packet marking or creation capabilities. Only markers can mark and seal a data packet. Some markers apply a fixed level, others allow the operator to select the marking level. Throughout the table, filter and marker identifiers in parenthesis show which marker or filter is active in supplying a capability or imposing a restriction.

Filters four and five do not appear in table I and could be removed in some instances as shown by the long dashed line in Figure 3. They are introduced to provide a greater degree of local control and security confidence. The short dashed line delineates local "zones of control." Filter 4, under the local agency's physical control, provides a high degree of confidence that only unclassified and secret data will be allowed to leave

the local agency. The filter in this configuration closely resembles the RECON guard.

Consider a connection between Terminal b and Host D at the other agency. Even though Terminal b can view all data and can even create Top secret data restricted to the local agency (TS1), filter F4 allows only unclassified and secret transmissions to host D. On the other hand all data contained in host D is releasable and is marked TS2 by M4 which is passed by filters F5 and F3 allowing its use by the terminal. TS2 marked data could then be forwarded by the terminal to either of its local hosts during a later connection.

As a second example, consider a connection between Terminal b and Host C, the New York times data retrieval system. Retrieved information is marked unclassified by M3 and released by F5 and F3. Queries from terminal b are marked unclassified by M2 with its switch in the U position. Queries thus marked are released by F4 and F6 before they are received by Host C. Should the operator mistakenly set the terminal marker to secret (S), the query would be blocked by F6. Similarly a missetting at TS1 would be blocked by F4. Marked data released from terminal B's computer/display unit would be similarly filtered. Unmarked data from the computer/display would be blocked by F4.

6. CONCLUSION

Practical solution of the multi-system terminal problem is possible using authenticators. Data is permanently marked as to its classification level and released only to systems with the appropriate viewing rights. Terminals may communicate with and view data from several systems operating at different security levels simultaneously. Solution implementation maximizes use of existing terminal equipment and capitalizes on readily available security technology, to yield a low risk, near term solution approach.

Guard System Final Report

Sytek TR-84010

June 14, 1984

Attachment E

GUARD SYSTEM FINAL REPORT

SYTEK TR-84010

June 14, 1984



1225 Charleston Road
Mountain View, CA 94043
(415) 966-7300

GUARD SYSTEM FINAL REPORT

SYTEK TR-84010

June 14, 1984

Prepared For:

[Redacted Box]

Office of Research and Development

Washington, D.C. 20505

SYTEK, Inc.

1225 Charleston Road

Mountain View, CA 94043

STAT

CONTENTS

1.	INTRODUCTION.....	1
1.1	Keywords.....	2
2.	SIGNIFICANT ASPECTS OF THE GUARD SYSTEM PROJECT.....	3
2.1	Hardware Domain Isolation.....	3
2.2	Processor per Domain Architecture.....	3
2.3	Data Protection Using Cryptographic Sealing.....	4
2.4	Adherence to Dissemination Policy.....	4
2.5	New Informal Verification Techniques.....	4
2.6	Guard Test System Developed.....	4
3.	SYSTEM FUNCTIONAL OVERVIEW.....	5
3.1	Security Officer Interface Device.....	5
3.2	Update Guard.....	5
3.3	Online Guard.....	6
3.4	Guard Test System.....	7
4.	DEVELOPMENT.....	8
4.1	Software.....	8
4.2	Hardware Architecture.....	8
4.3	Software Architecture.....	9
4.4	Development Environment.....	10
5.	ON SITE TEST ENVIRONMENT.....	12
5.1	Guard Test System.....	12
5.2	Update Guard to Guard Test System Connection.....	12
6.	VIRTUES REALIZED.....	13
6.1	No Compromise in Security.....	13
6.2	Ease of Software Modification.....	13
6.3	Legitimate Traffic Modulation.....	14
7.	VERIFICATION.....	15
7.1	Verification Technology.....	16
7.1.1	The Guard System Verification Approach.....	16
7.1.2	Verification Review Board.....	20
7.1.3	Designing for Verification.....	21
7.2	Project Management Issues.....	22
8.	LESSONS LEARNED.....	24
8.1	Project Management.....	24
8.2	Customer Coordination and Cooperation.....	24
8.3	Means to Test the Guard System.....	25
8.4	Hardware Debugging.....	25
8.5	Establishment of Development Environment.....	26
9.	CONCLUSION.....	27
	REFERENCES.....	28

Guard System Final Report

1. INTRODUCTION

This document comprises the final report of the RECON Guard System prototype guard device development and implementation effort (Contract No. 82F770400). The program has successfully demonstrated all aspects of data security that were attempted.

Following the conclusion of Customer confidence testing, the Guard program demonstrated the feasibility and design of a combined hardware and software device that insures data security is maintained when a sensitive data base is connected to a user community over a network.

The concept is not dependent on either the network or data base used in the demonstration project; rather it may be employed in any situation where data is transmitted from one point to another.

This technology represents an important advancement in the arena of computer information and data security. The success of this technology provides the means to revolutionize procedures used historically to insure that improper access to sensitive information is prevented. The Guard technology now allows the automated sharing of multilevel information among various levels of users with assurance that no information marked at a certain level will be disseminated to a destination level that is inappropriate.

As demonstrated by this project, the desired security policy is enforced without the need to trust existing computer systems and associated application programs. It has been demonstrated that the additional security provided by the Guard System technology may be applied to systems without disruption of current operating procedures.

Guard System Final Report

1.1 Keywords

- ◆ RECON
- ◆ COINS
- ◆ Guard System
- ◆ Data Security
- ◆ Multilevel Data Sharing
- ◆ Information Dissemination
- ◆ Processor Per Domain Architecture
- ◆ Hardware Kernal
- ◆ Hardware Isolation
- ◆ Cryptographic Sealing
- ◆ Authenticator
- ◆ Verification
- ◆ Security Policy
- ◆ Guard Test System
- ◆ Legitimate Traffic Modulation

Guard System Final Report

2. SIGNIFICANT ASPECTS OF THE GUARD SYSTEM PROJECT

2.1 Hardware Domain Isolation

The Guard System architecture realizes three separate domains which accomplish the isolation of the sensitive data base from the user network connection. Two domains are used to interface to the world external to the Guard System. One domain, termed the User Side, deals exclusively with the Guard System's communication with the network connection. Another domain, termed the Data Side, deals exclusively with the Guard System's communication with the the Data Base connection.

