NPS-CS-09-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

Trusted Emergency Management

by

Timothy Levin, Cynthia Irvine, Terry Benzel, Thuy Nguyen,
Paul Clark, Ganesha Bhaskara

March 2009

Prepared for: the National Science Foundation and the Defense Advanced Research Projects Agency

THIS PAGE LEFT INTENTIONALLY BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Provost

This report was prepared for and funded by: NSF and DARPA.

This report was prepared by:

_____
Timothy E. Levin
Research Associate Professor

Reviewed by:

_____
Peter J. Denning
Chairman
Department of Computer Science

Released by:

_____
Karl Van Bibber
Vice President and
Dean of Research

THIS PAGE LEFT INTENTIONALLY BLANK

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2009	Research; 10/16/06 – 10/16/07

4. TITLE AND SUBTITLE

Trusted Emergency Management

5. FUNDING

Grant numbers: CNS-0430566 and CNS-0430598

6. AUTHOR(S)

Timothy Levin, Cynthia Irvine, Terry Benzel, Thuy Nguyen,
Paul Clark, Ganesha Bhaskara

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Center for Information Systems Security Studies and Research (NPS CISR)
1411 Cunningham Rd., Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-CS-09-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Science Foundation, 4201 Wilson Blvd. 1175 N. ArlingtonVA22230
DARPA, 3701 Fairfax Drive, Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words.)

The ability for emergency first responders to access sensitive information for which they have not been pre-vetted can save lives and property. We describe a trusted emergency management solution for ensuring that sensitive information is protected from unauthorized access, while allowing for extraordinary access to be authorized under the duress of an emergency. Our solution comprises an emergency access control policy, an operational model and a scalable system security architecture. The operational model involves end-users who are on call as first responders, providers of critical information, and a coordinating authority. Extraordinary access to information is allowed to occur only during emergencies, and only in a confined emergency partition, which is unavailable before the emergency and can be completely purged after the emergency. As all information remains within its assigned partition, after the emergency the system can meaningfully enforce its pre-emergency access control policy. A major component of the architecture is the end-user device, and we describe mechanisms on the device for secure storage of data, and for management of emergency state, to indicate feasibility.

14. SUBJECT TERMS

Operating systems: Security and protection: separation kernels, access controls, multilevel security
Organization and design: hierarchical design
Communications Management: network communication
Process Management: multiprocessing/multiprogramming/multitasking
Hardware: Register-transfer-level-implementation: design

15. NUMBER OF PAGES

30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE LEFT INTENTIONALLY BLANK

NAVAL POSTGRADUATE SCHOOL
CENTER FOR INFORMATION SYSTEMS SECURITY STUDIES AND RESEARCH

CENTER OF ACADEMIC EXCELLENCE

SecureCore

Trustworthy Commodity Computation and Communication

Technical Report: NPS-CS-09-001

Trusted Emergency Management

Timothy Levin, Cynthia Irvine, Terry Benzel, Thuy Nguyen, Paul Clark, Ganesha Bhaskara

March, 2009

THIS PAGE LEFT INTENTIONALLY BLANK

This material is based upon work supported by the National Science Foundation under Grants No. CNS-0430566 and CNS-0430598 with support from DARPA ATO. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of DARPA ATO.

Author Affiliations

Timothy Levin, Cynthia Irvine, Thuy Nguyen and Paul Clark:
Naval Postgraduate School
Center for Information Systems Security Studies and Research
Computer Science Department
Naval Postgraduate School
Monterey, California 93943

Terry Benzel and Ganesha Bhaskara
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, Ca 90292

Abstract

The ability for emergency first responders to access sensitive information for which they have not been pre-vetted can save lives and property. We describe a trusted emergency management solution for ensuring that sensitive information is protected from unauthorized access, while allowing for extraordinary access to be authorized under the duress of an emergency. Our solution comprises an emergency access control policy, an operational model and a scalable system security architecture. The operational model involves end-users who are on call as first responders, providers of critical information, and a coordinating authority. Extraordinary access to information is allowed to occur only during emergencies, and only in a confined emergency partition, which is unavailable before the emergency and can be completely purged after the emergency. As all information remains within its assigned partition, after the emergency the system can meaningfully enforce its pre-emergency access control policy. A major component of the architecture is the end-user device, and we describe mechanisms on the device for secure storage of data, and for management of emergency state, to indicate feasibility.

Table of Contents

1 INTRODUCTION 1
2 MODEL FOR EMERGENCY INFORMATION MANAGEMENT 2
3 TRANSIENT TRUST SECURITY POLICY 3
4 SECURITY ARCHITECTURE FOR EMERGENCY INFORMATION MANAGEMENT 4
5 SECURE STORAGE SOLUTION 7
6 EMERGENCY STATE MANAGEMENT SOLUTION 9
7 RELATED WORK 12
8 CONCLUSIONS 13
REFERENCES 14
DISTRIBUTION 17

THIS PAGE LEFT INTENTIONALLY BLANK

Trusted Emergency Management

Timothy Levin*, Cynthia Irvine*, Terry Benzel[+], Thuy Nguyen*,
Paul Clark*, Ganesha Bhaskara[+]

Naval Postgraduate School* and University of Southern California Information Science Institute[+]
{Levin, Irvine, tdnguyen, pcclark}@nps.edu, {tbenzel, bhaskara}@isi.edu

Abstract. The ability for emergency first responders to access sensitive information for which they have not been pre-vetted can save lives and property. We describe a trusted emergency management solution for ensuring that sensitive information is protected from unauthorized access, while allowing for extraordinary access to be authorized under the duress of an emergency. Our solution comprises an emergency access control policy, an operational model and a scalable system security architecture. The operational model involves end-users who are on call as first responders, providers of critical information, and a coordinating authority. Extraordinary access to information is allowed to occur only during emergencies, and only in a confined emergency partition, which is unavailable before the emergency and can be completely purged after the emergency. As all information remains within its assigned partition, after the emergency the system can meaningfully enforce its pre-emergency access control policy. A major component of the architecture is the end-user device, and we describe mechanisms on the device for secure storage of data, and for management of emergency state, to indicate feasibility.

