Fortezza-enabled Multi-level Sensitive Simulations

Eric A.M. Luiijf M.Sc.Eng.
TNO Physics and Electronics Laboratory (TNO-FEL)
P.O. Box 96864, 2509 JG The Hague
The Netherlands, +31 70 374 0312
luiijf@fel.tno.nl

Amitabh Dey, James Watson
SPARTA, Inc., Orlando, FL
amitabh_dey@sparta.com
jwatson@sparta.com

Carl Muckenhirn
SPARTA, Inc., Columbia, MD
Carl_Muckenhirn@sparta.com

Mike Garnsey
STRICOM, Orlando, FL
Mike_Garnsey@stricom.army.mil

Keywords:
DIS, Fortezza, ISDN, MISSI, multi-level, SABI, security, sensitive, simulation, TIBI

ABSTRACT: Current security models in simulation environments depend on interconnected System High enclaves. A secret and below interconnect (SABI) using a guard might interconnect the various enclave classification levels. Problem areas include: restricted "need-to-know" (privacy/sensitivity) support, the dependence on expensive or capacity-limited link encryptors, the difficulty to clear all personnel involved in a multi-national and/or large exercise, and the throughput limitations associated with centralized guard mechanisms.
In the joint Sensitive Simulation (SENSIM) project STRICOM, TNO-FEL and SPARTA Inc. researched the use of application level Fortezza® end-to-end encryption in a multi-level sensitive but unclassified (SBU) secured simulation environment. The use of digital signatures to authenticate and protect the integrity of simulation management information has been studied as well. This paper discusses the background of this new approach, the international experiments using an ISDN link, the initial results, improvements made, conclusions and recommendations for further study. This project was sponsored jointly by the US Army STRICOM and the Royal Netherlands Army (RNlA).

1. Introduction

To date, all USA and international efforts in securing distributed interactive simulations (DIS and HLA) are targeted towards network level security services, e.g., data link encryption between multiple system high secured environments, the so-called *enclaves.*

Current DMSO plans for the security architecture in the High Level Architecture for Simulations (HLA) rely upon system high environments and guards as sanitizing boxes between networks at different classification levels [2]. The reason is that multi-level security (MLS) in DIS and HLA environments is a complex technical and organizational issue. From [2], it is not expected that these MLS solutions will be accredited under the demanding security agency process for at least the next few years.

Within these System High environments, data flows unencrypted through the network. Under certain circumstances one wants to protect sensitive information in simulators within the System High environment itself. Examples include stealth plane simulators, capabilities of advanced weapons systems and tactical doctrine.

In order to meet the demands of simulation architectures such as the DMSO HLA, security should be exercised closer to the actual consumers and producers of sensitive and potentially classified information. Eventually, support of multi-level simulations, that is a single simulation operating on, and producing, information at more than one security level (classification, compartment, caveat) will

require the simulation (or a simulation framework) to directly access or perform security services.

The US National Security Agency's (NSA) Multi-level Information Systems Security Initiative (MISSI) or its Dutch equivalent, the Technisch Informatie Beveiligings Initiatief (TIBI), are producing technical security solutions which may be applied to the modeling and simulation environment. Of particular interest is use of the US Fortezza® cryptocard and the accompanying Fortezza cryptologic library to secure modeling and simulation applications. For background information on Fortezza, see [5] and [6].

STRICOM, SPARTA and TNO-FEL jointly developed a new approach to the aforementioned security problems by introducing application level security in the system high simulation enclave environment. Our Sensitive Simulation (SENSIM) project included joint experiments to demonstrate the approach in a multi-level sensitive, but unclassified (SBU) test environment. This environment comprised of one LAN at STRICOM in Orlando and one LAN at TNO-FEL in the Netherlands, interconnected via ISDN, with ModSAF computer generated force simulations using DIS protocols.

The objective of this paper is to describe the new concept and demonstration of multi-level secure encryption at the application level supporting a multi-level secure distributed simulation. Results achieved are presented and discussed, leading to issues and directions for future work.

The concept and intentions of the SENSIM project have been presented earlier at the ITSEC'97 conference [1]. This paper extends [1] with results and observations.

2. Statement of the problem

For some time, various security issues have limited training and simulation exercises. These have ranged from pedestrian concerns about the performance of particular security equipment, to critical concerns of classification and security accreditation mismatches between environments. Segregation and isolation of information among exercise participants is also a major concern. Segregation may be needed due to national policies (i.e., US or NL ONLY restrictions) or proprietary concerns.
In discussions with SPARTA and TNO-FEL, STRICOM expressed a desire to provide a security solution which will allow US and allied simulation activities to execute cooperative exercises containing sensitive information over unprotected networks. In particular, STRICOM is working with its British and Dutch counterparts and would like to provide a method of protecting information exchanged among the parties as well as allowing parties to restrict access of the other parties as needed.

The immediate problem is to provide a sensitivity (privacy) protection mechanism, to US and non-US allied simulation activities. This mechanism must provide to the participants the ability to securely share information which may ultimately be classified information.

3. Current security approaches

3.1 System High and Dedicated

Current practices within the defense modeling and simulation community for protection of classified activities fall into three basic camps. The first is the traditional defense isolation of the classified computation within a physically secured enclave. The systems, if more than one is involved, are interconnected via standard networking technologies and they all operate in a "System High" or "Dedicated" mode (see Figure 3.1).