To enforce the security aspects of the Guard System, the User and Data Side domains possess absolutely no means of communicating with each other. Rather than rely on software techniques to accomplish domain separation, the Guard System domain separation is completely enforced through its hardware architecture.

Since there is no communication between the User Side and Data Side domains, there exists no condition under which the software of the User Side and Data Side could alone or in concert be written such that a compromise in security would be possible. As a result, the User Side and Data Side software may be written in any way that is convenient without the need of verification. For this reason, these two domains may contain 'untrusted' software without the possibility of compromising security.

The third domain is trusted in that it is responsible for the enforcement of the chosen security policy. This domain, termed the Master, communicates with the untrusted User Side and Data Side connections and will pass information from one untrusted domain to the other only if it has determined that the passing information will not violate the security policy. The Master domain must undergo verification (described later) due to its security critical function.

2.2 Processor per Domain Architecture

The major hardware architectural innovation stems from the realization of the three isolated domains by separate microprocessor single board computers. Separate software runs independently on each processor, and within the Guard System, each processor communicates exclusively with the Master processor board.

The processor per domain architecture affords absolute separation of domains unlike attempts in the past where only one processor under software control attempted to enforce isolation.

Guard System Final Report

2.3 Data Protection Using Cryptographic Sealing

The integrity of sensitive information passing from the data base to the network user is protected by a form of cryptographic sealing that insures data has not been previously tampered with following entry into the data base. Sensitive data is marked with a cryptographic seal prior to entry into the data base. In this demonstration project the marking function was accomplished off-line using magnetic tapes, however, this need not be an off-line process.

2.4 Adherence to Dissemination Policy

The combined effect of verification of security critical system software (verification demonstrates adherence to the security policy), and the use of cryptographic sealing of record contents, prevents the dissemination of records that are not approved for release by the Security Officer.

2.5 New Informal Verification Techniques

A new software verification technique was employed that assures the proper security function of software written for security critical processors. The technique begins with definition of the system security policy which is: "Information indelibly marked for distribution to specified communities is disseminated only to those communities." A clear and concise explanation of the verification technique may be found in the Guard System Verification Report, SYTEK TR-84009, and later in this document.

2.6 Guard Test System Developed

An external test environment was developed which allows full testing of the Guard System's operational and security characteristics. The GTS is capable of emulating functions of RECON and COINS in each of three different configurations:

- ◆ Emulation of RECON only.
- ◆ Emulation of COINS only.
- ◆ Emulation of both RECON and COINS.

Guard System Final Report

3. SYSTEM FUNCTIONAL OVERVIEW

The object of the Guard System is to indelibly mark records with information that controls to whom the records may be disseminated or released. This is accomplished by three major subsystems:

- ◆ SOID - Security Officer Interface Device
- ◆ UDG - Update Guard
- ◆ OLG - Online Guard

3.1 Security Officer Interface Device

The SOID is used by a security officer to establish releasable Categories based on the content of several key fields contained in the records. The security officer will select two numeric quantities called the Key and the Initial Value (IV) which are used to compute a unique checksum or Authenticator for each record added to the data base. The Key is 56 bits in length, and the IV is 64 bits in length. The Authenticator for each individual record is a function of the Key, IV, and the entire contents of the record. The Authenticator is 64 bits in length. Only the security officer has knowledge of the Key and IV.

The information entered by the Security Officer and processed by the SOID is written into programmable read only memory (PROM) semiconductor devices which are then installed into a special purpose single board computer called the Authenticator Generator Board (AGB). The AGB's are within the trusted Master domain.

3.2 Update Guard

The UDG uses an appropriate set of SOID produced data proms each containing a Category Expression, Key, and IV, to compute and append unique Authenticators to each record presented via an input magnetic tape. The UDG will determine which (if any) categories specified by the security officer match in turn the specific contents of each record. If a matching category is found, it is then permissible to mark that record for release based on that category. The marking is accomplished by applying a chain encryption algorithm to the record using the secret Key and IV from the appropriate SOID data prom. If any particular record matches no category of the set produced by the SOID, a null authenticator is appended to the record. Each record processed by the UDG is written to an output tape that is subsequently loaded into the data base.

Guard System Final Report

3.3 Online Guard

The OLG accepts queries from prospective network users and passes them on to the data base machine. All record responses are subject to the filtering action of the OLG and one of two possible actions will occur:

- ◆ RECORD RELEASE - Only if all of several conditions are met.
- ◆ RECORD DENIAL - If any of several conditions are not met.

The following are conditions that must be met in order to permit record release:

1. The OLG must contain at least one SOID data prom that has a Category Expression matching the contents of the record's key fields.
2. If one or more matches are found, the Authenticator for each matching category is recomputed using the record's contents and each respective Key and IV.
3. The computed Authenticators are compared with the Authenticator contained in the record. If one and only one match exists, the record may be released.

Conditions that lead to record denial include:

1. If no matches occur amongst the set of Category Expressions contained in the OLG, a record is attempting to pass which the security officer has determined may not be released over the network connection.
2. If a matching category expression is found, and the computed Authenticator does not match the Authenticator contained in the record, the contents of the record have been altered at some point between the UDG attaching the Authenticator and presentation to the OLG device. This implies record tampering either while entering into the data base, while in the data base, or on its way out of the data base to the OLG separating the data base from the user network.

If a record is releasable, it is sent on to the network user. If a record is not releasable, it is written to a monitor device for security reasons and no indication of any kind is sent to the network user.

Guard System Final Report

3.4 Guard Test System

The GTS consists of a set of programs written in the C language running on a Momentum Hawk Unix system built around the Motorola 68000 microprocessor.

The GTS possesses the ability to emulate either the user network connection, the data base connection, or both at the same time. The GTS employs a menu driven user interface and performs the following functions:

- ◆ Operator transmission of any byte value to either the User or Data sides of the guard.
- ◆ Operator creation of disc files containing user queries and data base responses. This includes all IBM JCL and record formats as observed during actual connection to the data base machine.
- ◆ Automatic monitoring and display of all data into and out of each of the Guard System's User Side and Data Side connections.
- ◆ Execution of operator specified test suite or command files. All individual GTS command operations may be orchestrated to execute automatically under GTS control and repeat continuously for some operator specified time.
- ◆ When simulating the user network connection, the operator may submit properly formed data base queries and observe responses actually supplied by the Government's data base machine. Of course the Guard System will be performing its required filtering function.
- ◆ When simulating the data base connection, the operator may supply IBM JCL and data base records contained in GTS disk files to the Guard System for processing and possible release to an actual user network connection and data base query.