1 Introduction

During crises, *first-responders* can often save lives and preserve property better if they have access to certain sensitive or restricted information – such as the addresses of the elderly or handicapped, or the blueprints for transportation control systems and private buildings. However, the large population of potential first responders makes it infeasible to pre-screen them all, e.g. via national security clearances. On the other hand, if sensitive information is not protected adequately, the resulting damage could be comparable in scope to the lives and property otherwise saved. Currently, even temporary access to such sensitive information may not be available to first responders, because of privacy, liability, and other concerns, although the need for dynamic security policies and systems, which can adapt to changing circumstances, is recognized [21].

To support effective emergency response, participating organizations must have confidence that the information they make available is protected from unauthorized access and propogation. In this paper, we provide a new policy and operational model for the management of emergency information. We describe a technical foundation for the automated realization of the emergency information model in a modern IT environment. At the heart of this solution is a security architecture that utilizes the latest state-of-the-art hardware security concepts as well as the newest generation of secure *separation kernel* technology. Together, these results provide an environment for emergency information management and protection that is:

- Demonstrably assurable to enforce its security policies,
- Persistent during "emergency" as well as non-emergency periods, and
- Global across all participating organizations

Our approach provides verifiable techniques for both enabling temporary access to sensitive information needed during catastrophes and emergencies, and strictly prohibiting the propagation of that information – and ultimately rescinding it after the emergency. We also describe innovative techniques for effective management of emergency *state* in a distributed environment and for efficient use of new processor-internal encryption mechanisms for the protection of stored data.

To be practical, i.e., easy to use, commercially feasible, as well as effective, a solution to this problem must be usable "in the field" and provide dual-use functions on familiar equipment. This ensures that first responders are adept in the use of the device and that it is likely to be at hand if and when an emergency arises. The target platform for research and validation of our approach is a handheld computer, the *E-device*. Our security solutions in this form factor provide a mobile emergency-response capability that enables rapid, knowledgeable, emergency response, promotes usability, and ensures Third Party information protection during emergencies. The result is a trusted foundation for more effective crisis management activities.

The remainder of the document is structured as follows. Section 2 describes the conceptual model including the primary entities involved, the elements of trust between them, and the system security policy that governs access to information. Section 3 describes the security architecture to support this policy and model. Sections 4 and 5 describe engineering details of our innovative secure data storage and emergency state management mechanisms. Related work and conclusions follow in Sections 6 and 7.

2 Model for Emergency Information Management

In the emergency-response milieu we consider the following roles, each of which has a direct stake in effectiveness of the E-device:

- *First responders* – members of our primary health and safety organizations (e.g., medical, police and fire), as well as transportation, communication, construction, maintenance and other workers who may be called on at the scene of a disaster.
- *The Authority* – an organization that coordinates emergency response in a given context, such as the Department of Homeland Security, or other disaster-relief coordinating organization. Within a large enterprise, the Authority could be a dedicated department.
- *Third Party* data providers that supply emergency information. As a simplifying assumption for our initial work, we consider Third Parties to be mutually trusting – forming a single class of trust – and are represented simply as a single Third Party.

We refer to information that is designated to be available to emergency first responders, which they may not have been vetted or cleared to see, as "emergency information." Emergency information is "owned" by third parties, who may not wish it to be generally distributed or shared with other third parties. In many cases, there are financial, regulatory and political hurdles to sharing information, as well.

Some examples of emergency information that can enable effective emergency response relate to:

- Physical security– building plans, transportation logistics, critical control mechanisms
- Privacy of customers and employees – medical, communication, and employment records
- Nationally classified information – "continuity of government plans" and information about chemical/biological warfare supplies and antidotes.
- Trade secrets – the inner workings of pharmaceutical and electronic products

First responders are provided with E-devices, which provide both day-to-day and emergency-response information and communications needs. The E-Device is initialized with certain emergency information, and the Authority can transmit additional emergency information to the E-device in the field. The collection of E-devices, along with the trusted systems with which the Authority and the Third Party communicate with E-devices, along with the communication infrastructure, comprise the distributed emergency information management system, or *Emergency Network*, for short.

For simplicity in the initial model, we characterize the *emergency state* of the Emergency Network as either "on" or "off." The Authority manages the emergency state, and communicates state changes to E-devices and Third Parties.

Next, we look at the policy and trust model for emergency information management, followed by the business and operational arrangements that support a secure configuration.

3 Transient Trust Security Policy

The objective is to enable first responders to access emergency information in a *secure* manner. We assume the existence of a *strict policy* regarding the authorized accesses of users to data objects. We define an *emergency policy* that allows additional, "extraordinary accesses" by end users to emergency information, which may occur only during an emergency.[1] While these accesses are not violations of the security policy, they are "beyond the pale" of usual MAC and DAC controls [15], and are thus additional constraints are appropriate. Together the strict policy and the emergency policy can be thought of as the emergency network's overall security policy.