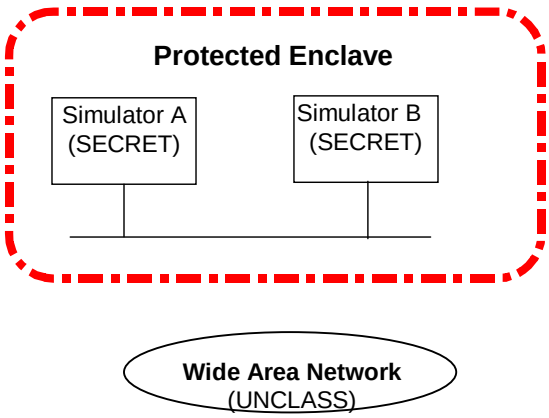


Figure 3.1: System High Protected Enclave Network

This approach requires that all participants in the exercise be cleared to the same level and are at one single location. For many activities this is not an issue, but for large exercises, requiring many players, possibly from allied or coalition forces, the ability to clear all players in time for all information in the exercise may not exist.

The next approach basically extends the boundaries of the "protected enclave" through the use of network encryption devices on a point to point basis. In this case, again, all simulations operate at the same classification level in either a "System High" or "Dedicated" mode (see Figure 3.2). This model can provide good performance for distributed simulations, but at the expense of dedicated telecommunications circuits between participating

enclaves. Note that a separate channel is required between each site with two encryptors on each link.

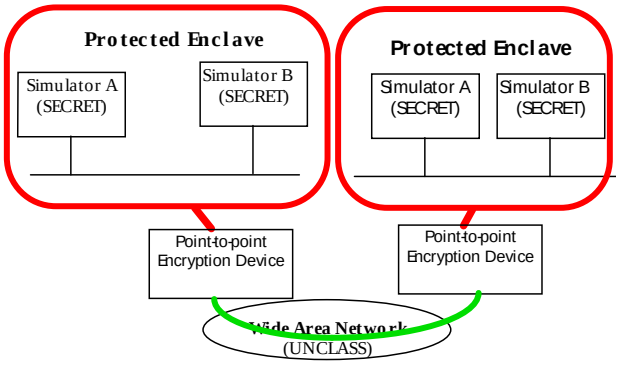


Figure 3.2: System High Interconnected Enclaves

The third security option is that employed e.g. by the Defense Simulation Internet (DSInet). This approach is basically an extension of the second approach with protected enclaves interconnected over networked connections rather than point-to-point connections. Theoretically this architecture provides the ability for distributed simulations to interact with an arbitrary number of protected enclaves through the use of a single network connection using multi-casting protocols and a single network encryption device. However, in addition to the currently experienced performance problems with the encryption devices, this architecture still does not provide any protection beyond the “enclave” level.

3.2 HLA security architecture

A special security task force is targeted to develop a security architecture for the High Level Architecture (HLA) for Simulation. Several projects were granted by DMSO to develop a multi-level security HLA environment. Multi-level security in the HLA environment means that a *secret and below interconnect* (SABI) guard between a high classification level *System High* environment, e.g. Secret, and a lower classification level, e.g. Unclassified, secures the information flow between both environments.
Only allowed and sanitized traffic flows should pass the guard. Trusted Information Systems (TIS) is involved in the study and design. TIS presented the diagram in Figure 3.3 at the 1997 Spring Simulation Interoperability Workshop [2]. As shown the guard secures the information flow between two classification levels, each being a system high or dedicated environment. Note that the connection arrow near MRCI stands for a bulk encrypted wide area data link solution.

The sanitizing job of the guard is difficult. Different views of the "same" information depending on one's classification level or need-to-know in a distributed simulation environment have to be presented. Solving this complex information problem in such a way that a general solution can be accredited by security agencies is estimated to take several years. Note that this security architecture has a pre-requisite System High or dedicated local and/or wide area network for each security level. However, within one single security domain no solution for multiple segregated sensitivity compartments has been presented.

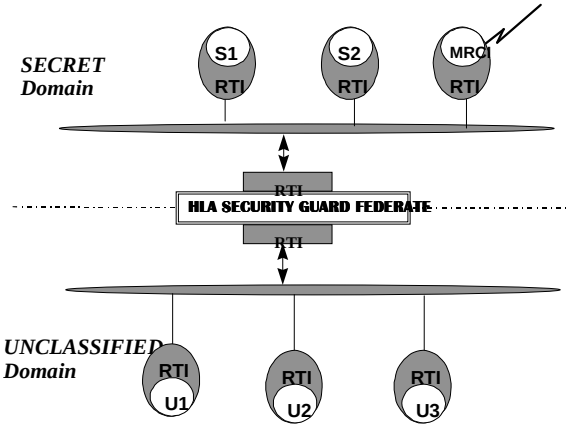


Figure 3.3: HLA Secured Combined Federation Architecture (source: [2]).

4. The Multi-level Information Systems Security Initiative (MISSI)

NSA's Multilevel Information Systems Security Initiative (MISSI) and the Dutch TIBI aim at providing security solutions to a wide variety of information systems applications. Implementation of the security solutions is accomplished through the use of several “building block” security products.

PC Card (PCMCIA) format cryptographic devices which were used in our project provide data encryption and decryption, data integrity, user identification and authentication, and user non-repudiation functions (sign/verify, hash, time stamp). See Table 4.1.

Table 4.1: Overview of Fortezza functions and bit and key lengths

Cryptocard Function	Name	Description	Standard
Public Key Exchange	KEA	Key Exchange Algorithm 160 bit private key	Fortezza Diffie-Helman variant
Message Encryption	SKIPJACK	Type II Algorithm 80 bit key	NSA / FIPS 185
Digital Signature	DSA	Digital Signature Algorithm 1024 bit modulus	NIST FIPS 186
Hashing	SHA-1	Secure Hash Algorithm - Rev 1 160 bit	NIST FIPS180-1
Timestamp	N/A	Uses Secure Hash Algorithm Digital Signature Algorithm (DSA) 160 bit	FORTEZZA
Password	PIN	4-12 bytes Personal Identification Number	FORTEZZA
Certificate	N/A	Fortezza 2820 bytes	CCITT X.509

Table 4.2: The Fortezza Cryptologic Interface (CI) Library functions (from [5])