Guard System Final Report

4. DEVELOPMENT

Guard system development began in 1982 with the completion of the Security Guard Device Functional Specification [1], April 1982, and the RECON Security Guard System Software Design Specification [2], January 1983.

4.1 Software

Software for seven different single board computers (SBC's) was designed, coded, and tested:

- ◆ Update Guard Input Slave
- ◆ Update Guard Master
- ◆ Update Guard Output Slave

- ◆ Online Guard User Side
- ◆ Online Guard Master
- ◆ Online Guard Data Side

- ◆ Authenticator Generator Board

Software for the Guard Test System was designed, coded, and tested.

4.2 Hardware Architecture

With only one exception, the Guard System was built with off-the-shelf hardware. A special board was designed and built by Sytek called the Authenticator Generator Board (AGB) as no commercially available equivalent existed.

The Security Officer Interface Device (SOID) consists of a Wicat S/150 self-contained Unix microcomputer with attached prom programmer for the creation of AGB data proms.

Both the UDG and OLG are built with Intel 86/12A single board computers (SBC's) contained in a Multibus chassis. In addition to the Intel SBC's, each chassis may contain several AGB's which perform the chain encryption function.

An important concept of the Guard System hardware architecture is the separate domains enforced in hardware. The three domains in each of the UDG and OLG are as follows:

Online Guard:	Untrusted	Trusted	Untrusted
	_____	_____	_____
	User Side	Master AGB's	Data Side

Guard System Final Report

Update Guard:	<u>Trusted</u>	<u>Trusted</u>	<u>Untrusted</u>
"	Input Slave	Master AGB's	Output Slave

The Master may transfer data to or from any other processor, however no other processor may transfer data to or from any other processor. Thus, for example, in the OLG, there is no way that the User Side or Data Side may communicate with any other processor except the Master. Since the Master is considered a trusted processor and the software executed by the processor is verified, absolute data security is enforced.

All communications between processors occurs via the Multibus chassis into which all processor boards are placed. As a result of simple electrical modifications to the User Side, Data Side, Input Slave, and Output Slave, Intel 86/12A boards, they have no capability of initiating a bus request which is required in order to communicate with other processors. Each of these boards may respond to a request across the bus, however only the Master is capable of initiating such a request. The AGB was designed and manufactured by Sytek to have no facility to make bus requests.

The purpose of the separate domains is to make possible the interconnection of the dissimilar environments of the user network connection and data base connection while insuring absolute security. Since the User and Data side processors are not trusted processors in that nothing they do can cause a compromise in security, the software is free to be written in whatever way necessary to accomplish the connections to their outside environments.

The UDG Input Slave is considered to be a trusted environment. It requires verification due to the need for unpacking data base records from an IBM format tape prior to presentation to the Master for possible authentication.

4.3 Software Architecture

Each SBC contains all of its software programmed into programmable read only memory (PROM) held on board. Each SBC is self-contained, and there is no operating system or disc storage devices in the Guard System.

All processors operate independently of each other and are responsible for managing their own ability to communicate with the outside world and the Master processor where appropriate.

Guard System Final Report

The software of each SBC is written in the Pascal language with some extensions written in 8086 assembly language where necessary. Pascal was chosen as it possesses properties that, when applied, aid in software verification.

The following Pascal constructs prove useful in writing software for systems that are to be verified:

- ◆ Block structured language.
- ◆ Restrictions on variable access.
- ◆ Structured data types.
- ◆ Ownership levels.

Great care was taken in the design of software that resides on trusted processors as such software is requires verification. From the beginning the need for verification was a primary consideration in the design of the OLG and UDG Masters, and the AGB. As a result of careful application of Pascal constructs and attention to programming style by those writing actual software, the trusted processors could be argued against their comprehensive set of security assertions with relative ease.

Had the software design and programming effort on trusted processors not been performed with verification strongly in mind, it would be difficult, if not impossible, to assert and verify the required security properties of the Guard System.

A fundamental design property of the Guard System is the separation of domains into trusted and untrusted environments. All security related properties of the Guard System are entrusted to the secure domain of the Master and AGB's. Interconnection of the Guard System to the external environment (the sensitive data base and the user network) is performed within the two untrusted domains of the User Side and Data Side processors. Since no actions occurring in the untrusted domains can possibly cause a compromise in security, the software on these processors may be written in any way that is convenient and that permits proper function. Typically the task of connecting to external environments is not a simple one. It is always a desirable goal to minimize the amount of a system that requires verification. Verification can be a slow and costly activity which, nonetheless, is required of a system serving a security critical function.

4.4 Development Environment

All Guard System software was developed on a VAX 11-750 operating under Berkeley 4.1 Unix. The majority of the source code for all SBC's was written in Pascal in order to aid in verification tasks. Some assembly routines were written where necessary.

Guard System Final Report

All source code is maintained under source control provided by VAX-Unix. The source control provided facilities for tracking changes in source and insured a locking mechanism for source integrity while under development.

A cross compiler, cross assembler, and cross linker were employed to enable complete software development in the multi user environment of the VAX. Operations on the VAX culminate with a downloadable object module suitable for use in a microprocessor development environment.

Readied code modules would then be transmitted to an Intel MDS/ICE-86 microcomputer development and emulation computer. Debugging occurred on the MDS and ultimately the code sent from the VAX is programmed into proms to be installed on the appropriate SBC. All code is executed from on-board proms as there is no disc storage or operating system in use.

While the development environment described provided good controls, there were also drawbacks in the particular tools employed. The cross compiler was found to have bugs that were circumvented by choosing other equivalent Pascal syntax that accomplished the same original goal. The chosen cross compiler and associated software ran particularly slow and proved annoying during rapid code development. As a result of the cross compilation process, all higher level language symbols used in the source code were lost by the time debugging was possible in the MDS environment and thus examination of referenced memory proved tedious. Due to the size of the code on each board, and the limitations of the MDS, even a simple code change will require the time consuming process of reprogramming up to eight proms.

Guard System Final Report

5. ON SITE TEST ENVIRONMENT

Two tools developed by Sytek and incorporated on site have proved invaluable in enabling integration and Government confidence testing.

5.1 Guard Test System

The GTS described earlier has provided a means to completely test all functions of the Guard System without direct connection to either the user network connection or the data base connection. In fact, acceptance testing of the Guard System's security function was conducted using the GTS to drive the Guard prior to the Government provision of either a user network or data base connection.