The temporal constraint on extraordinary accesses reduces the window of opportunity resulting from adverse security events that are outside of the control of the trusted computing base, such as can result from the inadvertent disclosure of a password, or the behavior of a malicious insider.

Consistency of local and global emergency state is important, although the natural variability of mobile device connectivity limits the degree of consistency achievable. The emergency network stakeholders must agree on a revocation policy for when an E-device loses connectivity with the Authority during an emergency For example, it may be allowable for the user to continue to have access to the device's local emergency information, as limited by a timeout value that provides an upper bound on delayed revocation [9][19], until connectivity is reestablished and the E-device can re-synchronize with the global emergency state.

3.1 Trust Model

The Emergency Network trust model – the explicit behavioral dependencies between stakeholders, and supporting cryptographic key exchanges – is subject to many variables. We present the following arrangement sand procedures as a framework for discussion and analysis of our approach, but many other arrangements are possible within its solution space.

The Authority establishes an agreement (e.g., MOU or SLA) with the Third Parties regarding their various expected behaviors, such as the precise information sharing policies and level of protection that will be afforded to shared information. The E-device developer must design and implement it to meet all of the security requirements and policies agreed to by the stakeholders. Depending on the assurance requirements specified in those agreements, the E-device may be subject to an independent evaluation, such as defined by the Common Criteria [3], to validate its security properties.

The Authority and the Third Party may agree for the E-device to host one or more Third Party "trusted" applications in the *Trusted Partition* (discussed below), which provides a protected processing environment for these applications. The specific security and functional characteristics of the trusted applications that the Third Party depends on are outside of the scope of this paper, although in any event, enforcement of transient trust security policy would not depend on Third Party applications.

[1] *Transient trust* is the concept of securely bestowing temporary, additional privileges to users so that they can more effectively achieve results desired by information stakeholders.

The Third Parties also rely on the Authority to declare the start and end of the emergency in an appropriate and secure manner; to correctly configure the E-devices at a device-configuration *Depot*; and to subsequently protect any secret cryptographic keys entrusted to it.

The Authority and individual Third Parties exchange symmetric-cryptography keying material of the quality agreed upon. Additionally, our approach supports the following operations and features:

- The Authority initializes the E-devices, including installation of cryptographic keys, Third Party applications and pre-distributed Third Party emergency data.
- The Authority shares a separate secret key with each device, which is installed into the hardware at the Depot. Third Parties may not access this key.
- The Authority shares a separate secret with each Third Party, which may not be shared with any of the E-devices.
- Third Parties may have a secret key installed on certain E-devices to support secure communication with those devices.

The Authority is a proxy for the Third Parties in terms of enforcement the security policy. No attempt is made to protect Third Party communications, data or code from access or modification by the Authority, given that the Authority installs all cryptographic keys and trusted software.

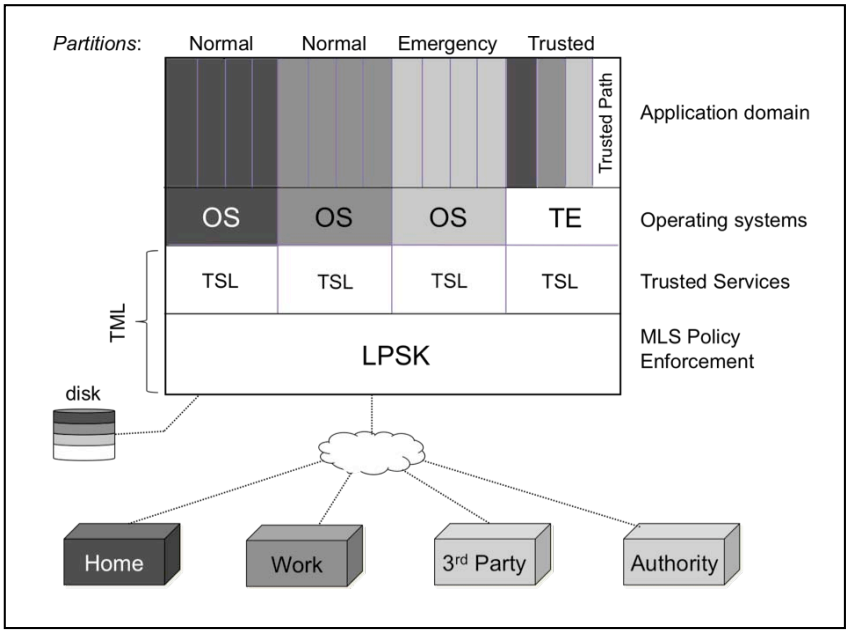

Figure 1. SecureCore Security Architecture

4 Security Architecture for Emergency Information Management

The technological foundation of our solution is the SecureCore security architecture [11][6][13] shown in Figure 1. We also postulate the availability of certain currently proposed hardware security primitives, as discussed below.

The nexus of security enforcement on the E-device, the Trusted Management Layer (TML), divides all information and programs on the E-device into discrete *partitions*. Users can only

interact with one partition at a time. The TML creates and manages the *Emergency Partition* as a protected staging area where emergency information is confined, and where users are provided with transient access to the information. The Emergency partition may be configured to permit access to non-emergency data as well. Emergency data is only stored in the Emergency Partition, and the TML ensures that information does not leave the Emergency Partition (except for writing to the Third Party via a trusted network channel). The TML ensures that only authorized users may access the Emergency (or any other) Partition. The net result is that the only way a user can ever access emergency information is through an Emergency Partition during an emergency.

