A Security Architecture for Transient Trust

Cynthia E. Irvine, Timothy E. Levin, Paul C. Clark, Thuy D. Nguyen
Naval Postgraduate School
1411 Cunningham road
Monterey, CA 93943-5201
831-656-2395
{ irvine, levin, pcclark, tdnguyen}@nps.edu

ABSTRACT
In extraordinary situations, certain individuals may require access to information for which they are not normally authorized. For example, to facilitate rescue of people trapped inside of a burning building, firefighters may need its detailed floor plan -- information that may not typically be accessible to emergency responders. Thus, it is necessary to provide *transient trust* so that such sensitive information is available to selected individuals only during the emergency. The architecture presented here is designed to support transient trust. It encompasses pre-positioned, updateable domains for use exclusively during emergencies along with a set of "normal" domains with different sensitivity levels. Allocated to partitions, these domains are entered via a high integrity trusted path service located in a separate trusted partition. Interaction among subjects in different partitions is controlled by a high assurance separation kernel, and efficient use of devices is achieved through the application of a three-part device model. The resulting architecture enforces mandatory security policies, yet ensures secure and revocable access to a class of information during declared emergencies.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures, D.4.6 [Operating Systems]: Security and Protection

General Terms
Design, Security.

Keywords
Virtualization, Multilevel Security, Emergency Management, Separation Kernel

1. INTRODUCTION
A common paradigm of secure computing is the tradeoff between usability and security. While timely access to accurate information can be a significant competitive differentiator for nations as well as enterprises, the ability to protect that information can be just as significant. Frustration with the inability to access information in recent armed conflicts and civil catastrophes clarified for many decision-makers that the balance between information *availability* and *protection* is a risk management action that should not be constrained by a rigid security policy. Rather, it has become clear to many that the mechanisms for controlling and accessing information must incorporate multiple situational factors to ensure the risk/benefit tradeoff is correctly calculated, and that the decisions must be automated to ensure timely access. [29][40]

We present a security architecture for supporting emergency access to information that incorporates and extends current separation kernel technology to provide high assurance of the confinement and revocation of sensitive information accessed during an emergency, such that the risk of allowing extraordinary access is mitigated. To achieve this assurance we confine emergency information temporally – it can only be accessed extraordinarily during a discrete emergency – and spatially – it is not allowed to leave a special "emergency partition" that is protected by a highly robust separation kernel.[28]

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 actions; promotes usability; and ensures sensitive emergency-support information is protected as it is communicated, processed and stored. The result is a trusted foundation for effective crisis management activities.

To describe our architecture, we first highlight its major contributions and provide background on access control, risk management and the tension between the need for information protection and flexibility. A brief description of emergency operations sets the stage for a presentation of the architecture, which describes its protection domains and the allocation of functionality and policies to those domains. Our analysis includes user interaction, transient trust, the use of hardware, and our prototype. A comparison with related work is followed by a summary of this paper.

2. CONTRIBUTIONS
We describe a coherent security architecture, based on a separation kernel, that features several innovations:

- A special isolated environment for extraordinary access to sensitive information. This "emergency partition" features high assurance of data confinement and revocation, as well as the ability for the user to process data with commercial applications. Confinement is further ensured by closing the

CSAW'08, October 31, 2008, Fairfax, Virginia, USA.
ACM 978-1-60558-300-6/08/10.

emergency partition to users except during discrete "emergency" periods, after which information updates can be relayed to a central authority and the entire partition can be automatically purged.

- A dedicated environment for high integrity processing. Using a trusted path, the user accesses a trusted partition in which simple high assurance applications provide services such as trusted signing and communication.
- Interaction between partitions, whether through communication devices or other objects, is controlled completely by the separation kernel, with kernel assurance. This is a fundamentally different approach from other recent separation kernel security architectures.[22]
- An efficient device model in which the kernel ensures the separation of activities through the virtualization of shared devices, without depending on trusted subject programs. In general, access to a device does not require a context switch to another partition. Also, any physical device that is dedicated to a given partition (viz., not shared) can be passed through to the external domain for direct access by the client OS.