Library Commands
CI_Initialize
CI_Terminate

Management Commands		
CI_Close	CI_Lock	CI_Select
CI_GetConfiguration	CI_Open	CI_Unlock
CI_GetState	CI_Reset	CI_FirmwareUpdate

Cryptologic Commands		
CI_ChangePIN*	CI_GetHash	CI_Save
CI_CheckPIN	CI_GetPersonalityList	CI_SetConfiguration
CI_Decrypt	CI_GetStatus	CI_SetKey
CI_DeleteCertificate	CI_GetTime	CI_SetMode
CI_DeleteKey	CI_Hash	CI_SetPersonality
CI_Encrypt	CI_InitializeHash	CI_SetTime*
CI_ExtractX*	CI_InstallX	CI_Sign
CI_GenerateIV	CI_LoadCertificate	CI_TimeStamp
CI_GenerateMEK	CI_LoadDSAParameters	CI_UnwrapKey
CI_GenerateRa	CI_LoadInitValues*	CI_VerifySignature
CI_GenerateRandom	CI_LoadIV	CI_VerifyTimestamp
CI_GenerateTEK	CI_LoadX	CI_WrapKey
CI_GenerateX	CI_RelayX	CI_Zeroize
CI_GetCertificate	CI_Restore	

* Site Security Officer only function

There are two basic versions: Fortezza® for use in sensitive but unclassified (SBU) environments and Secret environments under certain circumstances; and Krypton® for all levels of classification, with implementation restrictions.

A wide variety of applications have been enabled to use the Fortezza card, including: e-mail, file transfer, storage, EDI/E-Commerce, search & retrieval, dbase access authentication and WWW. All these applications make use of a single software interface, the common cryptographic application program interface or CAPI. On each platform that supports Fortezza the CAPI is supported by a cryptologic library, the CI_lib.

5. Application security, a new approach

Rather than securing an exercise at the communications network level, and accepting the drawbacks embodied in that approach, (performance bottleneck, single-level system high or dedicated exercises), we propose to provide security at the simulation application level through the use of MISSI (or for the Netherlands: TIBI) technology. The current MISSI approach is to migrate security services from application independent areas (such as a communications network) to application dependent areas for systems on federations which are capable of, or require, identification and segregation of information. Distributed combat simulation network could be such a case.

The SENSIM project was drafted to explore the efficiency and problems associated with providing security services at the simulation application to network interface using Fortezza technology. Figure 5.1 is an adaptation of a figure from [4], which depicts encryption/decryption of information at the application interface by application calls. Placing security at this level allows the simulation host (and its applications) to segregate and protect data based on simulation/exercise specific security policies. In the current environments, such an architecture may be used to isolate various categories of information at the same security level (i.e. caveats within SECRET). Migration of the simulation host operating system to a trusted platform (such as a Compartmented Mode Workstation) will allow segregation of information of several security levels.

The intent is that the protected sensitive simulator (or federate) is able to acquire information of both insensitive and (other) sensitive simulators. The sensitive simulator is able to keep in touch with sensitive cooperative simulators by using security devices that belong to the same group (having the same key). At the same time, the simulator is able to sanitize information (e.g. entity state) and make that available to all simulators.

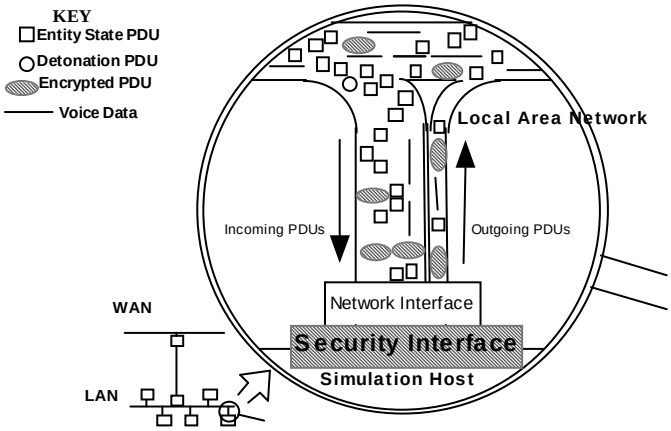


Figure 5.1: Application-level secured simulation to network interface

Technical implementation of this architecture can be achieved by integrating Fortezza security services into the DIS Interface Library (DIL) or the HLA Run Time Interface (RTI). Integration at this level will allow security services to be applied directly from the simulation (through an expanded security interface), on a host or class basis.

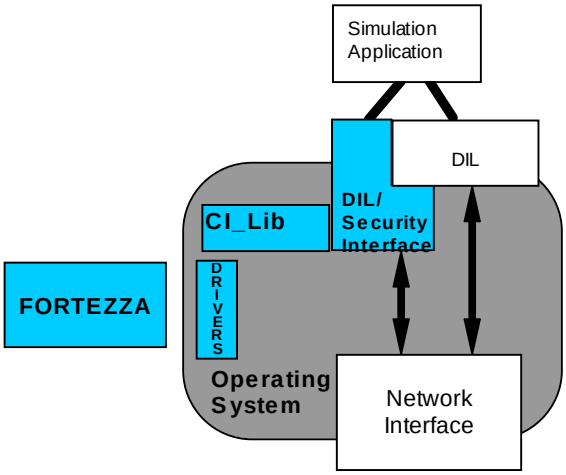


Figure 5.2: Fortezza enabled DIL-interface

Initially, in the SENSIM exploratory project, the system performs security services on a per DIS PDU-type basis, that is, all PDUs of a particular type being sent, will be protected in the same manner (e.g. encrypted). This will allow further unmodified simulations (such as MODSAF) to exchange information through a Fortezza protected channel.

Integration of Fortezza with the DIS Interface Library is quite similar to other Fortezza integration efforts to date. In the DIS-case, the underlying UDP protocol being secured is "stateless" with each "package" being an atomic event. This eases development since no persistent protocol state must be kept on each package as it is transmitted, and all information required to determine the security services is included in the package. On the other hand, additional overhead is introduced which might conflict with the near real-time performance requirements. The project investigated whether such drawbacks are available or not.