The TML comprises the Least Privilege Separation Kernel [16][12] and the Trusted Services Layer (TSL) layer [6]. The LPSK creates *exported resources* from the platform's physical resources, partitions the exported resources and controls interactions between the partitions. The TSL layer virtualizes certain LPSK resources for the use of partition applications (e.g., commercial OSs), and associates security labels with the kernel's exported resources.

The *Trusted Partition* contains the *Trusted Executive* (TE), which provides process services for high integrity applications, such as secure sealing of documents and secure session management, as well as the *Trusted Path Application* (TPA), which is the gatekeeper for user access to partitions, as directed by the TML.

The *Normal Partitions*, i.e., those other than Trusted and Emergency Partitions, and *Emergency Partitions* are intended to host a commercial OS and typical office applications, providing familiar and functionally rich user interfaces.

The strict security policy is enforced on the E-device through the TML's lattice-based security policy, which provides the desired qualities of global and persistent policy enforcement, including the usual MLS restrictions on access to data. For support of transient trust, the emergency policy allows extraordinary accesses, within predefined and well-understood limits with respect to the lattice. The details for the use of MLS labels in this manner are currently being investigated.

On the E-device, the TML communicates with the Authority to manage local awareness of emergency state. Once a particular emergency ends, the TML renders the Emergency Partition inaccessible to the E-device user. The TML can be configured so that after the emergency a snapshot of emergency data is either copied to a static emergency storage partition (e.g., for audit of generated or modified data), or updated data is transmitted to the Third Party. A daemon process in the Emergency Partition manages the reception of emergency data in the field, e.g., for data updates as well as to "rotate" information to the device when the device's storage capacity is insufficient – otherwise the Emergency Partition is quiescent during the emergency-off state.

4.1 Hardware Cryptographic Primitives

We assume the availability of several hardware cryptographic primitives like those postulated for the *Secret Protected* (SP) processor [14] [7]. While we characterize these features as part of the hardware instruction set, some may be suitably instantiated through an off-chip device [20].

The processor differentiates a *crypto processing* (or "concealed execution") mode, wherein access to the cryptographic primitives is provided to software. This mode is entered and exited through the execution of special instructions.

The set of contiguous instructions to be executed while in crypto processing mode is called a *Trusted Software Module* (TSM). Before installation, a TSM is hashed with a processor-resident, device-specific "root" key (DRK). When a TSM is loaded into memory for execution, the instructions are validated relative to the previously generated hash values – providing integrity of the program instructions at runtime.

Several non-volatile processor registers are available only when in the crypto processing mode: one (DRK) that can be written only once per boot; and two general storage registers (called SRH and CEM_buffer).

Three crypto-transform functions are provided: sp_derive, which hashes two words with the DRK; secure_store, which *marks* a cache line for processing (i.e., hashing and encryption with the DRK) upon eviction from the processor cache (thus, the transform occurs asynchronously to the execution of the secure_store instruction); and secure_load, which decrypts memory as it is loaded into the processor cache and validates its hash.

4.2 Crypto-Hardware Protection of TML

We introduce another processor mode, called *code integrity check* (CIC) mode, for protection of the privileged "supervisor" program. At compile time, the TML code is hashed with the DRK. When such code is executed, the inline hash values are validated, and execution is halted if the validation fails. This ensures that the TML code can only be executed on the intended device, and any changes to TML code are detected. While the *crypto-processing* mode is single-threaded, CIC mode supports multithreading.

4.3 Subjects and Objects

There are zero or more processes per partition. A multi-program process can be structured to utilize the several rings (i.e., hardware privilege levels) available outside of the TML, e.g., one program per ring. Thus, a "subject" is a process-ring pair, and the use of multiple programs within a process can provide efficient and secure interaction between subjects.

The TML exports several object abstractions: memory segments, and synchronization primitives; and two object group structures: disk volumes, and segment volumes.

4.4 Disk Volumes

A disk volume is a contiguous set of physical disk sectors that is exported to a particular partition by the TML as a (virtual) disk device. The guest OS of one partition can mount the disk volume of another partition, as allowed by the TML security policy, which enables sharing of persistent data between the partitions. However, if the guest OS creates its own object abstractions within a volume, the TML cannot help enforce OS restrictions (e.g., for applying the principle of least privilege) regarding the separation of those "objects."

Table 1. Authentication and communication security

Party 1	Party 2	Key Basis
Authority	E-device Trusted Component	Device Root Key (DRK)
Authority	Third Party	Keys agreed to in SLA/MOU
Third Party	E-device Normal Partition	Keys provided by Third Party and installed at Depot

4.5 Segments and Segment Volumes

Segments are individual memory objects exported and protected by LPSK (as opposed to OS-created objects inside of a disk volume). User-accessible segments are organized in exported segment volumes managed by the LPSK. Similarly, the TML maintains one or more internal segment volumes for its own use.

The segment volume structures (i.e., all of the segments and related metadata that comprise a segment volume) are sealed and encrypted on disk. A general description of how the TML secures segment volumes is provided in Section 5.

4.6 Trusted Channels and Trusted Paths

Trusted channels are cryptographically secured communication paths between two trusted components (e.g., the TML and the trusted computer of the Authority or Third Party). The TML can also export a given trusted channel to a specific partition's application domain, via a logical I/O device, which provides connectivity between the applications in that partition and the trusted component at the other end of the trusted channel.

A *trusted path* is a secured communication path between a *user* and a trusted component. If the Authority utilizes trusted software to access a trusted channel between his computer and the TML, a *remote trusted path* to the TML results.