3. BACKGROUND

There are many approaches for controlled sharing of information; however, when the effects of information compromise are high – e.g., loss of life or limb, or other serious damage to an individual, enterprise or nation – policies and mechanisms that can ensure global and persistent protection of the information are required. Research has shown that policies for this sort of protection are representative of a label-based lattice abstraction [8], in which information flow or access control is maintained via labels applied to active subjects and passive objects. A *supporting policy* [9] generally requires that subjects' labels should not be of more sensitivity than the related user's *clearance*. Coupled with an automated security policy that prohibits a subject from observing information of greater sensitivity, [6] the supporting policy ensures that users cannot read "up." Complementary rules [6] prevent accidental or malicious "write down" by automated subjects. This describes the traditional "rigid" security policy mentioned in the introduction. It should be noted that government operational practices supporting this policy do not generally allow clearances to be raised without a corresponding, and very expensive, background check. Similarly, to support global and persistent policy enforcement, object labels are not allowed to change; however, through the use of special high assurance mechanisms and human review, trusted components may move information from high objects to low ones.

On the other hand, a risk-management approach to security might conclude that the benefit of providing a person with information for which he has not been vetted outweighs the risk (e.g., to the information owner) of exposure of that information. In emergencies, it is desirable that such a decision be made automatically, rather than waiting for human review. In a grossly simplified example, a decision engine could be calibrated to allow extraordinary access to a datum if the information availability would save lives, even though the information exposure would put at risk a certain amount of property. (Policies and mechanisms for providing these calibrations are outside of the scope of this paper).

However, this risk-management approach is problematic, since it is not likely in a traditional multi-level secure (MLS) policy enforcement environment that the required emergency access could be accomplished by simply declassifying information or raising user clearances, without appropriate and time consuming procedures. For example, consider when classified information is found to have leaked into a traditional unclassified environment (e.g., see [27][37][38]): the environment must be "decontaminated" by a highly skilled technical team. Such a team may expand the investigation to include all connected environments to which the information could have transitively leaked and all persons who may have viewed the information in any of the environments. Computers and communications systems are immediately isolated and then purged of the information, such that any permissions to or copies of the information are revoked. The users are "debriefed" to retrieve any hard copies of the information they may have made, to ensure they agree to keep the knowledge secure, and to understand the residual risk of users' knowledge. If decontamination is successful, the environment may be allowed to go back "on line;" otherwise the equipment may be confined to the classified environment, or destroyed.

In the next sections we describe how extraordinary access to information can be provided to selected users, and afterward can be automatically and completely revoked without the need to treat the user's environment as having been contaminated.[1] First, we review our concepts for management of information in an emergency.

4. EMERGENCY OPERATIONS

The basic assumption is that many emergencies could be handled more effectively if vital, sensitive information (e.g., detailed schematics of a large building or transit system) could be communicated in a timely and secure manner to first responders.[12] *Emergencies* involve various government and non-government responding organizations, with one organization acting as a coordinating *authority* (e.g., the Department of Homeland Security). Other organizations represent the employers of first responders, or are *Third Party* providers of information that may be required during an emergency. The Authority establishes operating agreements or memoranda of understanding (MOUs) with the other organizations in advance of an emergency, which define the authority, responsibilities and operating parameters for the *emergency response network*.

The field device for first responders is a handheld that is suitable for normal day-to-day data processing and communication as well as emergency response usage (hereafter referred to as an E-Device). The E-device is designed for use by a single user at a time, much like a Personal Digital Assistant (PDA). The E-device supports different security contexts for different roles, as well as an emergency context (discussed below). Whether the E-device is to be passed among multiple users is a matter of policy. The E-device can be configured so that selected partitions would be available only to certain individuals.

