Securing the Dissemination of Emergency Response Data with an Integrated Hardware-Software Architecture

Timothy E. Levin[1], Jeffrey S. Dwoskin[2], Ganesha Bhaskara[3], Thuy D. Nguyen[1], Paul C. Clark[1], Ruby B. Lee[2], Cynthia E. Irvine[1], and Terry V. Benzel[3]

[1] Naval Postgraduate School, Monterey, CA 93943, USA
{levin,tdnguyen,pcclark,irvine}@nps.edu
[2] Princeton University, Princeton, NJ 08544, USA
{jdwoskin,rblee}@princeton.edu
[3] USC Information Sciences Institute, Marina Del Rey, CA 90292
{bhaskara,tbenzel}@isi.edu

Abstract. During many crises, access to sensitive emergency-support information is required to save lives and property. For example, for effective evacuations first responders need the names and addresses of non-ambulatory residents. Yet, currently, access to such information may not be possible because government policy makers and third-party data providers lack confidence that today's IT systems will protect their data. Our approach to the management of emergency information provides first responders with temporary, transient access to sensitive information, and ensures that the information is revoked after the emergency. The following contributions are presented: a systematic analysis of the basic forms of trusted communication supported by the architecture; a comprehensive method for secure, distributed emergency state management; a method to allow a userspace application to securely display data; a multifaceted system analysis of the confinement of emergency information and the secure and complete revocation of access to that information at the closure of an emergency.

Keywords: Information Assurance, Computer Security, Policy Enforcement, Secret Protection (SP), Transient Trust, Emergency Response.

1 Introduction

In a crisis, first-responders can often save lives and limit damage if they have access to certain sensitive or restricted information. Examples of such emergency information are: building floor plans, schematics for infrastructure or transit systems in a city, or medical records of victims. However, government policy makers and third-party data providers lack confidence in the ability of emergency IT systems to protect their data relative to confidentiality, sensitivity, privacy, liability, and other concerns, and are unwilling to release such data, even on a temporary basis, to civilian first responders[1].

L. Chen, C.J. Mitchell, and A. Martin (Eds.): Trust 2009, LNCS 5471, pp. 133–152, 2009.

In this paper, we address the essential confidentiality, integrity, access control, revocation and data containment capabilities needed in future emergency response systems. We show how a platform based on commercial off-the-shelf technology can be used for the secure management of emergency information and we detail how it can (1) communicate securely with a trusted authority, first responders, and information providers; (2) manage distributed emergency state with high integrity and assurance; and (3) enable emergency access to sensitive information while strictly maintaining its confinement as well as revoking access with high assurance.

In the sections that follow, we show how a handheld Emergency Device, the *E-Device*, utilizing the latest generation separation kernel technology [2,3] and state-of-the-art hardware security concepts [4], can be a trusted foundation for secure crisis response.

2 Secure Platform Architecture

The secure platform architecture of the E-Device (see Fig. 1) is based on a commercial general-purpose processor (nominally an x86) enhanced with the Authority-Mode Secret Protection (SP) architecture features [4], a Trusted Management Layer (TML) — comprising a least privileged separation kernel [2], and a security services layer that virtualizes certain separation kernel resources — and trusted software programs that run in one of the TML-provided partitions. These hardware/software components, with the addition of a Trusted Software Module (TSM) application supported by SP, form the Trusted Computing Base (TCB) for our E-Device.

2.1 Secure Software Stack

The trusted software for the E-Device utilizes Intel x86 privilege levels to protect the resources in each level from the subjects in less-privileged domains. The two most privileged layers are collectively referred to as the TML, while the next most privileged layers house a trusted executive and the Trusted Path Application (TPA).

The TML provides software-based security enforcement, and is supported by security features of the underlying Intel x86 and SP hardware. While support for many different client OSes is not the focus of this research, the TML is designed to provide to the commodity OS a standard execution environment with respect to platform hardware features.

The TML manages all system hardware resources (e.g., memory, devices, processors, etc.). It reserves some resources for its own use, and exports other resources in the form of processes, memory segments, I/O devices, raw disk volumes, segment volumes, etc. The TML separates all exported resources into distinct *partitions*, governs the flow of information between partitions and between individual exported resources, and protects raw disk volumes by encryption.

The TML creates three types of partitions: *Emergency, Trusted* and *Normal*. A normal partition and an emergency partition each run a commodity

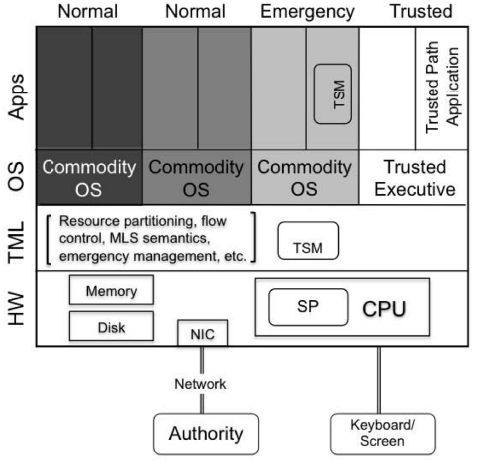

Fig. 1. Security architecture instantiated for an emergency scenario

(untrusted) OS. The Trusted Partition runs an extremely compact trusted executive. Information flow rules and partition definitions provided to the TML define a partial ordering of flows between partitions, indicative of a multilevel security (MLS) policy, and corresponding security labels may be associated with partitions.

The TML manages interactive user sessions through what is referred to as *partition focus*, whereby the user can interact with one partition at a time, via the system keyboard and screen.

For applications in the trusted partition, the trusted executive provides simple OS-like support. The trusted executive manages passwords and provides user authentication services. The TPA is invoked by pressing what is known as the *secure attention key* (SAK), such as Ctrl-Alt-Delete. When the TML detects the SAK, it changes partition focus to the Trusted Partition, which starts the TPA. The TPA provides a variety of other security services to the user, including: setting the security sensitivity label for a session, changing a password, and shutting down the E-Device.