Trusted channel keys and related sensitive channel parameters are stored in a TML database and are not available to applications; other parameters are made available to applications as necessary (e.g., the device handle). The TML encrypts the database with a key derived from the DRK. The TML can recreate this key using its TSM functions, so the key does not need to be stored in persistent memory. The E-device can be configured so that security levels are bound to certain remote IP addresses. When network data is sent or received by a partition, the E-device must ensure that the remote-address level matches the partition level and matches or is within the level range of the physical device.

Table 1 shows the cryptographic keys used as the basis for creating Trusted channels. For example, a trusted component of the E-device may establish a trusted channel with the Authority. The channel security is based on a session key generated with the sp_derive instruction from a non-generated key will be secret.

5 Secure Storage Solution

5.1 Sealing procedures

Segment volumes can be sealed to protect their integrity at various intervals or in response to different events such as a request to shut down the system. For this description, we assumed that there is one exported and one internal volume (called evol and ivol), but extension to support multiple segment volumes should be straightforward.

secret seed that is shared with the Authority; since the DRK utilized by sp_derive is secret, the

The TML calls an internal TSM with volume check-sum values it has calculated, which moves each check-sum to a register and then calls sp_derive to generate and store the seal (shown as V-hash in Figure 2) to the SRH register.

5.2 Storage procedures

Efficient encrypted disk storage functions can be created from transient memory encryption functions (e.g., sp_secure_store), by pushing encrypted data out of cache and then using DMA to

move it onto the disk[2] (see details in Section 5.3. Alternatively, if it is desired to use programmed I/O to write to disk, data marked with secure_store can be loaded in a normal manner (i.e., by not using secure_load) which will cause it to arrive in the processor register encrypted.

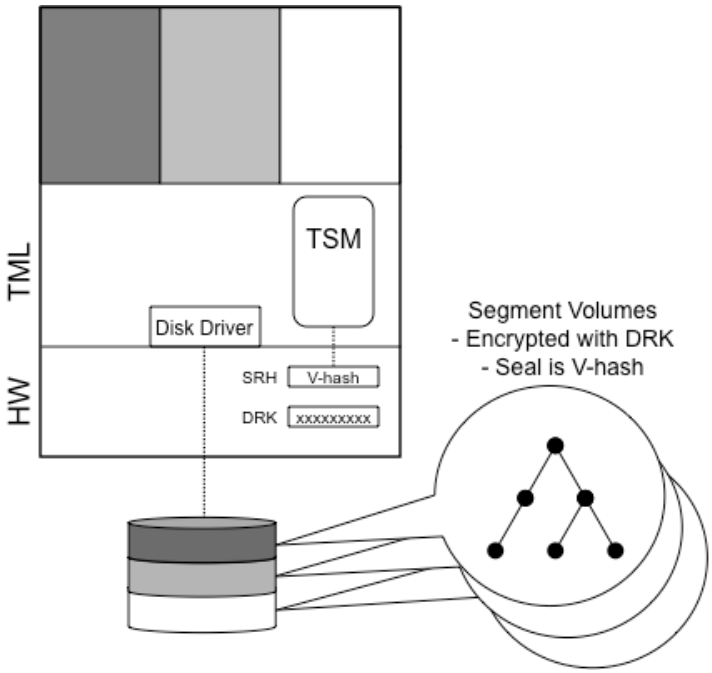

Figure 2. Segment volume protection

When the data is reloaded it will be decrypted with the DRK. We assume that the integrity validation data from the encryption phase can be accessed to validate the loaded, decrypted data, or that the integrity check can otherwise be bypassed or ignored (since we separately cryptographically sign the volumes, the hardware integrity check is not needed).

Encryption of user data (exported volumes) can be handled in different ways. For example, the tml_disk_flush function defined below can be exported to segment-aware applications, or the TML could coerce any calls to write to the disk to use this logic.

Another possibility is to configure sp_secure_store to use a value in one of the generally-accessible crypto registers, in which case, the user's stored, encrypted data could be portable to another of his or her E-devices (or to another computer that is enabled with similar hardware cryptographic functions) in which the user has installed the encryption key.

5.3 Hardware Encryption Details

To encrypt a segment on disk, applications (or modules within TML) call tml_secure_flush with the handle of the segment. tml_disk_flush first ensures that all of the "cache lines" of the segment are marked for encryption by calling _tml_mark_secure, and then calls lpsk_flush2disk to write the encrypted segment to disk.

[2] This assumes that the encryption function is not vulnerable to known plaintext attacks, which would expose the DRK.

```
tml_disk_flush(user_seg: evol_seg)=
   for i = 1 .. Length(user_seg) do
      if not user_seg.i.SecureData then
         _tml_mark_secure(user_seg.i, user_seg, i)
   lpsk_flush2disk(user_seg)
```

_tml_mark_secure is handed a word of data and a memory destination (segment and offset). It moves the word into a general-purpose register and then calls _TSM_secure_store.

```
_tml_mark_secure(w: word, dest_seg: ivol_seg, offset: word) =
   hw_move(R2, w)
   _TSM_secure_store(dest_seg, offset, R2))
```

The internal procedure _TSM_secure_store calls sp_secure_store to mark the cache line for encryption:

```
_TSM_secure_store(dest_seg, offset, R2)=(
   sp_secure_store(dest_seg, offset, R2))
```

lpsk_flush2disk pushes to main memory the elements of the segment that are in the processor cache, which causes SP to encrypt them, and then writes the entire segment to the disk. If the operation is called from within an TSM, a DMA disk device must be used to copy the encrypted segment directly from memory onto the disk, because using the processor to write to disk would cause the memory to be decrypted first - i.e., by pulling it back into the processor in order to write to the disk. If not in a TSM, then programmed I/O or DMA I/O can be used to write to the disk.

```
lpsk_flush2disk(user_seg) =
   for i = 1 .. Length(user_seg) do  //flush seg cache lines
         x86_clflush(i)
         _dma_device_write(user_seg)
```

6 Emergency State Management Solution

Emergency state management is a security critical operation as the access to emergency information is dependent on the state of emergency as seen by the E-device. Thus, emergency state management must be assured to a degree commensurate with the required level of trust of the Emergency Network.