Tools contained within the GTS have aided in the discovery of actual data base system characteristics during Guard System integration efforts.

The GTS has allowed Government confidence testing to proceed without the need for both the user network and the Government data base to be connected to the Online Guard at the same time, or at all.

5.2 Update Guard to Guard Test System Connection.

The provision for special direct connection between the UDG and the GTS was developed to enable actual records authenticated to be included in the GTS collection of response records, as well as to be loaded into the actual Government test data base.

By having identical records on the GTS during data base emulation, the very same records used during network emulation may again be used.

Guard System Final Report

6. VIRTUES REALIZED6.1 No Compromise in Security

"
The Guard System underwent numerous tests in attempts to violate its security properties. Tests were independently performed by Sytek personnel, the Government COTR, an impartial Government representative (J. P. Anderson Co.), and the Government's own security staff. At no time during testing did the Guard System fail its security function.

Tests performed include, but were not limited to:

- ◆ Abnormal communications protocols.
- ◆ Abnormal record sequences.
- ◆ Abnormal record formats.
- ◆ Abnormal Guard System configurations.
- ◆ Record size boundary conditions.
- ◆ Tampering with record key field contents.
- ◆ Altering valid record message contents.
- ◆ Altering the Authenticator within records.
- ◆ Embedding messages in valid records.

In each and every case, the Guard System performed its security function correctly without deviation.

6.2 Ease of Software Modification

The modular design of the Guard System afforded separation of logically dissimilar functions. This separation of function and distributed processor per domain architecture isolated changes necessary during testing and integration to a subset of the entire Guard System.

In the case of all processors, structured design and coding practices enable straightforward software changes. The most volatile software environment during integration and testing concerned the OLG Data Side which interfaces to the Government's data base.

In order to connect to the Government data base, the Guard System was required to appear as an IBM remote job entry (RJE) station consisting of a card-reader/printer-punch device. All emulation of such a device was performed by the untrusted Data Side slave processor. Many iterations were required during integration to adjust the Data Side's ability to converse in acceptable IBM JCL language commands. As each new JCL quirk was discovered, changes to the Data Side code were trivial to implement.

Guard System Final Report

During integration it was found that the format of linear data base records read from the IBM input tape into the UDG differed from the format of linear records returned in IBM format from the Government's data base machine. This discrepancy was discovered as valid releasable records were not being released by the Guard System due to the difference in record formats. The Guard System was again demonstrating its security function. A simple correction was made in both the Update and Online Guards to provide for this difference.

Due to the modular and structured design of the Guard System software, when the two different formats of the DN.SO field surfaced, a correction to accommodate both field formats was minimal. Due to the organization of the Guard System software, and the methods employed in the verification effort, there was virtually no effect to the software verification resulting from this change.

When an operational Host Access System (HAS) exists, any changes found necessary during integration will have no effect on any trusted software, or the untrusted Data Side.

6.3 Legitimate Traffic Modulation

Legitimate traffic modulation is a technique where an attempt may be made to pass information by encoding the information in what otherwise is normal system operation. For example, a status return line could be modulated to encode information represented by the timing of the status request and return, or the number of times a particular status is returned with respect to another particular status.

The Guard System has been designed to limit the bandwidth of legitimate traffic module to such an extent that it is impractical to attempt to pass information in this manner. It would be far more practical for someone who desires to pass information to physically copy or remove the information from one environment, and carry it to the other environment. Classical security procedures deal well with physical movement of sensitive information.

Guard System Final Report

7. VERIFICATION

The verification of the Guard System is a rigorous review of the system which demonstrates that it meets particular security requirements. Verification demonstrates that "a system specification is consistent with (i.e. enforces) a certain security model, or that a lower level specification corresponds to (is a correct refinement of) a higher level specification." [14] In the case of the Guard System, the trusted software is shown, through intermediate levels, to satisfy the system security requirements.

While massive reports are often produced to document the verification effort (e.g. [10]), the true, intangible product of verification is a degree of assurance that the implementation of the system is consistent with the security requirements of that system. Trade-offs must be made for each application, e.g. between system development cost (including verification) and degree of assurance. No general purpose verification methodology has been identified which is optimal for all applications, for all security policies, and for all budgets. What Sytek has attempted to do for the Guard System is specify the optimal verification methodology for the particular application.

The verification of the Guard System was informal. Informal verification demonstrates the consistency between each level of specification in precise prose. This is in contrast to formal verification, where the specifications are structured to allow mathematical analysis, and formal, mathematical techniques are used for the verification.

The verification was performed in three phases. In Phase I, subpolicies were defined for each of the Online and Update Guards, and these subpolicies, taken together, were shown to imply the overall system security policy. Phase II further refined these subpolicies into sets of assertions about the behavior of the trusted software, and these assertions were shown to imply the corresponding subpolicies. Phase III then showed that the trusted software satisfied the corresponding assertions, completing the demonstration that the Guard System implementation (in the form of software) enforces the system security policy.

The following sections discuss various aspects of the verification effort for the Guard System, and verification technology in general, as outlined below:

- ◆ Verification Technology
- ◆ Project Management Issues

The section regarding verification technology addresses the approach taken to the Guard System verification. The strengths and weaknesses of the approach taken are discussed in this

Guard System Final Report

section, along with the impact of the verification upon the overall system implementation. The section pertaining to management issues highlights those aspects of project management for the Guard System which were affected by the verification effort, and emphasizes some considerations for future projects requiring verification.

7.1 Verification Technology

Verification technology refers to the analytical method of verifying a system. This section, then, addresses the verification method used for the Guard System. The strengths and weaknesses of the method used are discussed here, along with the impact of the verification upon the overall system implementation.

First, the approach taken to the verification of the Guard System is described. The issues of an informal (precise prose) verification, hardware isolation of the trusted domain, and the phased approach to verification are discussed here. The sections which follow this section describe the strengths and weaknesses of the methodology used, and of the resulting verification arguments.

7.1.1 The Guard System Verification Approach

There are three key aspects to the approach taken to verifying the Guard System. The first aspect, already mentioned briefly, is the informal methodology (arguments stated in precise prose) used to demonstrate the consistency between the top level specifications and the security policy, and between succeeding levels of specifications. Note that in this context, the software source code is a low level specification for the Guard System.