2.2 Trusted Software Protected by SP Hardware

A novel aspect of our architecture is provided by SP's Trusted Software Module (TSM). TSMs have two critical characteristics: they are protected from observation and modification by non-TSM software (including the OS), and they have exclusive access to SP crypto-transforms and to two processor-resident data registers.

A TSM is defined to be code that is executable in a special processor mode called Concealed Execution Mode (CEM) that is entered via the *Begin_CEM* instruction. Once in CEM, special SP instructions can be executed — *SP_derive*, *Secure_Load*, *Secure_Store*, and *End_CEM* — and special SP registers can be

used — Device Root Key (DRK) and Storage Root Hash (SRH). Other CEM-only instructions are provided to read and write the SRH register. Also, when in CEM, the processor checks the integrity of each instruction cache line of the TSM with respect to compiled-in signatures based on the DRK.

The *Secure_Load* and *Secure_Store* instructions cause memory locations to be tagged as TSM-only while they reside in on-chip caches, and to be automatically encrypted and hashed when they move to off-chip memory to prevent unauthorized observation or modification. CEM execution is suspended during an interrupt and is automatically resumed upon return, with data in processor registers protected from the OS by the hardware.

The protection of TSM code can be viewed as being independent from the protection of the underlying OS, as follows. The trustworthiness of a program is generally understood to be limited by the trustworthiness of the programs that it depends on. The TSM, which runs in the Emergency Partition "on top of" an untrusted OS, does not make calls to the OS (i.e., and can not, since the OS is not a TSM) and so a TSM is not "functionally" dependent on the OS, and can actually be more trustworthy than the OS itself.

SP stores two master secrets in non-volatile registers on-chip, which provide the roots of trust for the E-Device's operations. The DRK, which never leaves the processor, protects the integrity of TSM code and is also used by *SP_derive* to cryptographically derive new keys for the TSM. The SRH can only be read or written by the TSM, and provides it with a small amount of on-chip storage, which can be used to store the output of crypto-hashing operations for protecting larger persistent data structures such as key chains (a hierarchy of keys) and the policies regarding use of those keys. This allows the TSM to moderate all access to the protected data and enforce the corresponding policies on each access.

The Authority can communicate with the E-Device during operation, providing updates to keys and policies in its persistent secure storage and to send new encrypted data to the E-Device. When establishing a secure communication channel, the parties use derived keys. Therefore, the E-Device's ability to generate those keys and communicate demonstrates that it still has the correct DRK value and signed TSM code, and serves as an implicit authentication, since only the E-Device and the Authority know the DRK.

3 Information Sharing during an Emergency

Our concept for emergency response involves a network of participants (the *Emergency Network*), including a coordinating *Authority*, the expected *first responders*, and *third party data providers* who maintain information that is expected to be useful during an emergency. The Authority manages the distribution of keys and policies via its own secure computing facility, and coordinates emergency response for a given crisis, including alerting all entities on the Emergency Network of the emergency and disseminating emergency data to first responders.

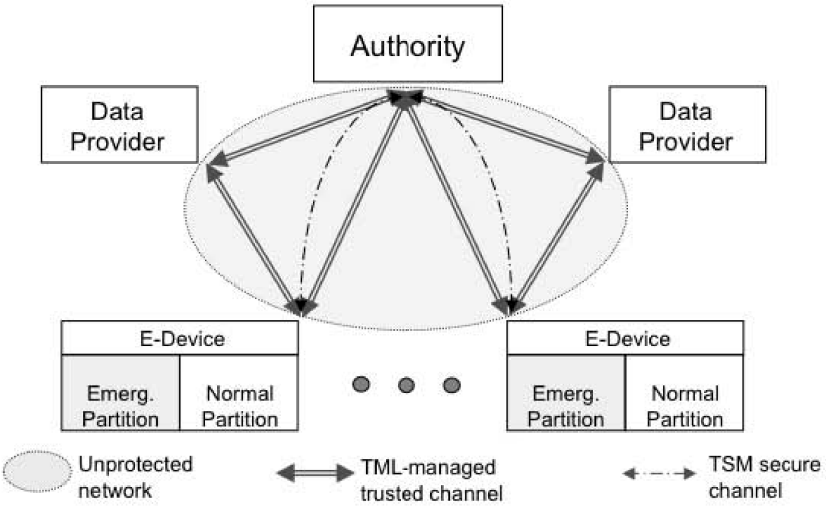

Fig. 2. Emergency Network Architecture

While E-Devices may be owned by various emergency network participants, their correct configuration is the responsibility of the authority, and they will be operated in the field by first responders. Fig. 2 shows the emergency network.

The operation of the Emergency Network is governed by prearranged organizational security policies [5]. These include the lattice-based MLS policy enforced by the TML and the access policy enforced by the TSM in the application domain.

3.1 Emergency Operation

The Authority maintains a binary global emergency state, i.e., ON or OFF, and notifies the authorized E-Devices of any state changes. The E-Devices may grant access to emergency information if the state is ON, and must deny access when OFF. Secure synchronization of the global state is discussed below.

When an emergency is declared, the Authority sends state-change notifications to the E-Devices. Once the TML interprets the message, it prompts the user to access the Emergency Partition, via the TPA.

While the emergency is in effect, the user can access any active partition the user is cleared to see, including the preconfigured Emergency Partition. Within the Emergency Partition, finer granularity application-specific access controls on emergency data may be provided by an application domain Emergency Management TSM.

When the emergency is over, the Authority announces a change to the global emergency state, prompting the TML to start the emergency closure process. It displays an end-of-emergency message which prompts the user to change focus to the Trusted partition (to be completed within a configurable period), revokes access to the Emergency Partition, and restores the Emergency Partition to its original state.