Recall that the Authority maintains the global emergency state of the network and communicates state changes to E-devices. Trusted components of the E-device (such as the Emergency Manager, “e-manager,” and TPA in Figure 3) receive, store, and respond to emergency status updates. Although these functions can be implemented in software, hardware can provide additional assurance (i.e., assuming a validated implementation, the non-malleability of hardware logic supports confidence in its continued correctness) with the addition of key “state management” primitives.

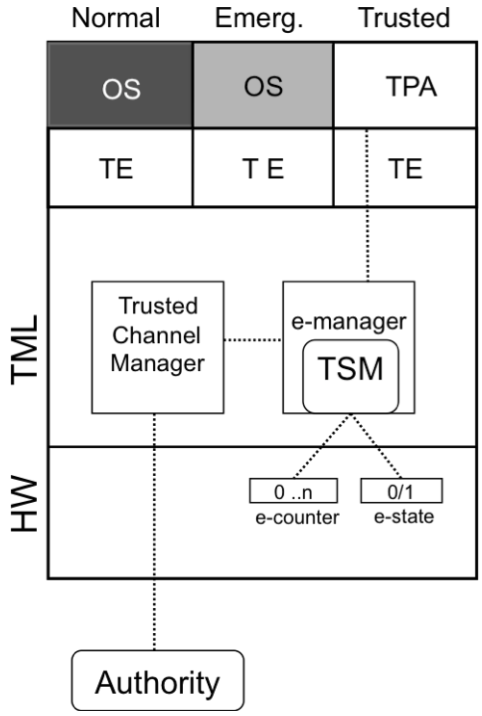

Figure 3. Local Emergency state management

The processor is extended with several state-management features: Two new instructions: hw_update_state and hw_get_state; a local state counter: e_counter; and a state bit: e_state.

6.1 Emergency State Transition Protocol

The Authority keeps a record of the E-device_id and DRK for each E-device. It also maintains the emergency state-change number.

To announce a change to the emergency state, the Authority updates its own information (in this case, the start of an emergency):

```
emergency_counter = emergency_counter + 1
emergency_state = 1  // emergency is “on”
```

… and then generates an emergency message for one or more E-devices, and sends them each over a trusted channel (see Section 6.4). The message contains two parts, a payload and the hash of the payload.

The payload contains the new emergency state and the global state-change number; it is encrypted with each E-device’s DRK (in this case, reflecting device number n):

```
payload = _authority_encrypt(e-state, e-counter, DRKn)
```

The crypto-hash of the encrypted payload is also based on the target E-device’s DRK:

```
hash = _authority_keyed_hash(payload, DRKn)
```

The message is the concatenation of the payload and the hash:

```
message = _authority_concatenate(payload, hash)
```

As shown in Figure 3 a TSL trusted channel manager on the E-device receives the message, and sends it to the e-manager. The e-manager calls its TSM to update the emergency state.

If hw_update_state was successful, the e-manager sends an acknowledgement to the Authority and calls its TSM to get the value of e-state.

The e-manager takes appropriate action depending on whether an emergency has started or stopped. In the former case, it will signal to the TPA to announce an emergency to the user and make the Emergency Partition available; or it will terminate an existing emergency.

6.2 State Management Hardware Support Details

Hardware processes hw_update_state with the following actions (see Table 2), which from the interface to the hardware appear to be atomic (internal logical functions that are not available at the hardware interface are prefaced with a tilde "~"). Before processing, hardware ensures that the calling program has sufficient privilege.

```
hw_update_state(message) =
   tmp := ~sp_check_hash (DRK, message.payload, message.hash)
   if tmp = "success" then
         tmp := ~sp_decrypt(DRK, message.payload, tmp_values)
         if tmp = "success" then
               if tmp_values.counter > e_counter then
                     e_counter := tmp_values.counter
                     e-state := tmp_values.state
               else tmp = "bad_ecounter"
   return (tmp)  // "success" or a specific error
```

hw_update_state validates the hash against the payload using the E-device's DRK, and then decrypts the payload with the DRK. It also checks that the new counter value is greater than the previous value to ensure that hardware has not been requested to perform duplicate updates. If successful, hardware then writes the decrypted payload state to the hardware e-state register.

Table 2. State Management Instructions

Instruction	Arguments	Exceptions
hw_update_state	subject: implicit message: string	1. Subject lacks sufficient privilege 2. Message has wrong format 3. The message counter value is too low 4. The message hash does not match
hw_get_state	subject: implicit GR1: word	
The calling subject is an implicit parameter, which indicates the security attributes of the program or module that invokes the hardware instruction.		

6.3 Emergency State Protocol Security Analysis

The correct management of emergencies depends on the correct interpretation of the global emergency state by E-devices. Threats to this correctness include corrupted, impersonated or disrupted communication of the emergency message, and corrupted, impersonated or disrupted processing or storage of the emergency state on the local E-device. Additionally, in situations where confidentiality of emergency state changes is required, eavesdropping of emergency

communication, storage, and processing is a concern. These concerns are addressed in the sections below.