The architecture of a Fortezza integrated simulator is shown in Figure 5.2. Most of the security components are non-developmental items. The only development required is in the DIL/Security interface, which interfaces the security services with the DIS Interface Library or the RTI in the HLA environment. The scope of work for the DIL/Security Interface entails designing the secured DIS PDU format, developing the interface mechanism/API between the DIL and Fortezza subsystem, and developing the "security API" which may be exported to simulation applications.

6. Proof of concept and expectations

6.1 Demonstration setup

In order to examine issues surrounding this security architecture, SPARTA in conjunction with STRICOM (US) and TNO-FEL (Netherlands), used multiple MODSAF semi-automated force generators in simple multi-level secure scenarios.

Figure 6.1 shows one of these scenarios, using RED, BLUE and GOLD players. The system was set to allow RED forces to interoperate and hide and protect certain information from the BLUE forces and *vice versa*. The GOLD players were able to receive and interact with both sets of forces. Segregation of information was enforced by selective distribution of cryptographic keys to the parties. In particular, the RED forces only exchanged keys with other RED forces and the GOLD players; similarly, the BLUE forces exchanged keys with other BLUE forces and the GOLD players.

The demonstration addressed several important Modeling and Simulation issues. The first objective was the analysis and testing of the Fortezza system performance in distributed simulation applications. Measurement of the throughput, delay, and overhead characteristics of a DIS PDU level security system was made. Secondly, the

project explored all issues related to the use of such "multi-level" secure simulations.

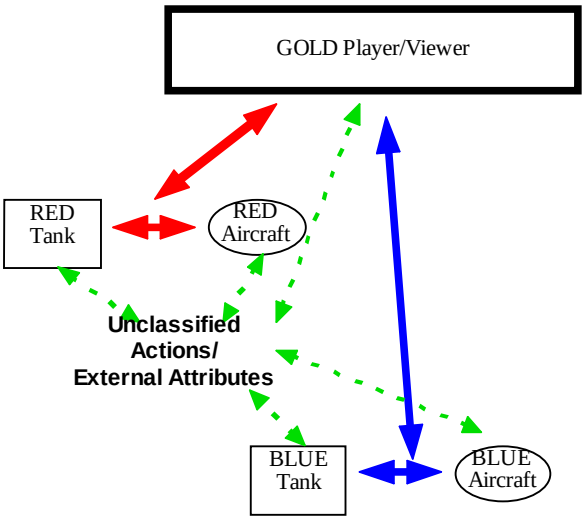


Figure 6.1: Multi-level Secure scenario

6.2 Fortezza speed expectations

SPARTA made a pre-demonstration assessment on the performance of Fortezza® PCMCIA cards and achieved an average command time of 282 ms, zero byte message time was 51 ms and a throughput of 2.4 Mbps using a SUN workstation and an external SCSI PCMCIA-reader. The throughput was limited by the commercial PCMCIA card interface not the Fortezza processor (which at 40Mhz and 8 bits per clock cycle runs at 320 Mbps). An average PDU generation rate of approximately 10-15 PDUs per second for a pair of tanks during a ground engagement was assumed based on prior observed DIS results. These figures were conservatively taken to be all entity state PDUs of 1.528 kbit message length.

TNO-FEL also made some throughput measurements on the Fortezza card using the *ftzatest* program supplied by SPARTA on a SGI Indigo2 with a SCSI PCMCIA card reader. Two different types of measurements were made: a) executing the encryption and the decryption processes in such a way that the Fortezza key management functions are only called once per measurement, b) execute the Fortezza key management functions for each encrypt/decrypt cycle.
The initialization cryptographic functions are CI_DeleteKey, CI_GenerateMEK (master encryption key), CI_SetKey and CI_GenerateIV (initialization vector). For obvious reasons process a) is the fastest. By the way the CI_Encrypt and CI_Decrypt functions only act upon tuples of 64 bits (8 bytes) using the (default) 64 bit Cipher Block Chaining (64 bit CBC) mode. Encryption requires thus the data to be padded up to the next multiple of 8 bytes.

From Appendix Figure A one can see that the time per encryption or decryption call for a block of information is nearly the same if no additional calls are necessary for changing session keys (encrypt2/decrypt2 cases). For block sizes up to 4 Kbytes the time per call increases is quite flat: a delay of 46.4 ms plus roughly 2 ms/Kbyte. Just above 4 Kbytes, an unexplained increased time/call jump occurs. Note that there is an asymmetric effect between encryption and decryption.

In our demonstration, we used the Fortezza card to encrypt and decrypt DIS IEEE 1278.1 protocol data units (PDUs). PDUs were required to fit into one UDP Ethernet packet, limiting the PDU size to a maximum of 1500 bytes. The encryption and decryption speed for this "working range" was expected to be just over 46 ms, a time mainly required by the cryptologic library to verify the PCMCIA card status, to initialize it for encryption/decryption as well as the SCSI driver/interface overhead.

Because of the relatively slow Fortezza CI_lib library initialization time, the encryption/decryption throughput more or less doubles with the block size (see the almost straight logarithmic line in Appendix Figure B). The speed doubles up to the block size of 9 Kbytes. For larger block sizes a slow down occurs, probably due to non-Fortezza issues (e.g. memory management).

6.3 Fortezza aware ModSAF

Rather than encrypting the content of all DIS protocol data units or PDUs, SENSIM tried to encrypt only the information of stealth entities. Technically, an additional functionality has been added to ModSAF's low level PDU-to-network interface. Both the encryption and decryption flows are depicted in Figure 6.2 respectively Figure 6.3.