4 New Security Mechanisms

This section presents a systematic analysis of the available secure communication channels and describes mechanisms for completing the trust chain from the remote Authority to the E-device, and to its Emergency Partition and display, including: emergency state management, trustworthy display and mechanisms for revocation of sensitive data.

4.1 Secure Communication Channels

A secure communications protocol protects against message content disclosure and modification, as well as traffic analysis and insertion or deletion of packets.

We consider a channel to be a *secure channel* if it uses a secure protocol, the protocol is implemented correctly, and the channel endpoints are secure against both the modification of their behavior and against unauthorized disclosure of channel and keying information.

The E-Device, while simple, supports several different forms of secure communications channels, which provide emergency systems designers with flexibility in constructing new systems. Three basic channels are shown in Fig. 3: the *Trusted Channel* (A), the *TSM-TSM Channel* (D), and the *Trusted Path* (B).

A Trusted Channel (A) is a secure channel between two TCBs (e.g., a TML or a trusted system) [6]. A TSM-TSM Channel (D) provides cryptographic assurance against message disclosure and modification between application TSMs, e.g., on different machines. A Trusted Path (B) is a secure channel between a user and the TML on the E-Device, implemented by the TPA.

A *Remote Trusted Path* (E) is a secure channel between a remote user (e.g., the Authority) and a TML, which is constructed by combining a Trusted Path with the remote end of a Trusted Channel. An *Extended Trusted Channel* (F)

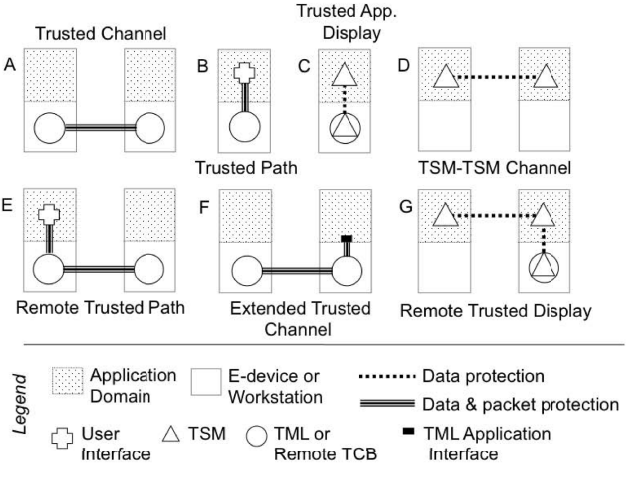

Fig. 3. Types of Channels

is a secure channel created by extending the I/O interface of a Trusted Channel to the TML interface of a given partition, which allows applications to securely interact with a remote TCB.

A *Trusted Application Display* (C), discussed below, enables an application (i.e., the Emergency Partition TSM) executing in the context of an insecure OS to securely write to the screen, which is managed by the TML. A *Remote Trusted Display* (G) connects a TSM-TSM Channel with a Trusted Application Display so a remote system, such as that used by the Authority, can display data to the local user with assurance against message disclosure and modification.

The TML exports to partitions *virtual NICs* (see Fig. 4), which are logical devices, each with an IP address, for use by the OS and applications in a specific partition.

The TML manages the negotiation of session keys and cryptographic algorithms, as well as the cryptographic transformation of data for encrypted channels via the IPsec "security association" paradigm [7], although the entire IPsec suite is not required (e.g., crypto-transforms are statically assigned). An Extended Trusted Channel therefore embodies the mapping of a partition ID to a remote IP address and a security association. Communication channels, display channels and logical devices are configured during E-Device initialization with information such as: each channel's remote TCB address, and security level; various keying material; and the mapping of Extended Trusted Channels to specific partitions. Security levels are also assigned to partitions and physical devices.

An out-of-band distributed "root" secret key that is shared with the TCB at the other end of a trusted channel is the basis for channel session keys [8]. For trusted channels between the E-Device and the Authority, the DRK is used as the root secret. For other trusted channels, such as those with third party data

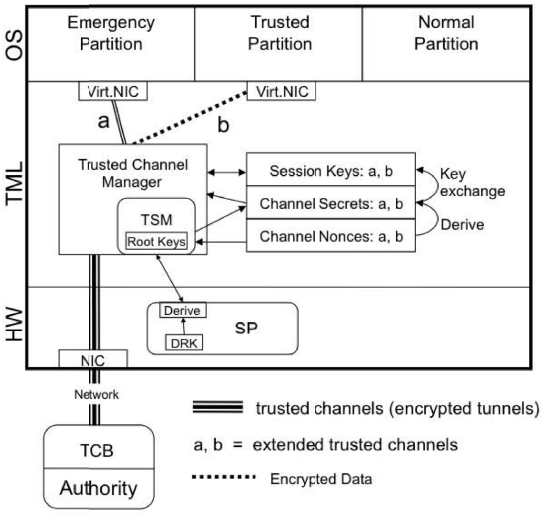

Fig. 4. Networking Support of the TML

providers, a shared secret key is stored by a TSM in the TML's Trusted Channel Manager instead.

Nonces to support generation of session keys are dynamically exchanged or periodically distributed. The TSM hashes the nonce with the root secret to derive a temporary shared secret, with which it can generate a session key using a standard key exchange protocol, such as TLS. Alternatively, the derived value itself could be used as a session key. All communications between endpoints within the emergency network use these channels.

4.2 Emergency State Management

As discussed above, the Authority manages the global emergency state with *emergency state management messages*. State management consists of the following steps: (1) message generation; (2) message transmission; and (3) message processing on the E-Device. Each of these steps needs to be trustworthy to ensure consistent and correct emergency state management.

In addition to the emergency state, the Authority maintains a counter of the number of state changes it has issued; also, it maintains a record of the acknowledged state and counter values for each E-Device, along with its DRK. When the Authority changes the global state, it must securely synchronize with each E-Device. On the individual E-Device, local emergency state and a state transition counter are maintained by a state-management (E-State) TSM in the TML.