6.3.1 Communications integrity

It does not appear that the uniqueness of the emergency message can be ensured in a simple one-way communication from the Authority to the hardware, as it would be difficult for the E-device to distinguish previous authentic messages that it had missed from authentic current messages.

In contrast, the use of the trusted channel ensures that the emergency message is from a trusted source and is current. The emergency hash and counter provide additional defense-in-depth that the message is authentic, current, and unmodified during communication.

6.3.2 Communications availability

Availability is problematic, as we cannot guarantee connectivity of the E-devices. However, we expect the E-devices to use best effort techniques to stay synchronized with the Authority.

An E-device may be offline occasionally, for example during extreme emergency conditions, but the proposed emergency policy allows the user to continue to have access to emergency data if the E-device loses connectivity during the emergency-on state. During the offline period, it is conceivable the emergency could even have cycled on and off one or more times.

The individual E-device does not need to be concerned with offline state changes; only that it should re-synchronize its emergency state (i.e., request an emergency state packet from the Authority) when connectivity is restored, and do so before any further user activity is allowed.

6.3.3 Storage and processing integrity

On the E-device, emergency state is managed exclusively by trusted components. State changes are validated and stored in hardware. Access to these state elements is restricted to TSMs. Hardware support enhances protection against attacks on the storage and processing of emergency state and reduces the amount of software that must be trusted for emergency state management.

One potential threat is that corrupted trusted software (while the security architecture provides high assurance self-protection, this analysis is provided to support defense-in-depth) on the E-device could seal messages with the DRK by using sp_derive, and then spoof the E-device into believing the wrong emergency status. However, protection against these counterfeit emergency messages is accomplished by the use of a different emergency-signing algorithm than that used by sp_derive (e.g., the seals could be a different length).

6.3.4 Emergency confidentiality

Confidentiality is ensured through several means: the use of a trusted channel for communications, the use of trusted components for management of emergency state, and hardware support for hiding the content of emergency messages during processing and storage. (This even hides the value of upcoming state changes from the trusted components until after the hardware state change has occurred).

7 Related Work

Our work utilizes currently-proposed concepts for CPU-based cryptographic support. Although mechanisms for cryptographic support using coprocessors are available, e.g. the IBM 4758 co-processor [18] and the TCG Trusted Platform Module [20], these schemes do not provide the CPU-level of protection for system software and data. Thus they are more vulnerable to internal attack by elements within the platform architecture.

In 2006, IBM announced "Secure Blue", an architecture that features encryption built into the microprocessor intended to protect the integrity the system as well as protect data on chip and in transit to remote systems [10]. To date, little additional information is available, however, the announcement did not discuss the trustworthiness of the architecture, nor did it address the problem of separating information associated with events or based upon mandatory policies.

Fu and Xu describe an access control model for mobile computing environments that includes spatial and temporal factors, however this work does not address the concerns of mandatory policy enforcement or the practical implementation of such a system [8]. A model for temporal access control in the context of databases has been proposed [2] and an authorization model for access to data during restricted time intervals was presented by Afinidad [1]. Neither addresses the problem of event-based access that would be required in an emergency. Similarly, implementations of temporal access controls, e.g. [5], are for predefined time intervals rather than aperiodic events.

Ross Anderson proposed a model [**Error! Reference source not found.**] for the security and privacy of patients' medical information, which accommodates "emergency" access to information, e.g., for extraordinary medical or legal events. In this model, restrictions on access to patient records are controlled via access control lists. No changes to the lists are made without previous user consent. A record is kept of all accesses to records and changes to access control lists, and the user is notified of any emergency over-ride of the rule of consent or of the access list mechanism. Our approach differs from this model in several ways, most notably in the abilities to control when emergency overrides may occur, to control the extent of emergency override, and the capability to revoke permissions to information in real time.

Yang and Shin [22] describe techniques to use a page-based access control mechanism within a hypervisor. This approach differs from ours in that it does not address emergency access; it relies on an untrusted hypervisor; and it potentially requires almost twice the amount of system memory.

The OASIS EDXL provides a standard for the format of information exchange during emergencies [17]. The specification states that the confidentiality level of each payload should be provided within each content object. In addition, payloads may be encrypted and multiple encrypted payloads may appear in the same message. The architecture presented here would provide a trusted context for the management of EDXL data.

8 Conclusions

In this paper, we addressed the primary information assurance problem faced in emergency management: the secure dissemination and control of sensitive organizational information during a crisis. Information sharing during an emergency is distinguished by the need to allow extraordinary access to information for relatively short periods, and the need for information sharing by separate organizations with varying security requirements and varying amounts of mutual trust.

Our approach provides a business and operational trust model that addresses the inherent policy concerns of a multi-organizational emergency network. We also presented the concept of *transient trust*, and described how it is secured through both spatial and temporal confinement of emergency information.

An integrated technical solution for management of emergency information is encompassed in our security architecture, which is based on the Least Privileged Separation Kernel, and utilizes next generation hardware-based cryptographic primitives. We also describe two point solutions for emergency management, both of which leverage new hardware security primitives within the context of our security architecture: using transient-memory encryption for secure data storage, and a coherent system for distributed emergency state management.