During setup of the emergency network and the E-devices, emergency-related information and cryptographic keys are installed on the devices. Emergency-specific data can also be sent from the authority or third parties to the E-Device when it is in the

[1] It is understood in this work that modifications to law, MLS rules and MLS policy enforcement infrastructures may be required to support an emergency access, or "need to share" paradigm that allows special access to information in extraordinary circumstances.[4][29]

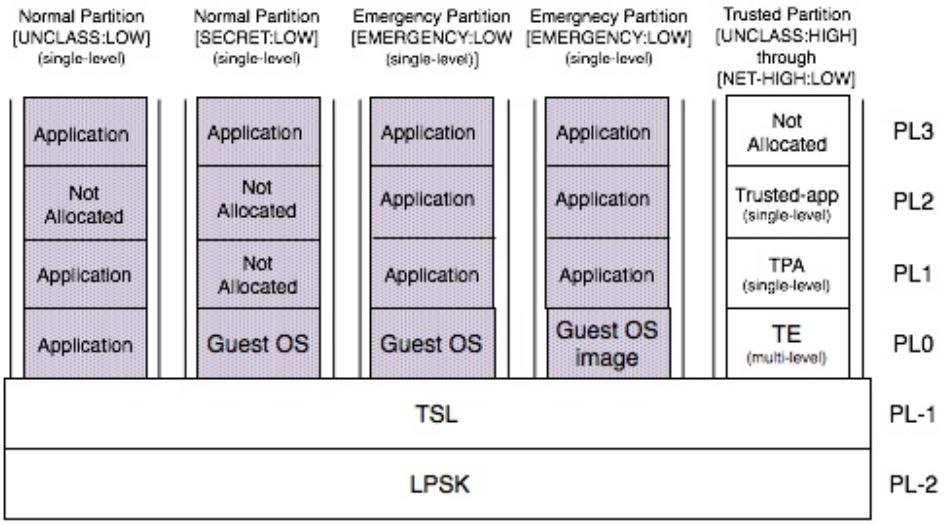

Figure 1. Transient Trust Architecture. Partition sensitivity labels have the format: [SECRECY:INTEGRITY].

field, either during the emergency or to update emergency-related information during normal periods.

5. ARCHITECTURE

The Transient Trust architecture encompasses the Least Privilege Separation Kernel (LPSK) running on x86 hardware (with enhancements in certain configurations [19]), the Trusted Security Services Layer (TSL), the Trusted Executive (TE), and the Trusted Path Application (TPA; see Figure 1).

The TE provides minimal high-level operating system-like services to the applications that depend on it. The TPA provides a trustworthy interface between the user and the other trusted components of the E-Device. Table 1 summarizes the functions allocated to the system layers, each of which is discussed in the sections that follow.

5.1 LPSK

The LPSK is designed to meet the security requirements of the NSA-validated separation kernel protection profile (SKPP).[28] The LPSK virtualizes and allocates shared resources, such that each partition encompasses a resource set that appears to be entirely its own. For physical resources that can only be accessed by one process at a time (e.g., the CPU), the LPSK virtualizes the resource such that the temporal usage patterns of subjects in different partitions are not apparent to each other. For resources such as memory that can be accessed by different processes "simultaneously" (i.e., with respect to a virtualized CPU, or assuming multiple execution cores) the separation kernel may allocate to processes different and non-interacting portions of the resource to ensure separation. Furthermore, kernel utilization of its own internal resources must also preserve the desired separation properties.

The LPSK manages all hardware resources; it reserves some resources for itself; and it exports various resource abstractions at its interface, such as processes, memory objects, devices, etc. The LPSK separates all of the resources it exports into distinct "partitions", and enforces a native policy of separation of partitions, with explicitly allowed exceptions (viz. inter-partition *flows*) that are defined in the LPSK configuration data. As required by the SKPP, these flows form a partial ordering.[28] In addition, the LPSK implements the optional *least privilege* policy requirements of the SKPP, in which it controls access by subjects to individual resources. Hardware task management and other mechanisms are used to maintain isolation between partitions as well as between processes.[20] Threads within a process, if used at the application layer, e.g., by a client OS to create its own processes, are not distinguished with respect to the information flow policy. Eventcount and sequencer objects [33] are exported for inter-process synchronization. As permitted by policy, memory and network resources may be shared by partitions and may be used for inter-partition communication.