When the Authority declares an emergency, it increments its counter associated with the E-Device and generates an emergency state management message for each E-Device that consists of: (a) a command type indicator (indicating a state update message); (b) a payload of the new state and counter, encrypted with an encryption key derived from the DRK; (c) a signature (cryptographic keyed hash) of the encrypted payload and command, using a signing key derived from the DRK; and (d) two nonces to derive the encryption and signing keys.

The emergency state management message is sent to the E-Device through a *Remote Trusted Path* channel. Only the E-Device to which the emergency state management message was intended is able to successfully process the message. The E-State TSM independently [1] verifies the originator of the state management message. For emergency state management, two functions, *Update_State* and *Get_State* are implemented on the E-Device. The Update_State function checks the integrity of the message using the signature, and the counter. It also generates a response to the Authority by generating a signature over the message using a signing key derived from the signing nonce and the DRK. The E-State TSM sends the signature back to the Authority over the trusted channel, but does not need to send the message payload or nonce, since the Authority already has the initial update message.

The TML, via its TSM, uses *Get_State* to retrieve the new state, as discussed in Section 3.1. To ensure that the update of emergency state is trustworthy, only

[1] Decoupling the channel authentication from message authentication allows for flexibility to incorporate ad-hoc and/or peer-to-peer transmission of emergency state management messages in the future.

the TML can pass update messages from the trusted channel with the Authority into the E-State TSM; and only the E-State TSM can invoke *Update_State* and *Get_State*.

For high-threat deployment environments, enhanced assurance is provided by a version of SP that includes state management primitives in its ISA. Instead of implementing them in the E-State TSM software, this version, includes registers for a state variable and a state transition counter, as well as instructions for the *Update_State* and *Get_State* functions. With these hardware enhancements, even the TSM does not have the ability to directly modify the emergency state on the device.

4.3 Containment of Emergency Data

The organizational security policy enforced by the E-Device requires that emergency information from the data providers only be accessible to authorized users acting within the emergency partition, and only during a proper emergency declared by the Authority. MAC policy enforcement (by the TML) and DAC policy enforcement (by the Emergency Management TSM) jointly restrict information flows on the E-Device before, during and after the emergency.

Emergency data is installed at the Authority's secure facility or sent to the E-Device from the Authority and data providers over Trusted Channels, and is confined in the Emergency Partition. The data may be further encrypted so that it can only be accessed by the Emergency Management TSM application, which can enforce more granular access policies within the Emergency partition.

The Authority establishes an *Extended Trusted Channel* to the Emergency Partition. The Emergency Partition and the Authority are allocated an "emergency" MLS label that is distinct from that associated with other partitions. The TML attaches the "emergency" label to data it receives over the channel, restricting the data to only this partition of the E-Device. Data flows for emergency management are shown in Fig. 5, as follows: (1) the Authority propagates changes to the global emergency state and receives confirmation from the device; (2) the Authority sends keys, policies, and revocations to the Emergency Management TSM, via an Extended Trusted Channel managed by the Trusted Channel Manager (TCM); (3) the authority sends encrypted emergency data to the Emergency Partition, via an Extended Trusted Channel; (4) data providers provide additional data for the authority to send to the E-Device; (5) when needed, the Emergency Management TSM decrypts the emergency data with keys and policies in its storage.

Trusted channels between the TML and the Authority are protected using freshly negotiated channel secrets for each connection, based on the DRK rather than a stored root key. As a result, only parties with access to the DRK (the Authority and the E-Device) can authorize an Extended Trusted Channel to the Emergency Partition.

Aside from these secure channels, the information cannot flow out of the Emergency Partition, e.g., to any other partition, device or network, ensuring that emergency data cannot be revealed outside of the equivalence class of

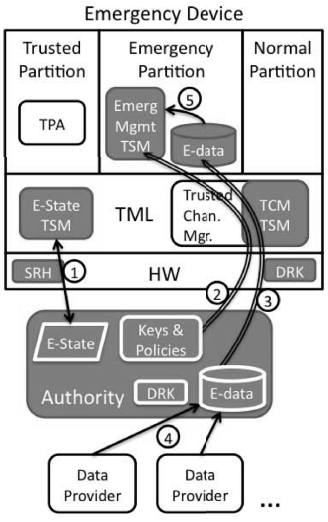

Fig. 5. Data Flow for Emergency Management

components labeled as "Emergency." Additionally, the Emergency Partition itself is only made available to the user by the TML and TPA when an emergency has been declared, as previously described. This temporal restriction limits the threat of malicious insiders, and in combination with the partition's spatial separation, provides defense in depth for the confinement of emergency information.

The Authority provides its own emergency data to the E-Device, and can convey data provided by third party data providers. When the latter data is proxied through the Authority and the third party has no direct communication with the E-Device, the third party does not need and is not given the privileges associated with the "Emergency" label.

If a third party is considered trusted, it can be included in the "Emergency" equivalence class and allowed to establish an Extended Trusted Channel directly to the Emergency Partition. Since the Third Party does not have access to the E-Device's DRK, it and the Emergency Management TSM share an "emergency" key, which is stored locally in the TSM's persistent secure storage. Even when third party data is provided directly, the TSM on the E-Device can still be configured to only accept policies for that data directly from the Authority.

All emergency data sent over the Extended Trusted Channel is encrypted by the Authority or data provider prior to transmission using keys only available (on the E-Device) to the TSM. This enables the TSM to enforce its discretionary access control policy on use of the data by the responder or any software within the Emergency Partition, and audit the use of the data, even though it executes alongside untrusted software in the Emergency Partition.