The current experimental code determines the fact that an ESPDU is of the stealth type by looking at the size of the ESPDU. In case an arriving ESPDU is larger than 200 bytes, the PDU will follow the decryption path. As optional articulation parameters (N*16 bytes) might cause an ESPDU to become larger than this discriminating size, in future new encrypted versions of the PDU types need to be defined.

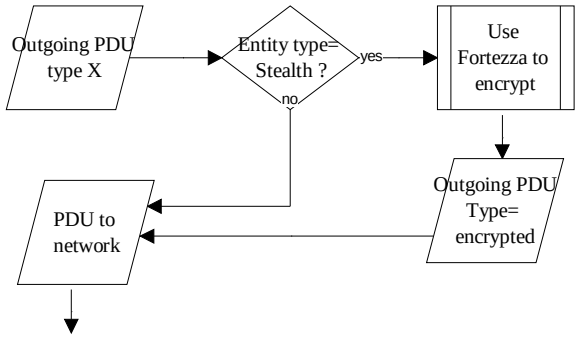


Figure 6.2: Processing Fortezza enhanced outgoing PDU stream and encrypting stealth entity information

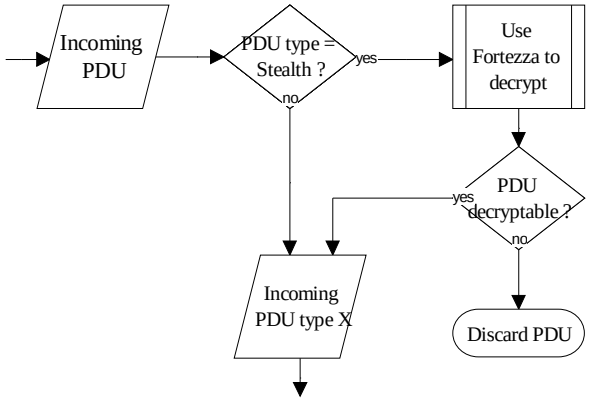


Figure 6.3: Processing Fortezza enhanced incoming PDU stream and decrypting stealth entity information

The layout of encrypted ESPDU is shown together with the standard IEEE 1278.1 ESPDU in Appendix Figure C. Because of UDP delivery is not guaranteed (PDUs might not arrive) cryptographic information needs to be added to the ESPDU. This includes the 128 bytes Ra, which is the encryption random number necessary for the Key Exchange Algorithm (KEA); the 24 bytes Initialization Vector (IV), and 12 bytes Message Encryption Key (MEK). In total, the stealth ESPDU is 164 bytes larger than a standard ESPDU. From Appendix Figure C one can see that the ESPDU more than doubles in size when containing encrypted information.

6.4 Proof of principle

The first intent of the SENSIM experiments was to show a proof-of-principle of application based security. The Fortezza-aware ModSAF code was developed by Sparta under contract to STRICOM. For reasons of simplicity, three different versions of ModSAF were built: ModSAF-E (encrypting), ModSAF-D (decrypting) and ModSAF-U, was a "Fortezza unaware" version that had to disregard the encrypted ESPDUs that were longer in size.

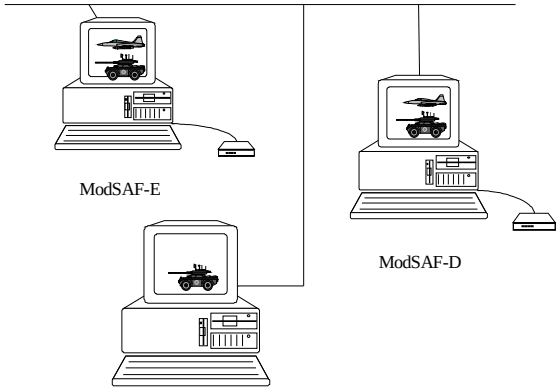


Figure 6.4: The "stealth" entity (e.g. plane) is not seen by normal DIS-players (ModSAF-U)

At the encrypting side, the public key of the decryptor is required. This key had to be extracted from the "personality" certificate loaded in the Fortezza card of the decrypting side. In the same way, at the decrypting side needed the public key from the encryptor to unpack the session key. This required modifying a X.509 certificate extraction program to extract the public KEA and DSS keys.

With only minor problems being encountered, we connected these ModSAF variants across the North Atlantic using an ISDN-link. On the first attempt, a backdoor was discovered. ModSAF also issues Synthetic Environment Persistent Object (PO)-database information on the network, allowing the ModSAF-U to learn about the existence of the sensitive entities. Turning off the PO-switch at the start of the ModSAF-E resulted in the intended project result (see Figure 6.4).

7. Results of the experiments

7.1 Stealthy performance

TNO-FEL instrumented the encode and decode ModSAF versions. At the relevant places in the code wall-clock time calls were made. Analysis showed 116 ms encryption initialization time and 575 ms needed for encryption and key wrapping. This summed up to 692 ms on average. The decoding was only a little faster with 556 ms resulting in **1.2 seconds per transferred ESPDU**. These results were far from the expected results and in no way match the IEEE 1278.2 DIS maximum latency of 100 ms (tightly coupled) and 300 ms (loosely coupled). [7]

7.2 Performance improvements

In attempt to identify the cause, TNO-FEL extracted the essential code parts to avoid the lengthy recompile and linking time of ModSAF. Two programs were developed: one doing the encryption and writing PDUs to a file and another doing the decryption. Each Fortezza CI_lib call was measured. Appendix figure D contains the results.

Inspection showed that the Fortezza code encoded each ESPDU with a *new* "session key". The process, requiring a new random number and IV is very time consuming, as can been seen in Appendix figure D. Unfortunately, due to the stateless nature of the UDP protocol, an encrypted session "stream" of packets, sending the session key "envelope" only at the start, cannot be setup. As part of each ESPDU, the public key of the recipient wrapped session key needs to be sent in each UDP packet.