References

1. F. Afinidad, F., Levin, T. E. Irvine, C. E. and Nguyen, T. D. "A model for temporal interval authorizations," Hawaii International Conference on System Sciences, Software Technology Track, Information Security Education and Foundational Research, (Kauai, Hawaii), p. 218, January 2006.
2. Bertino, E. Bettini, C. Ferrari, E. and Samarati, P.. (1996) A Temporal Access Control Mechanism for Database Systems, IEEE Trans. on Knowledge and Data Engineering, Vol. 8, Nr. 1, pp. 67-80.
3. C. C. M. Board, *Common Criteria for Information Technology Security Evaluation*, Common Criteria Maintenance Board, 3.1 revision 1, September 2006.
4. Boneh, D. and Franklin, M. "Identity based encryption from the Weil pairing." *SIAM J. of Computing*, Vol. 32, No. 3, pp. 586-615, 2003.
5. Chiang, K., Nguyen, T. D. and Irvine, C. E. "A Linux implementation of temporal access controls," Proc. 8th IEEE Systems, Man, and Cybernetics Information Assurance Workshop, (West Point, NY), pp. 309–316, 2007.
6. P. C. Clark, C. E. Irvine, T. E. Levin, T. D. Nguyen, and T. M. Vidas, *SecureCore software architecture: Trusted path application (TPA) requirements*, Tech. Rep. NPS-CS-07-001, Naval Postgraduate School, Monterey, CA, December 2007
7. Dwoskin, J. and Lee, R. "Hardware-rooted Trust for Secure Key Management and Transient Trust." *CCS'07*, October 29–November 2, 2007, Alexandria, Virginia, USA.
8. Fu, S. and Xu, C.-Z. "A coordinated spatio-temporal access control model for mobile computing in coalition environments," in Proc. IEEE International Conference on Parallel and Distributed Processing Symposium (IPDPS'05), 8 pp., April 2005.
9. Grossman, G., "Immediacy in Distributed Trusted Systems." In *Proc. of Annual Computer Security Applications Conference*, New Orleans, Louisiana, December 11-15, 1995, IEEE, Computer Society Press.
10. IBM, IBM Extends Enhanced Data Security to Consumer Electronics Products. April 2006. http://www.cio.com/article/20075/IBM_to_Offer_Chip_Based_Encryption_for_ PCs_PDAs
11. Irvine, C. *Collaborative Research: SecureCore for Trustworthy Commodity Computing and Communications*. www.fastlane.nsf.gov, Award 0430566. 31 Mar. 2005.
12. Levin, T., Irvine, C., Weissman, C., Nguyen, T., "Analysis of Three Multilevel Security Architectures," Proc. of *Computer Security Architecture Workshop*, ACM. November 2, 2007, Fairfax, Virginia, USA.
13. T. Levin, G. Bhaskara, T. D. Nguyen, P. C. Clark, T. V. Benzel, and C. E. Irvine, *SecureCore security architecture: Authority mode and emergency management*, Tech. Rep. NPS-CS-07-012 and ISI-TR-647, Naval Postgraduate School and USC Information Science Institute, Monterey, CA, October 2007.
14. Levin, T., *SP Summary (with Authority Mode)*, NPS Technical Report NPS-CS-08-007, September 2007.
15. C. J. McCollum, J. R. Messing, and L. Notargiacomo, "Beyond the pale of mac and dac – defining new forms of access control," in *Symposium on Security and Privacy*, pp. 190 – 200, IEEE Computer Society, 1990.
16. National Security Agency. U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03, 29 June 2007.
17. OASIS, Emergency Data Exchange Language (EDXL) Distribution Element v1.0, May 2006. http://docs.oasis-open.org/emergency/edxl-de/v1.0/EDXL-DE_Spec_v1.0.pdf
18. Smith, S. and Weingart, S. "Building a high-performance, programmable secure coprocessor," Computer Networks, vol. 31, pp. 831–860, November 1999.

19. S. G. Stubblebine, “Recent-secure authentication: Enforcing revocation in distributed systems,” in Symposium on Security and Privacy, (Oakland, CA), IEEE Computer Society, 1995.
20. Trusted Computing Group. *TCG specification architecture overview*. Trusted computing group Rev 1.2, April 2004.
21. Wolfowitz, P. *Global Information Grid (GIG) Overarching Policy*. U.S. Department of Defense, directive number 8100.1, September 19 2002.
22. Yang J. and Shin, K., “Using hypervisor to provide data secrecy for user applications on a per-page basis,” in VEE ’08: *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 71–80, Seattle, WA: ACM, 2008.

THIS PAGE LEFT INTENTIONALLY BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 1
   8725 John J. Kingman Rd
   Ft. Belvoir, VA 22060

2. Dudley Knox Library — 1
   Naval Postgraduate School
   Monterey, CA 93943

3. Karl Levitt — 1
   National Science Foundation
   4201 Wilson Blvd.
   Arlington, VA 22230

4. Lee Badger — 1
   DARPA / IPTO
   3701 Fairfax Drive
   Arlington, VA 22203

5. Timothy E. Levin — 1
   Naval Postgraduate School
   Monterey, CA 93943

6. Ganesha Bhaskara — 1
   4676 Admiralty Way, Suite 1001
   Marina del Rey, CA 90292

7. Thuy D. Nguyen — 1
   Naval Postgraduate School
   Monterey, CA 93943

8. Paul C. Clark — 1
   Naval Postgraduate School
   Monterey, CA 93943

9. Terry V. Benzel — 1
   4676 Admiralty Way, Suite 1001
   Marina del Rey, CA 90292

10. Cynthia E. Irvine — 1
    Naval Postgraduate School
    Monterey, CA 93943

THIS PAGE LEFT INTENTIONALLY BLANK