Within the Emergency Partition, the TSM can release data to untrusted, feature-rich commercial applications for display. However, there is no assurance that untrusted applications will accurately display data when asked. Some information, e.g., that which is critical and easy to manipulate, may require greater

protection. For this, we provide the high-integrity *Trusted Application Display* mechanism, which allows an application-TSM to send text-only emergency data directly to a reserved region of the display via a secure call to the TML that bypasses the untrusted software in the Emergency Partition.

The trusted display mechanism provides an unspoofable means for an application domain program to display messages with high integrity such that they cannot be observed or modified by any untrusted software in the system. This mechanism is available to the TSM in the Emergency Partition and the TPA running in the Trusted Partition. Both the TSM and TPA are designed to be evaluated to ensure their correct behavior, which helps to ensure that the correct data is input to the trusted display mechanism.

The TML virtualizes the video graphics card such that it appears to each partition that it has control of the screen. These virtual devices pass input to the TML's secure display driver, which divides the physical display into two regions. One region is restricted for the TML-controlled high-integrity display (for example, the bottom two lines of text on the screen). The remaining region of the screen is exported to the partition with focus as normal.

High integrity data to be displayed is encrypted and either comes directly from the Authority for *Remote Trusted Display* or is chosen to be released by the TSM during its operation for Trusted Application Display. To pass the data securely to the TML, the TSM is divided into two pieces: an application-TSM in the partition and a kernel-TSM in the TML. The former is responsible for preparing the data for display and the latter for passing the plaintext data to the TML securely. SP's CEM protects the data as it is passed between privilege levels through the untrusted software in the Emergency Partition.

The application-TSM first decrypts the data using keys in its storage, and then stores the resulting display text in a memory buffer (at a known location) using *Secure_Store* instructions. This data is now only accessible in plaintext to TSM code. An x86 call-gate is used to transition from the application-TSM to the kernel-TSM without an interrupt. The kernel-TSM uses *Secure_Load* instructions to read from the CEM-protected memory buffer and regular *Store* instructions to write the cleartext data to the TML buffer. It then exits CEM mode and invokes the TML-provided Trusted Screen Handler, which sends the data to the TML's Trusted Screen Driver for display in the restricted display region. Finally, the kernel-TSM code re-enters CEM and returns to the call gate in the application-TSM, with a return value indicating the success or failure of the display operation.

To complete the data lifecycle, access to emergency data must be rescinded once the emergency is over. Data revocation takes place through complementary mechanisms, using both mandatory and discretionary access control.

The coarsest granularity of revocation available to the Authority is to declare the emergency to have ended. As described in Section 4.2, this results in the closing of the Emergency Partition to users and applications, and restores its code and data to the pre-emergency state. Stopping application activity and

overwriting the entire partition effectively removes all data generated or released inside the partition.

A finer-granularity of revocation is provided by the Emergency Management TSM itself, as described in [4]. Over and above the TSM-enforced policies restricting access to data based on expiration dates, usage counts, search query restrictions, etc., at its discretion, the Authority can communicate with the TSM and direct it to modify policies, keys, and other emergency restrictions to revoke access to existing data, for example, in preparation for ending the emergency.

Guarantees to the Authority and third parties about revocation and the state of the E-Device depend on connectivity and availability of the TSM. If the E-Device is disconnected temporarily from the network, or an application TSM managing communication with the Authority is subject to a functional denial of service attack, the Authority will be unable to synchronize the local emergency state with its own global state. An emergency expiration timer is provided by the TML, such that if connectivity with the Authority cannot be established within a defined time, the TML can end the emergency on the E-Device. The use of this timer may not be appropriate for all responders and is therefore configurable.

Once communication is restored, the E-Device can attest to the Authority that the requested updates to emergency state, policy, and keys have been made.

5 System Security Analysis

Overall system security can be understood in terms of the threats to which it will be exposed and how the system is capable of counteracting those threats.

5.1 Threat Model and Assumptions

We assume that the E-Device is initialized securely with TML-, TSM- and SP-specific keys. We also assume that the third parties and the Authority securely exchange the required keying material and protect their own keys from exposure.

We assume the standard Dolev-Yao model [9], that arbitrary parties can capture, modify or insert network traffic. Intentional or malicious network-level denial of service — as opposed to prevention of process functionality at the workstation — is outside the threat model. The threat model and analysis for SP, which includes spoofing, splicing and replay of TSM code, intermediate data in registers and memory, and secure persistent storage, are discussed in [4], including the protection of emergency data encrypted with TSM keys. The threat model for the TML is that applications of the TML, including guest OSs other than the trusted executive, are not trusted to conform to its policies, and may in fact be hostile. For example, at runtime, the software executing above the TML may attempt to access keys used by the TML to establish secure channels — and application software or the commodity operating systems may attempt to write emergency data to a location outside the partition or attempt to access high integrity information through low integrity mechanisms. The trusted executive is only trusted to manage its applications in a manner that does not introduce

covert channels between applications that are at different security levels. The persistent disk storage is encrypted and signed, and so is protected against, e.g., theft of the E-Device.

TCB software at the Authority, third parties and in the E-Device is assumed to behave correctly and securely. But, application TSMs are subject to inconsistencies in their execution environment, such as denial of access to the processor, as they execute independently from the trusted software that controls physical resources.

5.2 Security Capabilities

The architecture presented in this paper combines the secure execution and key confidentiality provided by SP hardware with information containment assurance provided by the TML. Further, secure boot ensures that the correct TML configuration is loaded on boot and SP's *code integrity checking* (CIC) ensures syntactic runtime integrity of TML code. This ensures that large classes of software attacks that involve code modification are prevented.

The DRK acts as the shared secret between the E-Device and the Authority. On the E-Device, the confidentiality provided by the SP hardware ensures that software never has access to the DRK directly, ensuring this shared secret is always protected with high assurance.