The second key aspect to the Guard System verification is the hardware isolation of the trusted domain. By isolating the trusted domain, which contains the software critical to enforcing the system security policy, only the software in that trusted domain needs to be verified. By isolating the trusted domain in hardware, instead of in software, it requires little effort to demonstrate that the isolation of domains is maintained.

The third of the three key aspects to this verification effort is the phased approach. This approach refined the system security policy into lower level specifications, which were then shown to be consistent with the Guard System software. This approach was necessary because of the difficulty of directly demonstrating the correspondence between the top level specification and the code, as is done in the typical verification approach.

The Typical Verification Approach

In order to understand the significance of these three aspects with regard to the verification, it is important to understand the typical verification approach. The following figure shows a

Guard System Final Report

model of the typical verification approach.

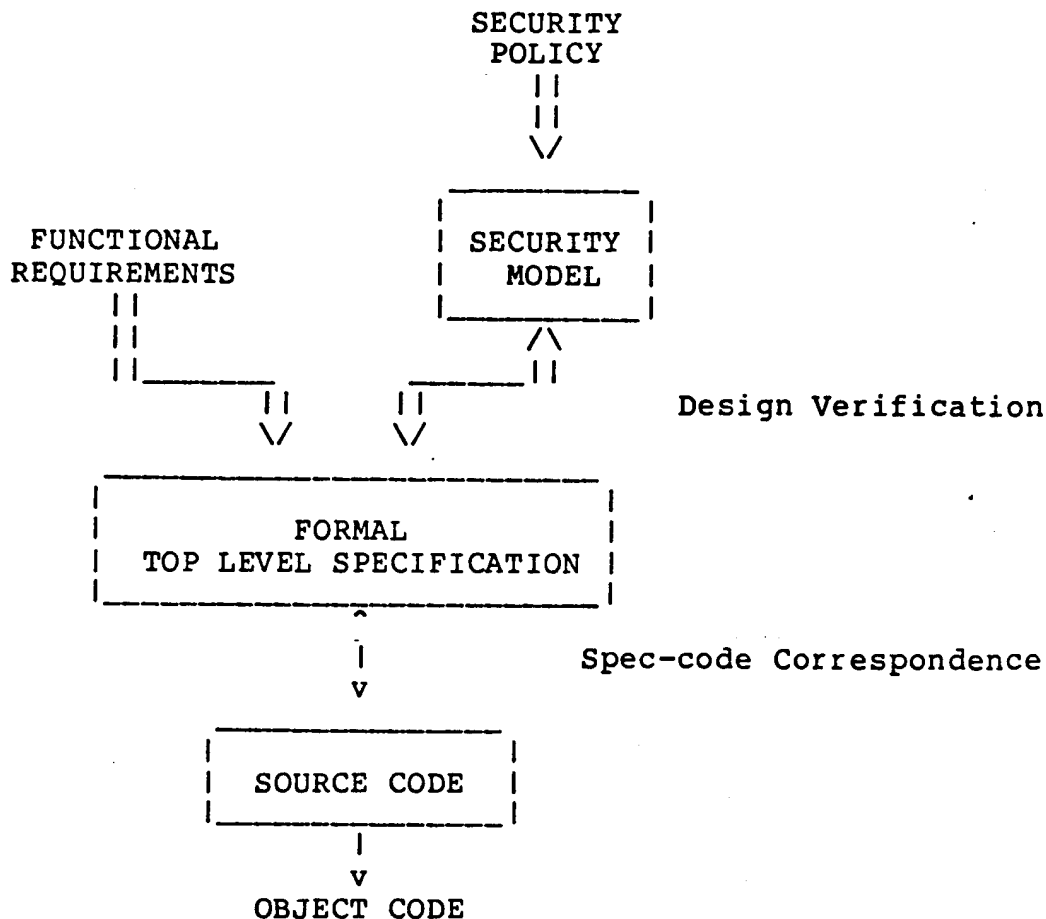


Figure 1. Typical Verification Approach

As the figure illustrates, the typical verification approach attempts to directly demonstrate the correspondence between the top level specification and the source code. Verification that the source code corresponds to this higher level specification "is very difficult with available tools and usually requires considerable human intervention." [14] Thus it either requires considerable resources, or (more often) the demonstration of correspondence is weak.

The typical approach is effective for assuring that the top level specification is consistent with the system security policy, but weak in assuring that the actual implementation (as represented by the source code) corresponds to the top level specification, and therefore also to the system security policy. Analogous to the adage that a chain is only as strong as its weakest link, the code-to-policy verification is only as strong as the code-to-specification verification, which has been shown

Guard System Final Report

to be relatively weak for the typical approach.

The Guard System Verification Approach

A more balanced approach was taken for the Guard System verification. As shown in the figure below, two parallel processes occur in this approach.

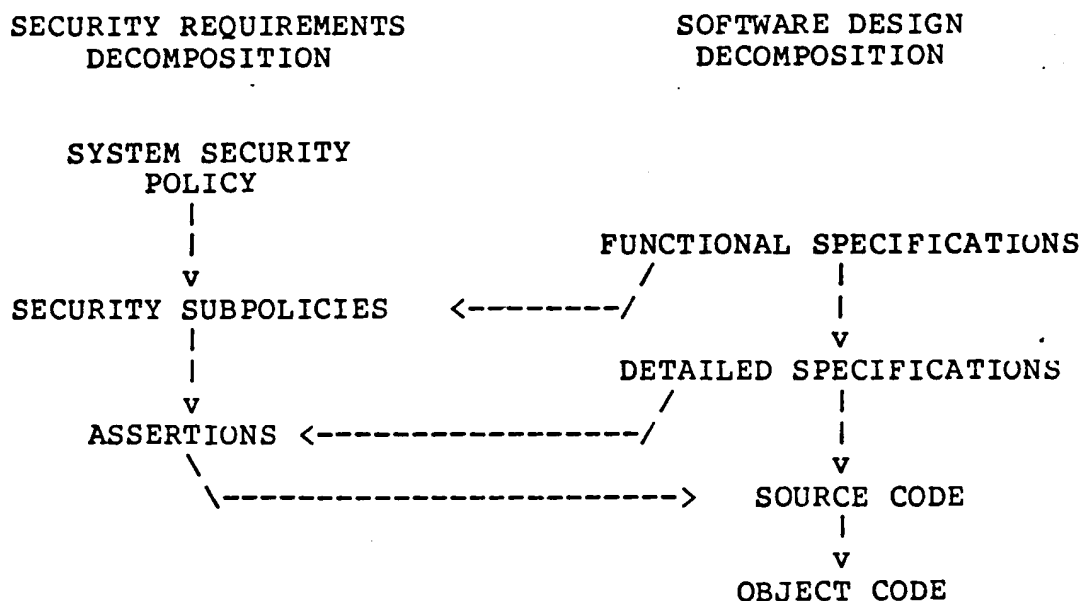


Figure 2. Security Requirements and Software Design Decomposition

First, the security requirements are refined from the system security policy to subpolicies, which are further refined into assertions about the source code modules. The software design is refined at the same time, from the functional specifications through the detailed specifications and eventually to the source code. While the two decompositions occur in parallel, they are not independent. The security subpolicies and assertions are influenced by the design, and the code is directly influenced by both the security policy and the design.