Rather than generating it each time again, a session key can be generated only once at the start of ModSAF-E. Then add the (now) static 164 bytes (Ra, IV, and MEK) encryption session information need only to be included in each encrypted ESPDU. The encryption routine would encrypt only the "plain" entity state data. Experimental code showed that this saves about 650 ms per ESPDU at the generator (encrypting) side.

In the recipient code, there is another problem. As more senders transmit encrypted ESPDUs, the Ra, IV, MEK-combination for each is different. Some number of Ra, IV, MEK sets can be cached at the receiving side. New or out-of-cache ESPDUs require the whole sequence of setting the proper encryption registers. Switching the decoder between different stealth senders can be based upon the Ra, IV, MEK-combination and an accompanying assigned Fortezza card register set with CI_SetKey. Shortcuts in the code determine whether the key is already "on the hook" save nearly 500 ms in the decoding process for a stealth ESPDU.

These code improvements lead to an encryption time per ESPDU packet of 46.5 ms for the second and subsequent packets, unless the initialization and random values need to be changed for cryptographic reasons.

The decoding process now requires 46.5 ms when the received packet comes from the same stealth sender as the previous (encrypted) one. When another (known) stealthy sender sent the packet, a context switch is required. The decryption then takes 100 ms. In case a packet arrives from an unknown sender, 100 + *n**350 ms is required. The number n is the number of tries needed to find the correct key in the public key ring of the current exercise.

7.3 Packet loss and the cryptographic engine

In working with the encryption and decryption programs, we simulated the drop of a UDP packet, and another issue became apparent. As the standard Fortezza Skipjack algorithm uses the encryption/ decryption outcome of previous blocks, no packet loss can be tolerated. This contradicts the DIS type of exercises where UDP packets might be lost and simulators (federates) might join at any time.

The only way to solve this in a non-reliable data stream environment is to use weaker cryptographic algorithms, e.g. ECB, that can recover immediately or after dropping only one packet. Using such a weaker algorithm however, might require more frequent changes of the random values. This requires renewal of the algorithm initialization values, meaning an interrupt of *1.2 seconds* of the exercise.

7.4 Missing group key concept

The Fortezza® encryption library and key management system is designed for a secure one-to-one communication path. In the Modeling and Simulation community, one needs to talk securely to groups of (stealthy) simulators.

Fortezza currently does not support a 'group key' concept. That means that each 'sender' and 'receiver' pair need to exchange public keys. Moreover, the number of random values plus keys-sets that can be loaded into the Fortezza card is limited to eight. This limits the usability of this solution at the present moment.

However, we were able to trick the Fortezza library and reloaded the saved initialization materials of another card. That meant that multiple senders could encrypt for the same receiver.

7.5 Signatures

To avoid that one can spoof simulation management information in a large exercise or the content of other data packets, TNO-FEL investigated the idea of using digital signatures and signature verification of for instance DIS SIMAN PDU's as well. The current slowness of Fortezza digital signature calculation (137 ms) and digital signature verification (213 ms), apart from the time required for hashing the PDU data, currently does not make this idea worthwhile in a near real-time M&S environment.

8. Conclusions

From our experiments, we conclude that:

1. The principle of our new approach of application based secured simulators and federates is valid.
2. The Fortezza card and library are designed for the secure transfer of single messages as Email (setup time is hardly noticed) and for data streams as file transfer (high speed obtained after a slow initialization). Fortezza currently seems *to be poorly designed* for near real-time traffic with many short messages and switching cryptographic key information for each message.
3. Fortezza *lacks* a 'group key' concept which is ideal for the Modeling and Simulation community. We recommend NSA implement and document such a concept within the MISSI products.
4. The maximum number of 8 active key sets in the Fortezza is far too low for large-scale M&S exercises.
5. The security community should discuss which encryption algorithms are allowed to be used for the type of security we tried to demonstrate. The problem of the recovery of 'lost packets' in the stream should be addressed.
6. The Fortezza digital signature and signature verification process is too slow for use in the near real-time M&S environments.
7. The security community should look into key management requirements and the frequency of initialization vector changes for these encryption algorithms. A trade-off between algorithm speed and the near real-time requirements of the Modeling and Simulation community should be made.

9. Suggested future work

First of all, the Fortezza performance problems can be expected to improve greatly when new versions of the Fortezza card become available having a high-speed Cardbus™ interface. Whether the expectations about this version hold in the near real-time Modeling and Simulation community environment as well, has to be validated.

Secondly, from [6] we understood that a software version of Fortezza is to appear. That might be faster and very useful in the simulator environment. Experiments should be conducted, e.g. using our test programs, as soon as that version becomes available.

NSA needs to implement a group key concept for the Modeling and Simulation environment.

The implementation of application level security into the HLA RTI requires a separate study. The overhead per transferred packet might be even factors higher than that within the DIS environment. The use of reliable communication paths, however, might be beneficial in respect to the encryption algorithms as an encrypted 'stream' concept can be maintained.

10. References

[1] Muckenheim,C., Dey,A., Correa,H., Garnsey,M.: “Multi-Level Secure Encryption For Distributed Simulation - Application of Fortezza to DIS”, 19th I/ITSEC, Orlando (FL), USA, December 1-4, 1997.

[2] Filsinger, J., "HLA Security Guard Federate.", 1997 Spring Simulation Interoperability Workshop, Paper No: 97S-SIW-163

[3] Filsinger, J., "A Security Engineering Process for the HLA.", 1997 Spring Simulation Interoperability Workshop, Paper No: 97S-SIW-041

[4] "DIS Vision", 13th Workshop for Standards of Distributed Simulation, September 1995

[5] Valuable public information on the Fortezza card: (photo of inside) http://www.litronic.com/images/-products/fortezza/stack.GIF; (documentation) http://www.rnbo.com/PROD/rmadillo/e/etoc.htm; http://www.armadillo.huntsville.al.us/Fortezza_docs/-index.html (e.g. the Fortezza Cryptologic Interface Programmers Guide version 1.52, jan'96); http://www.spyrus.com http://www.rnbo.com/mykoweb/index.htm http://www.nsa.org:8080

[6] See Missi and Fortezza conference articles at Missilab, http://beta.missilab.com

[7] IEEE 1278.1:1995 plus Annex:1997: Standard for Distributed Interactive Simulation - Communication Services *and* IEEE 1278.2:1995: Standard for Distributed Interactive Simulation - Application Protocol

Acknowledgments

This project was sponsored by the US Army STRICOM and the Royal Netherlands Army (RNlA) and was conducted under the umbrella of the Mutual Weapons Development Master data Exchange Agreement between the US AMC/STRICOM and the RNlA/DMKL (DEA Annex A-94-TN-1529).