Software is only allowed to use the DRK to derive other keys, which is sufficient for mutual authentication and secure channel establishment. The SP hardware along with the TSM ensures that all computation involving keys derived from the DRK, including intermediate data, never leaves the processor chip in unencrypted form. This avoids the possibility of any software outside the TSM getting access to derived keys. Since this is an invariant of the TSM and SP hardware, keys used for secure channels are always protected. The TML code is protected by the CIC mode of the SP processor, ensuring that the access control polices enforced by the TML cannot be changed by code modification attacks. The privilege levels provided by hardware ensure that subjects with lesser privilege than the TML cannot read or write to objects in the TML. This ensures that the TML always sets up the secure channels between the different endpoints on the E-Device and the Authority/third parties as configured.

A Trusted Path provides bidirectional security: (1) ensuring that user input to the TCB is accurately and securely received by the TCB, via a keyboard-to-TCB data pipeline (where the keyboard is a proxy for the user); and (2) ensuring that output from the TCB is seen by the user, without ambiguity or compromise, via the TCB-to-screen data pipeline (where the screen is a proxy for the user).

The Trusted Path is a secure channel, since it is secure and it provides a direct connection to the TCB endpoint via a secure interface provided by the TML, with no intervening untrusted components. The Trusted Channel is a secure channel, since it is assumed to use secure protocols and the endpoints are secure.

The E-Device has three TSMs: one is an Emergency Partition application, and the other two are TML modules. Since these TSMs provide system security

services, they are considered to be part of the TCB and are built to the same level of assurance as the TML. When they communicate, the TSMs on both ends of the TSM-TSM Channel have access to DRK-derived keys as well as CEM-protected memory, so the receiver can validate the integrity of messages and ensure confidentiality. The TSMs on the E-Device similarly share the same DRK-derived keys with the Authority, and two TSMs on different devices can be provided a shared secret by the Authority. We further assume that a TSM-TSM Channel to the Authority or another E-Device utilizes a Trusted Channel, adding another layer of protection over the network.

Any data used, transmitted, or displayed by an application-TSM is still subject to a functional denial of service by the untrusted OS, which may prevent its execution or tamper with its encrypted data — but the tampering will be detected.

In the Extended Trusted Channel, the TML makes Trusted Channels available to partitions as logical I/O devices, which provides a secure channel between the I/O device interface and the trusted component at the other end of the Trusted Channel. The Extended Trusted Channel can provide plain-text or encrypted data to a given partition (where the cryptographic functions are provided by applications in the partition).

In Normal and Emergency Partitions, a commercial OS manages the logical I/O device, and makes it available to *its* applications via an abstraction such as a socket. The security of the Extended Trusted Channel from the perspective of the OS application then depends on the security of the OS and any encryption of the data.

The Trusted Application Display is a uni-directional secure channel between the local application TSM and the user, assuming continuity in execution of the TSM and protection of TSM message data. Continuity of execution depends on the TSM's processing environment, including the ability of the application domain OS to schedule the TSM and other applications consistently and avoidance of attacks on application-TSM code, data buffers, or communications to the TML. While these would constitute a functional denial of service attack on the TSM, they could not compromise the confidentiality or integrity of the display data. Of course, the confidentiality of displayed data is not protected from out of band analog mechanisms, e.g., visual observation of the screen.

Combining a TSM-TSM Channel with the Trusted Application Display channel results in a Remote Trusted Display channel that is a single-direction channel whose security depends on the security of the component channels (i.e., the TSM-TSM Channel and Trusted Application Display channel discussed above).

Emergency state management message generation is a security critical operation. Only the Authority is able to generate valid messages for a given E-Device as they are based on the device-specific DRK, known only to the authority and the E-Device. Since SP hardware ensures software never has access to the DRK and the authority secures its copy of DRK, arbitrary parties cannot generate a valid message. Since the emergency state change generates a response that is also cryptographically signed by a DRK-derived key, the authority can be assured that the emergency state management was correctly processed by the intended E-Device.

There are several layers of protection that detect and prevent replay attacks on emergency state management messages: (a) all messages are transmitted over secure primary channels such that only the subjects with access to the endpoints of the secure channels can see even the encrypted message; (b) the TSM in the TML could also overlay a secure mutual-authentication protocol between the TSM and the Authority to prevent other parts of the TML from accessing the encrypted emergency state management message; (c) since the message is encrypted using keys derived from the DRK, only the TSM on the correct E-Device can decrypt and process the message; (d) the monotonically increasing counter ensures that a given message is never processed twice.

Once the emergency is declared, and the E-Device successfully changes its emergency state, the Emergency Partition is enabled and the user is able to access it. The TML ensures that the Emergency Partition can write only to channels leading to the Authority and to trusted data providers. Further, no other partition on the E-Device can read content in the emergency partition labeled as "Emergency." The TML ensures only known virtual device abstractions of trusted pre-configured physical devices are presented to the OS in the Emergency Partition, thus avoiding the possibility of the user being able to attach a device with unconstrained information flow properties to the partition.

When the emergency data is decrypted and displayed within the Emergency Partition, the untrusted applications or OS may keep parts of the clear text emergency data in memory and/or write it to disk, but the TML's Emergency-partition separation policy ensures that the data remains in the Emergency Partition. While all data in memory is erased or otherwise invalidated on a shutdown of the E-Device, the data on the disk may still be accessible if the Emergency Partition is still present. Any offline attacks on emergency data on disk are prevented as the TML protects all data on disk by encryption with keys derived from the DRK. These encryption keys are derived as needed and are never revealed or stored. These properties ensure emergency data containment during the emergency. When the declaration of emergency is rescinded, the Emergency Partition becomes inaccessible to the user and its contents are encrypted and stored for audit purposes or immediately deleted.

Applications in the Emergency Partition are not expected to display high integrity content, as both the applications and the OS are not trusted. Instead, high integrity information is displayed on the reserved portion of the screen, via the Trusted Application Display, with data that is appropriately encrypted and hashed. Since the TML manages the physical display, no partitions are given direct access to the portion of the screen reserved for high integrity display.