The verification was then done in three phases, as shown below:

Guard System Final Report

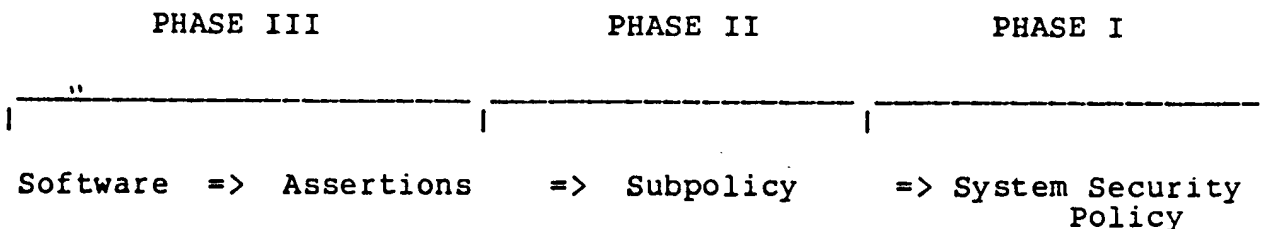


Figure 3. Three Phases of Verification

By refining the system security policy into intermediate forms, the wide gap between the top level specification and the source code of the typical verification is narrowed. Phase I showed that the subpolicies, derived from the system security policy and the functional specifications, were consistent with the security policy. Phase II further showed that the assertions, derived from the subpolicies and the detailed design specifications, were consistent with the subpolicies and therefore also with the system security policy itself.

Finally, Phase III demonstrated that the source code was consistent with the assertions, and therefore was also consistent with the system security policy. This final phase was effective because the intermediate security policy, as reflected in the assertions, was designed to satisfy the security policy within the limits of the software design, i.e. because it was designed to apply directly to the source code.

Evaluating the Guard System Verification Technique

A comparison between the typical verification and that done for the Guard System emphasizes two points. The design verification for the Guard System was not automated nor strictly algorithmic, as for formal verification, so human error could be significant. Precise prose was attempted in the Guard design verification (Phases I & II) to prevent ambiguity and reduce human error, but this technique is still prone to such error. The verification that the code satisfied the security policy (Phase III) provided greater assurance for the Guard System approach, however, since the verification arguments are generally simple and clear to the reviewer.

Since the weakest link in the Guard System verification chain is stronger than the weakest link in the typical verification chain, the overall chain is stronger. The advantages of the Guard verification technique can be summarized as follows:

- ◆ The Guard verification technique provides clear correspondence to the code level.
- ◆ The Guard technique emphasized the system security requirements throughout the design and implementation of the Guard.

Guard System Final Report

- ◆ The verification at each level was consistently demonstrated, improving the overall credibility of the verification from policy to implementation.

It is important to remember that the Guard System verification approach was selected based on this particular application. Specifically, the small size and simplicity of the system enabled this approach to be successful, as well as economically feasible. For other applications, it would be important to maintain a balanced approach; perhaps a combined methodology of formal design verification (using automated tools) and an informal verification from the code to a refined security policy specification. As more tools become available, an automated verification to the code level would be desirable.

7.1.2 Verification Review Board

Perhaps the greatest weakness in the informal verification of the Guard System is its susceptibility to human error. Since no automated tools were used to verify the arguments of consistency between levels of specification, the responsibility of doing so was given to the Verification Review Board.

In particular, Phase III of the verification (code-to-assertion correspondence) consists of arguments that each source code module meets the requirements represented by the assertions for that module. These arguments were written in precise prose by the programmers of the Guard System. The arguments were subjected to rigorous review and approval by a Verification Review Board consisting of impartial observers who were not responsible for the development of the code which implements the Guard design.

The Board reviewed the arguments for soundness of logic, and attempted to remove ambiguities in the text of the arguments. It was the persistent, iterative review of the Board which provided the greatest degree of assurance that the arguments were sound and complete.

Operational errors could result in a failure to enforce the system security policy. While the Verification Review Board did not intend to review the code itself for correctness, a small number of errors in the source code were found during the review process. Since Phase III of the verification was done after the (functional) acceptance testing for the system, these errors were obviously below the threshold of detection through standard functional testing. The added attention to the source code by the Verification Review Board and the programmers contributed to the level of confidence that the code indeed operates as specified, and that it satisfies all its security requirements.

Guard System Final Report

7.1.3 Designing for Verification

It is critical to any verification effort that the system be designed with the consideration that verification will be done. Verifying a completed system after the fact is difficult at best, possibly requiring that portions of the system be redone. The Guard System architecture was designed for verification, which greatly reduced the verification effort.

A typical approach to the design of a secure, multi-level operating system is to separate the security critical functions from the functions which are not security critical. The result is a security kernel which must be verified; the software which does not perform security critical functions does not need to be verified. The Guard System design incorporated this approach, by isolating the trusted domain in hardware. Only the software in the trusted domain required verification, thus minimizing effort.

At a lower level, much of the software in the trusted domain was written in such a way as to simplify the code level verification. Some of the general principles followed are listed below:

- ◆ As much of the software as possible was coded in the structured, high level language Pascal. Assembly language was used only when absolutely necessary.
- ◆ Wherever possible, pointers (addresses of data items which are directly accessible to the code) were NOT used.
- ◆ Wherever possible, global data, accessible to any module in the software, was NOT used.
- ◆ The call by value parameter passing mechanism was used to pass data between modules. Since this mechanism passes a copy of the value, not the value itself, it prevents modification of the original value.
- ◆ Good coding practices were encouraged.