The authors like to thank Micha Bloem and Lex Beijk (TNO-FEL) for their contributions to the project.

Author Biographies

ERIC LUIIJF, M.Sc.Eng. is a principal research consultant Telematics and Information Security at TNO-FEL in the Netherlands. He was the network architect for the first European DIS-demonstrations during two ITEC conferences in The Hague, using ISDN connections to link remote sites. He contributed to a joint STRICOM/TNO-FEL project on time synchronization of DIS-systems in Orlando with systems in The Hague (7243 km) using NTP and GPS. He is currently involved in end-to-end information security projects and the coordination of an information warfare-defense program.

AMITABH DEY, MSME, is the chief engineer for the Distributed Interactive Simulation Division of SPARTA, Inc.'s Information Systems Sector in Orlando, FL. He was the lead engineer on the integration of FORTEZZA to ModSAF portion of the joint STRICOM/TNO-FEL Multi-level secure project. He is currently involved in systems architecture modeling for JSIMS.

JAMES WATSON, PhD Eng. Mech, is the manager of the Distributed Interactive Simulation Division of SPARTA, Inc. Information Systems Sector in Orlando, FL. He was program manager of the STRICOM MLS concept study and developed the initial experimental demonstration approach.

CARL MUCKENHIRN, MS, is the lead engineer on SPARTA's MISSI development support to the US National Security Agency. He provided key concepts developed in the demonstration and principal technical coordination with NSA for the loan of the FORTEZZA technology in the SENSIM experiments.

MIKE GARNSEY is the STRICOM COTR and international liason for the MLS Concept Development BAA project.

Explanation of security terms

Compartmented Mode Workstation (CMW): a CMW is a workstation that conforms to the Trusted Computer Security Evaluation Criteria (TCSEC) B1 and parts of the B2 and B3-level features. It is a secure and a flexible multi-level, multi-compartment information processing networked system.

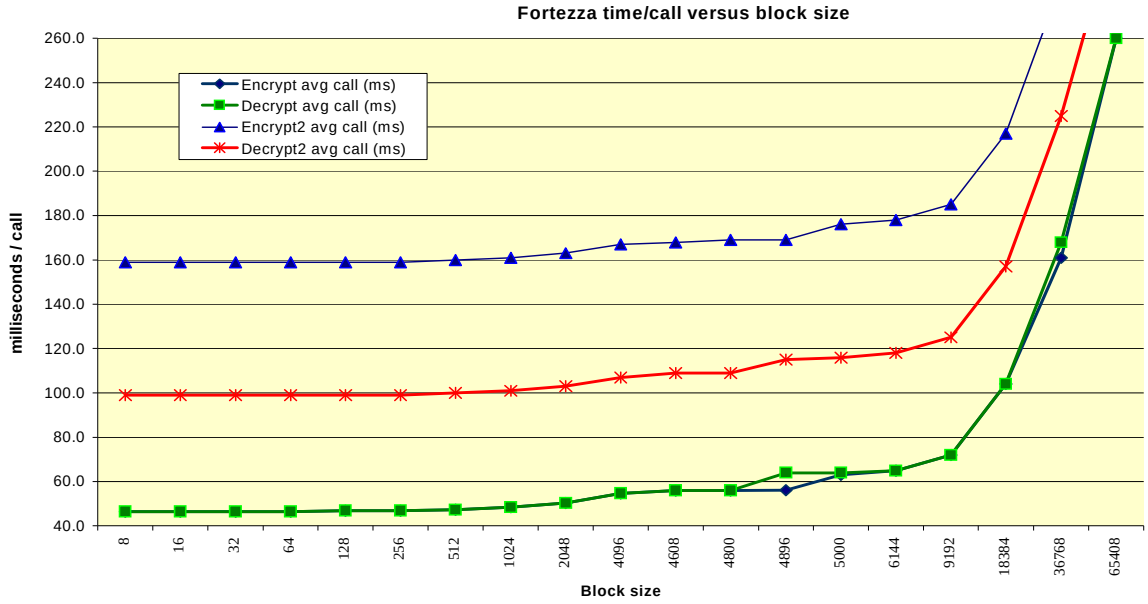
Dedicated: a security environment with only one classification level where all people are cleared to that data security level.

Multi-level security (MLS): a security environment able to process multiple classification levels simultaneously while some users are not screened to all levels of information in the system.