6 Validation

We have implemented an E-Device prototype that provides a worked example of how the coherent integration of complementary hardware and software security mechanisms can enhance security, and validates elements of our overall approach.

6.1 Prototype Implementation

The prototype demonstrates the feasibility of TSM technology as well as key layering and partitioning mechanisms in the TML.

The prototype TML runs on bare hardware on the x86 platform and provides multiple partitions which the user can switch between using the Secure Attention Key. The partitions each run a primitive trusted executive, providing I/O to the user-space application running above it. Authority-mode SP features, which have been independently prototyped [4], are provided here by a trusted kernel module which emulates its behavior and security properties.

Our prototype implementation of the integrated architecture demonstrates the feasibility of: (a) emergency management operations using remote trusted path; (b) the ability for the Authority and data providers to disseminate keys and data to the emergency partition; (c) the use of the trusted display mechanism provided by the TML to applications to securely display high integrity data; (d) protection of the code integrity of TSMs and its secure storage by the SP hardware; (e) prevention of access to TSM secure storage by the untrusted OS or untrusted applications; and (f) detection of simulated attacks on the remote trusted path, key/data usage policies, and confidentiality and integrity of emergency data using standard cryptographic algorithms. For the prototype, it is assumed that the TML-managed trusted channels for secure communication with the Authority and data providers are in place and the key management messages between the E-Device and the Authority/data providers are pre-computed.

7 Related Work

Previous work in processor-based cryptographic support include: SP [4], separation kernels [2], and "least privilege" security architectures [3]. We note that cryptographic coprocessor mechanisms [10,11] do not provide processor-level protection for system software and data, and may be more vulnerable to attack by elements within the platform. Also, while IBM announced an architecture [12] featuring processor-based encryption for protecting data on chip and in transit to remote systems, little information is available regarding system trustworthiness, or the separation of information based on events or mandatory policies.

The Turaya [13,14] and MILS [15,16] architectures are designed to host commercial operating systems and security services as parallel application-domain entities, with certain interactions between those entities controlled by a microkernel (e.g., L4) and a separation kernel, respectively. The security architecture presented here differs from these efforts in that it does not rely on application domain programs for enforcement of the primary underlying security policy, and it provides an interface for the enforcement of intra-OS least privilege policies as well as inter-OS sharing policies. Additionally, the Turaya and MILS efforts do not address the temporal confinement, revocation, and distributed state-change issues inherent to emergency management of information, and they do not

provide processor cryptographic transformations or processor protection of critical keying material.

Trusted Channels provide point-to-point encrypted tunnels between a TML and a remote TCB that has at least as much assurance of security enforcement as the TML. Trusted Channels are similar to Virtual Private Networks [17] although VPNs are usually expected to support arbitrary and changing sets of network nodes rather than point-to-point connections.

The "trusted path" has been understood as a requirement for secure computer systems since the early 1970s [18], and has been implemented in many high assurance [19,20] and commercial systems [21,22]. In our work, the Trusted Path Application implements a traditional user interface for trusted interaction between the user and the TCB (TML). The Remote Trusted Path extends the user's ability to communicate, from the local TCB to a remote TCB. Our work differs from previous remote trusted path results (e.g., [23,6]) in that the security of the communication is rooted in a processor-resident secret key, for communication with the Authority.

The Xen "hypervisor" provides support for security policies by way of "domains" that are similar to TML partitions. Security labels can be associated with a domain, and a security policy can be defined to describe resource isolation or controlled inter-domain information flow [24]. However, Xen was built specifically to provide hardware-assisted virtualization of operating systems [25] rather than a more generic operating environment with other services, such as those provided by the TML.

With the Extended Trusted Channel we replace the Trusted Path's *human* interface to the TCB with a direct *programmatic* interface, which allows applications to interact with a remote TCB (via a Trusted Channel). Similarly, for the Trusted Application Display, the TML exports a programmatic interface for submitting data to the TCB for display. A TML-resident TSM subsequently decrypts the data and then the TML displays it on the screen in a reserved area. Securing the computer display against subversion has been reflected in early work on multilevel windowing [26], and subsequent hardware and software-supported development [27,28]. Our work differs from these developments in providing a means for an application executing in the context of an insecure operating system to securely write to the screen.

In its Global Information Grid (GIG), the US Government has recognized that, in emergencies, the need to access information may be more important than the need to protect the information, and has developed extensive technical and policy roadmaps to support that vision [29,30]. Our framework for management of emergency information advances these concepts by providing a theory and concrete realization to confine information made available under extraordinary circumstances and to rescind access after the completion of those circumstances.

OASIS provides the EDXL standard [31] for information exchange during emergencies, such as payload and message encryption. Our architecture provides a trusted context for the management of EDXL data.

8 Conclusions

To address the inability of existing IT systems to support integrated information sharing, temporary emergency data access, and secure revocation of that access, we have developed an architectural solution that integrates hardware-anchored cryptographic protection with a high assurance software architecture that provides data separation and security services. We have proposed a secure hardware-software platform for an E-Device that can provide trustworthy dissemination and revocation of access to sensitive data during an emergency.

We described the architectural support for trusted communication channels, including a *remote trusted path* between the authority and the E-Device, as well as *trusted display channels* in the E-device. We integrated the SP protocols for DRK-based key-generation into the trusted channel mechanism to protect the storage of channel keys and ensure the authentication of parties who will gain access to the Emergency Partition.

We presented a comprehensive design for the management of distributed emergency state, which is critical for effective emergency response. We also described the *Trusted Application Display* to allow user-space applications to securely communicate with the user via direct x86-style call gates to a kernel-TSM. We also described the multifaceted containment of emergency data and reliable revocation of access at the end of the emergency, using a combination of hardware and software mechanisms and trust chains.