The practices listed above are concerned with limiting access to the data being processed by the Guard and making it clear when that access was granted. By limiting the ability of software modules to modify the data, the effort required to demonstrate that such modifications were not in conflict with the security policy was minimized. Making it clear wherever such modification was possible simplified the Phase III arguments, and made it easier for the Verification Review Board to review. This prevented oversights and other human errors. These practices also made it practical to identify each such access in the source code using automated text tools.

The use of good coding practices helps minimize errors, simplify maintenance of the software, and makes the source code easier to understand. All of these features make verification

Guard System Final Report

easier to do and maintain. Minimizing errors minimizes verification effort also, since code changes (due to errors or enhancements) require the Phase III verification arguments to be reviewed.

For future applications, guidelines similar to those listed here could be used. Such guidelines need to be tailored to the application and the implementation language. The experience of the Guard System verification indicates that coding for verification can reduce the verification effort, and, since it encourages good coding practices in general, it can reduce the effort for the project life cycle.

7.2 Project Management Issues

The verification process influences each stage of the system development life cycle. Project cost estimates, schedules, system architecture, and implementation must each be approached with the consideration that the system will be verified. Because of this life cycle influence, management of a project involving verification must pay particular attention to verification.

The direct fiscal consequence of verifying a system is tied to the labor intensive verification effort itself. There are several indirect consequences of verification, however. Some of these are addressed below.

One indirect consequence of any verification effort is the additional configuration management requirements. First, it is important that tight control be kept over verified source code; running on the wrong version of source code is as good as running on source code that has not been verified at all. Second, the experience of this project, and apparently many other examples exist, was that at some point the verified version of the code had to be split off from the operational version of the code. This split occurred as a result of changes to the code to make the verification easier and more convincing, and to correct any code errors found during the verification.

Related to this extra configuration cost is the extra effort required to merge the two versions of the source code (operational version and verified version) upon completion of the verification. Ideally, this would be done as changes were made to the verified code. This would be manageable if the code changes required were minimal, and if the resources existed to continuously do the merging. During the Guard System verification, resources were redirected from system integration support to verification, requiring that merging the two code versions be done after verification was complete. Doing this stretched the integration schedule from that which would have existed had the code not been verified.

Guard System Final Report

Code changes required by verification have a greater cost impact than code or design changes at earlier stages of the project, because it may require another look at the existing verification arguments as well as the additional integration and configuration management cost. Design and code reviews throughout the development stages can minimize the additional cost and effort. Reviews can shake out errors early, when the cost to correct the errors is less. It also allows the design and code to be reviewed for ease of verification, preventing major design changes at later project stages.

The value of functional testing must not be overlooked because of verification. The analytic and empirical approaches of verification and functional testing must complement each other. Specific attention should be given in tests plans to security properties of the system. This enables:

- ◆ Provision for overlapping, thus preventing omission.
- ◆ Assurance that weaknesses in the verification (such as the correctness of the compiler) are covered by functional testing.
- ◆ Assurance that the system as a whole operates as the source code/hardware schematics indicate (i.e. that the assumptions made in the verification are valid).

Guard System Final Report

8. LESSONS LEARNED8.1 Project Management

" Sytek expended a great deal of effort in the formulation of the Functional and Software Specification documents which proved very helpful to those who wrote the actual Guard System software.

Staffing was a serious problem throughout the project as it was difficult to acquire highly qualified staff members due to an industry wide shortage. Actual software coding began in January of 1983 by one person. It was not until March 1983 that the remaining two project programmers were hired. The effects of a compressed development schedule were felt throughout the project until system delivery in December 1983.

8.2 Customer Coordination and Cooperation

The Government could have been much more helpful in correctly specifying the interfaces to their systems.

The Update Guard was nearly finished when it was found that input tapes were of a particular IBM format and that SELHDR and Linear records were intermixed within tape records. The knowledge of correct tape formats acquired late in the design and implementation of the UDG slowed its ultimate completion. For almost the entire UDG development time, the government supplied only one test data tape. Prior to acceptance tests in October, 1983 the Government supplied an additional tape of a differing format which lead to one UDG change.

Several versions of an Interface Control Document (ICD) were produced by the Government that were to explain in detail all electrical characteristics, necessary protocols, and higher level procedures necessary to connect the OLG to the Government's data base system. All versions contained profound errors and omissions. Finally the OLG was designed to operate according to the July 19, 1983 ICD and acceptance tests were performed in October, 1983. Also in October, Sytek personnel were provided access to the Government's systems and beginning with examination of electrical signals, completely rewrote the ICD to be correct and complete. No Government personnel had adequate knowledge of their system to define an accurate ICD, however some government personnel were quite helpful in assisting in the generation of an accurate ICD, and in further integration efforts.

Sytek was informed by the Government during confidence testing that there exists two different interpretations for the field DN.SO belonging to a linear record. Those performing confidence testing required that both interpretations be supported by the Guard System. During 1983 when such items were considered in Guard System software design and development, Sytek was unable to obtain assistance from Government personnel to explain the record

Guard System Final Report

formats which would have exposed this misunderstanding and omission.

, During much of the time Sytek personnel desired to perform on site Guard System integration, the Government failed to provide escorts. A great deal of time was lost waiting for on site access.

A general lack of communication and cooperation exists between different divisions of the Customer's organization which greatly hindered efficient design, development, testing, and integration of the Guard System.

Future efforts would greatly benefit by the coordination of Customer divisions spanned by a project such as this. The identification of a single individual responsible for coordination across different divisions would prove most beneficial.

8.3 Means to Test the Guard System

The original Guard System contract did not provide for a means to perform comprehensive tests of the Guard System. Consequently, as the Guard System was being developed, a parallel effort was performed to develop a Guard Test System (GTS). The GTS proved invaluable in the Guard debugging effort, and has enabled Government confidence testing without both the user network and data base machine connected to the Guard System at the same time.

8.4 Hardware Debugging

Very few problems were found in the Sytek developed AGB. The symptoms encountered were intermittent and for a period of time elusive.

The two problems found were an unsteady cpu clock circuit, and one digital part not operating within specified tolerances.

Guard System Final Report

8.5 Establishment of Development Environment

A development environment was not established prior to the start of software development. The development environment consisted of tools used to accomplish software development, debugging, testing, and integration. The result was that a parallel effort was performed by the same programming staff as was doing actual code development. This proved to be frustrating and an underutilization of resources.