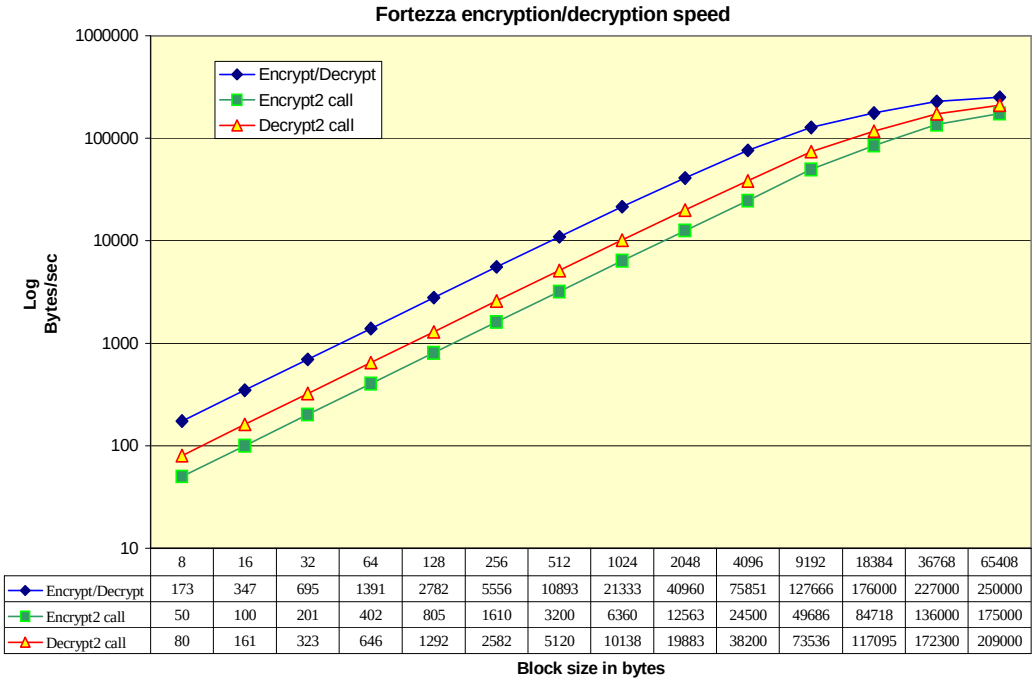
Secret and below interconnect (SABI): a trusted device interconnects a secret domain with a security domain of lower classification in a secure way [6].

System High: a security environment with multiple levels of data sensitivity where all people are cleared up to the highest data security level within that environment.

APPENDIX



Appendix Figure A: Fortezza encryption and decryption speed for time/call



	8	16	32	64	128	256	512	1024	2048	4096	9192	18384	36768	65408
Encrypt/Decrypt	173	347	695	1391	2782	5556	10893	21333	40960	75851	127666	176000	227000	250000
Encrypt2 call	50	100	201	402	805	1610	3200	6360	12563	24500	49686	84718	136000	175000
Decrypt2 call	80	161	323	646	1292	2582	5120	10138	19883	38200	73536	117095	172300	209000

Appendix Figure B: Fortezza encryption and decryption speed in bytes/sec

IP header + checksum		26 bytes
UDP header + checksum		16 bytes
ESPDU header (12 bytes)	Protocol	04 = DIS 2.0
	Exercise id	1 byte
	PDU type	01 = entity state
	Protocol family	01=entity information
	Time stamp	4 bytes
	Length	2 bytes 342 (0x156)
	Padding	2 bytes
Entity ID		6 bytes
Encrypted entity state 126+2 bytes + N*16 bytes	Includes 2 padding bytes necessary for Fortezza	
Ra 128 bytes	Random variable	Session key information
IV 24 bytes	Initialization vector	
MEK 12 bytes	Message encryption key	

IP header + checksum		26 bytes
UDP header + checksum		16 bytes
ESPDU header (12 bytes)	Protocol	1 byte (04 = DIS 2.0)
	Exercise id	1 byte
	PDU type	01 = entity state
	Protocol family	01=entity information
	Time stamp	4 bytes
	Length	2 bytes 132 (0x090)
	Padding	2 bytes
Entity ID		6 bytes
Entity state information 126 bytes	Entity state information in clear	
Articulation parameters	N* 16 bytes (N= 0..)	

Stealth ESPDU	326 + N*16 bytes excl. IP header	**Max. 104% overhead**
	352 + N*16 bytes incl. IP header	**Max. 89% overhead**

ESPDU in clear	160 + N*16 bytes excl. IP header
	186 + N*16 bytes incl. IP header

Appendix Figure C: Layout of encrypted ESPDU versus the standard ESPDU

	Timing	Used by ModSAF main.c	Used by ModSAF-E	Used by ModSAF-D
CI_Initialize (1)	78 ms	**X**	**X**	**X**
CI_GetConfiguration (2)	1330 ms	**X**	**X**	**X**
CI_GetState	7 ms			
CI_Open (3)	33 ms	**X**	**X**	**X**
CI_CheckPin	505 ms	**X**	**X**	**X**
CI_GetPersonalityList	48.5 ms			
CI_SetMode	27 ms			
CI_Close	105 ms			
CI_Decrypt (4)	47.5 ms			**X**
CI_DeleteKey	27 ms	**X**	**X**	**X**
CI_Encrypt (5)	46.5 ms		**X**	
CI_GenerateIV	34 ms		**X**	
CI_GenerateMEK	27 ms		**X**	
CI_GenerateRa	100 ms		**X**	
CI_GenerateTEK D	360 ms			**X**
CI_GenerateTEK E	350 ms		**X**	
CI_LoadIV	27 ms			**X**
CI_SetKey	27 ms		**X**	**X**
CI_SetPersonality	27 ms	**X**	**X**	**X**
CI_UnwrapKey	27 ms			**X**
CI_WrapKey	33 ms		**X**	
CI_InitializeHash	27 ms			
CI_Hash (6)	34 ms			
CI_GetHash	44 ms			
CI_Sign	131 ms			
CI_VerifySignature	238 ms			

Notes:
(1) Varying times have been measured. Subsequent runs typically take 325-330 ms.
(2) Time during the first run of the program. Subsequent program starts take 4-5 ms.
(3) When already opened, subsequent calls to CI_Open take 0.037 ms.
(4) CBC64 and ECB64 modes. OFB mode requires 0.5 - 1 ms more.
(5) CBC64 mode. ECB64 requires 1 ms more; OFB mode nearly 2 ms more.
(6) No significant difference was measured between in hashing 64 byte and 128 byte blocks.

Appendix D: Fortezza library call timings (SGI Indogo2, SCSI PCMCIA reader)