Finally, we have built a prototype that validates key concepts of the architecture, indicating the feasibility of using commodity mobile and wearable platforms for secure emergency-response data dissemination. In the future, in-depth usability and performance testing, as well as formal system security verification, will further validate this work.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grants No. CNS-0430566, CNS-0430487 and CNS-0430598 with support from DARPA ATO. This paper does not necessarily reflect the views of the National Science Foundation or of DARPA ATO.

References

1. Johns Hopkins University, National center for study of preparedness and catastrophic event response. Technical Report, `http://www.pacercenter.org`
2. IAD: U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. Version 1.021 edn. National Information Assurance Partnership (March 2007)
3. Levin, T.E., Irvine, C.E., Weissman, C., Nguyen, T.D.: Analysis of three multilevel security architectures. In: Proceedings 1st Computer Security Architecture Workshop, Fairfax, VA, 37–46 (November 2007)
4. Dwoskin, J.S., Lee, R.B.: Hardware-rooted trust for secure key management and transient trust. In: Proc. of 14th ACM conference on Computer and communications security, pp. 389–400. ACM, New York (2007)

5. Sterne, D.F.: On the buzzword "security policy". In: Proceedings of the IEEE Symposium on Research on Security and Privacy, Oakland, CA, pp. 219–230. IEEE Computer Society Press, Los Alamitos (1991)
6. CCMB: Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components. 3.1 revision 1 edn. Number CCMB-2006-09-001 in Criteria. Common Criteria Maintenance Board (September 2006)
7. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. Number 4301 in Request for Comments. The Internet Society (December 2005)
8. Badra, M., Hajjeh, I.: Key-exchange authentication using shared secrets. Computer 39(3), 58–66 (2006)
9. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proc. Of 22Th annual symposium on foundations of computer science. IEEE Computer Society press, Los Alamitos (1981)
10. Smith, S., Weingart, S.: Building a high-performance, programmable secure coprocessor. Computer Networks 31, 831–860 (1999)
11. Trusted Computing Group: TCG specification architecture overview. Technical Report Rev 1.2, Trusted Computing Group (April 28, 2004)
12. IBM: Ibm extends enhanced data security to consumer electronics products. Technical Report, http://www.cio.com/article/20075/IBM_to_Offer_Chip_ Based_Encryption_for_PCs_PDAs
13. Alkassar, A., Scheibel, M., Sadeghi, A.R., Stüble, C., Winandy, M.: Security architecture for device encryption and vpn. In: Proc. of Information Security Solution Europe (ISSE) (2006)
14. Sadeghi, A.R., Stüble, C., Pohlmann, N.: European Multilateral Secure Computing Base - Open Trusted Computing for You and Me. In: Datenschutz und Datensicherheit (DUD), pp. 548–554. Vieweg Verlag (2004)
15. Alves-Foss, J., Taylor, C., Oman, P.: A multi-layered approach to security in high assurance systems. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Big Island, HI (January 2004)
16. Vanfleet, W.M., Beckwith, R.W., Calloni, B., Luke, J.A., Taylor, C., Uchenick, G.: Mils: Architecture for high assurance embedded computing. CrossTalk 18(8), 12–16 (2005)
17. Gleeson, B., Lin, A., Heinanen, J., Armitage, G., Malis, A.: A framework for ip based virtual private networks. Technical Report RFC 2764, IETF (February 2000)
18. Bell, D.E., Fiske, R.S., Gasser, M., Tasker, P.S.: Secure on-line processing technology - final report. Technical Report ESD–TR-74–186, The MITRE Corporation, Bedford, MA (August 1974)
19. Solutions, G.G.: XTS-400, STOP 6.0, User's Manual. Getronics Government Solutions, LLC, Herndon, VA. Xtdoc0005-01 edn. (August 2002)
20. National Computer Security Center: Final Evaluation Report of Gemini Computers, Incorporated Gemini Trusted Network Processor, Version 1.01 (June 28, 1995)
21. Gligor, V., Burch, E., Chandersekaran, G., Chapman, R., Dotterer, L., Hecht, M., Jiang, W., Luckenbaugh, G., Vasudevan, N.: On the design and implementation of secure xenix workstations. In: IEEE Symposium on Security, pp. 102–117 (May 1986)
22. Bickel, R., Cook, M., Haney, J., Kerr, M., Parker, T.: Guide to Securing Microsoft Windows XP. National Security Agency (2002)
23. Burger, W., et al.: Remote trusted path mechanism for telnet. Number 07/150966 in Patent. International Business Machines Corporation, Armonk, NY (May 1989)

24. Xen User's Manual. Xen v3.0 edn. University of Cambridge (2005)
25. Barham, P., et al.: Xen and the art of virtualization. In: Proc. Nineteenth ACM Symposium on Operating System Principles, pp. 164–177 (2003)
26. Epstein, J., et al.: Evolution of a trusted b3 window system prototype. In: Proc. of the 1992 IEEE Symposium on Research in Security and Privacy (May 1992)
27. Anderson, M., North, C., Griffin, J., Milner, R., Yesberg, J., Yiu, K.: Starlight: Interactive link. In: Proceedings 12th Computer Security Applications Conference, San Diego, CA (December 1996)
28. Epstein, J.: Fifteen years after tx: A look back at high assurance multi-level secure windowing. In: Computer Security Applications Conference. ACSAC 22nd Annual, pp. 301–320 (2006)
29. National Security Agency. Executive Summary of the End-to-End IA Component of the GIG Integrated Architecture. Version 1.0 edn. National Security Agency Information Assurance Directorate (April 2005)
30. Wolfowitz, P.: Global Information Grid (GIG) Overarching Policy, directive number 8100.1. U.S. Department of Defense (September 2002)
31. OASIS Emergency Data Exchange Language (EDXL) Distribution Element. v1.0 edn, `http://docs.oasis-open.org/emergency/EDXL-DE/V1.0`