The development environment that was built included the following:

- ◆ Automated source control and code module construction scripts.
- ◆ The Pascal cross compiler.
- ◆ The 8086 cross assembler.
- ◆ The 8086 cross linker.
- ◆ Formatting programs to transfer code from the VAX to the MDS.
- ◆ Formatting programs to massage code once on the MDS.
- ◆ MDS/ICE-86 on board code debugging and prom programming facilities.

The environment developed was slow and inconvenient to use. Debugging was a tedious process as no symbolic addresses were available. Much more effort should be spent in design and implementation of future development environments prior to the start of software coding.

An example of a desirable development environment might include a microprocessor development system that itself contains the means for source control, code compilation, code execution, symbolic debugging, and the ability to execute code without the need to program PROMS. Such systems are now commercially available.

Guard System Final Report

9. CONCLUSION

The successful completion of the Guard System demonstration project shows that information may be shared with assurance that no compromise in security can occur.

The Guard functions as a conduit for data flow between two separate computer systems or devices where there is information contained on one device or system that, under certain circumstances, cannot be shared with the other. The Guard enforces a predefined security policy without reliance on any services provided by the two computer systems or devices to which it is connected.

The hardware architecture of the Guard System provides complete isolation between the two systems to which it is connected. Verified software is employed on processors that control the passage of information from one computer system to the other.

The Guard System is built around a general concept that may be employed in any situation where passage of information between two electronic systems is allowed subject to a strict and absolute security policy.

This technology represents a dramatic departure from classical operations where human intervention was previously relied upon to control the dissemination of sensitive information.

Guard System Final Report

REFERENCES

- [1] "Security (Guard) Device System Functional Specifications," SYTEK TR-82001 and amendments thereto, SYTEK TR-82013, Sytek, Inc., Sunnyvale, CA (1982).
- [2] "Security (Guard) Device System Software Detailed Design Specifications," SYTEK TR-82010, Sytek, Inc., Sunnyvale, CA (1982).
- [3] "Security (Guard) Device System Hardware Detailed Design and Schematics," SYTEK TR-82014, Sytek, Inc., Sunnyvale, CA (1982).
- [4] "Security (Guard) Device System Test/Evaluation Plan and Procedures," SYTEK TR-82016, Sytek, Inc., Sunnyvale, CA (1982).
- [5] "Security (Guard) Device System Design Verification Report," SYTEK TR-82017, Sytek, Inc., Sunnyvale, CA (1982).
- [6] "SEALION AGB Hardware Specifications," SYTEK TR-82027, Sytek, Inc., Sunnyvale, CA (1982).
- [7] "RECON GUARD System Acceptance Test Procedures," SYTEK TR-83009, Sytek, Inc., Sunnyvale, CA (1983).
- [8] "Security GUARD Interface Control Document," SYTEK TR-84003, Sytek, Inc., Sunnyvale, CA (1984).
- [9] "Guard System Operation and Maintenance," SYTEK TR-84004, Sytek, Inc., Sunnyvale, CA (1984).
- [10] "Guard System Verification Report," SYTEK TR-84009, Sytek, Inc., Sunnyvale, CA (1984).
- [11] "RECON Security Guard System Functional Specification," SYTEK TR-84011, Sytek, Inc., Sunnyvale, CA (1984).
- [12] "RECON Security Guard System Software Design Specification," SYTEK TR-84012, Sytek, Inc., Sunnyvale, CA (1984).
- [13] "Guard System Hardware Detailed Design," SYTEK TR-84013, Sytek, Inc., Sunnyvale, CA (1984).
- [14] "The Best Available Technologies for Computer Security," Carl E. Landwehr, IEEE Computer Magazine, July, 1983.

Sytek Current Status Report
(TAB-064-84 dtd 12 July 1984)

Attachment F



12 July 1984
TAB-064-84

STAT

[Redacted]

1820 North Fort Meyer Drive
Arlington, VA 22209

STAT

Dear [Redacted]

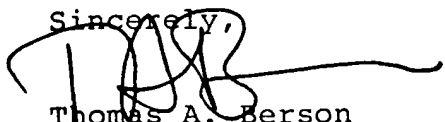
Sytek has been informed that the RECON Guard test team and OCR have completed testing successfully, and OCR has derived the requirements for an Operational Guard System.

The primary goal of this exercise was to prove the security functionality of the hardware architecture principles developed under this program. OS/ISSG was responsible for this security evaluation. The security test team completed their testing on 11 June 1984 with the findings that the prototype Guard System successfully performed the security functions required, thus validating the secure integrity of the design principles derived under this program. We understand that official documentation from the security test team and OCR will be made available to the RECON Guard project manager and subsequently to Sytek.

Our immediate efforts will now focus upon the verification briefings, and preparing for development of the Operational Guard System as directed by the government.

As per Amendment 07 to Contract 82F770400, prototype Guard Systems and test equipment have been returned to Sytek control for use in development of the Operational Guard System software. Having this equipment will allow us to more effectively proceed in the early stages of Operational Guard development.

We are awaiting formal approval to move this equipment to Sytek offices in order to reconfigure it in preparation for the Operational Guard System software development.

Sincerely,


Thomas A. Berson
Vice President
Data Security Division

TAB/kw

Enclosure



12 July 1984
TAB-064-84

STAT

[Redacted]

1820 North Fort Meyer Drive
Arlington, VA 22209

STAT

Dear [Redacted]

Sytek has been informed that the RECON Guard test team and OCR have completed testing successfully, and OCR has derived the requirements for an Operational Guard System.

The primary goal of this exercise was to prove the security functionality of the hardware architecture principles developed under this program. OS/ISSG was responsible for this security evaluation. The security test team completed their testing on 11 June 1984 with the findings that the prototype Guard System successfully performed the security functions required, thus validating the secure integrity of the design principles derived under this program. We understand that official documentation from the security test team and OCR will be made available to the RECON Guard project manager and subsequently to Sytek.

Our immediate efforts will now focus upon the verification briefings, and preparing for development of the Operational Guard System as directed by the government.

As per Amendment 07 to Contract 82F770400, prototype Guard Systems and test equipment have been returned to Sytek control for use in development of the Operational Guard System software. Having this equipment will allow us to more effectively proceed in the early stages of Operational Guard development.

We are awaiting formal approval to move this equipment to Sytek offices in order to reconfigure it in preparation for the Operational Guard System software development.

Sincerely,

Thomas A. Berson
Vice President
Data Security Division

TAB/kw

Enclosure