If there is more than one process in a partition, they execute according to any of a variety of scheduling protocols that may be defined in the LPSK configuration data until the partition's time slice is finished. As defined in the LPSK configuration data, execution changes between partitions are based on a fixed schedule, or lattice-scheduling algorithm similar to that described by Hu [13], to prevent covert timing channels.

A set of minimized device services are provided by the LPSK. This includes detection of the Secure Attention Key (SAK), which is managed by the TSL (See Section 5.4).

The LPSK accepts configuration data as input during initialization. The configuration data defines the creation and assignment of resources to partitions, data flow rules, partition schedule parameters, memory allocation of segments, etc. The configuration data also defines process initialization information such as per-process *handles* for individual memory objects, devices, as well as addressing information for raw disks (from which, for example, a client OS can create its own disk objects).

5.2 Trusted Security Services Layer

The Trusted Security Services Layer (TSL) virtualizes certain resources exported by the LPSK (e.g., networking devices, I/O devices and disk, as discussed in Section 5.4) and exports dynamic abstractions of static LPSK resources that a typical client

OS would expect to control (such as the creation of memory objects). The TSL API also includes a virtualized BIOS interface to emulate the native (raw hardware) environment expected by client OSes.

The TSL associates a human-readable sensitivity label with each partition, such that all resources in a partition have the same (implicit) label. Labels are assigned in concert with the partial ordering enforced by the LPSK, which can be configured to present a multilevel security policy at the TSL interface, if desired. This combination of the LPSK, providing strong isolation, with the TSL, providing policy-specific semantics, supports the development of "policy aware" [17] and trusted applications in the layers supported by the TSL.

Table 1 Layering Summary

Layer	Functions and Policies
TPA	Trusted Path interface to security-critical services
TE	Application Management User Management Operating System Services
TSL	MLS Support and Interpretation Resource Virtualization Object Management Focus Management Trusted Channel Management Internet routing and Inter-Partition Networking Emergency Management
LPSK	Partitioning of Resources Resource Management Mandatory Access Control Policy Enforcement Process/Partition Scheduling Cross-Partition and Inter-Process Communication Secure Attention Key (SAK) Detection

5.3 Partitions

The LPSK exports three types of partition: normal, trusted, and emergency. For simplicity, only a small number of normal and emergency partitions are described here, but many normal partitions (e.g., several per human-readable label) and emergency partitions can be configured. For example, there could be separate emergency partitions for each type of emergency: flood, fire, windstorm, pandemic, asteroid impact, volcano, etc. Emergency partition granularity would be specific to the first-responder organization.

Each normal partition is assigned a specific sensitivity level and is intended to support the user's regular data processing activities in a manner similar to a commercial handheld or PDA, with a simple client OS that hosts off-the-shelf applications. For example, there may be a normal partition for corporate-internal work, another for collaboration with a business partner, and one for personal use.

The trusted partition hosts a trusted path application (TPA) and other high integrity applications that require a high assurance execution environment, provided by the Trusted Executive. The TPA supports device login, partition selection, and selection of high integrity security services. The secure attention key allows users to invoke the Trusted Path Application without being spoofed by malicious applications. The LPSK services the SAK interrupt and passes control to the TSL, which invokes the TPA. The TPA is configurable to support single sign on, or a per-partition identification and authentication dialog. Of course, most policies would require robust identification and authentication for access to sensitive partitions.

The emergency partition holds data that the user is not authorized to see under normal conditions and which the LPSK prohibits from leaving the partition. In the event of an emergency, the central authority would signal the E-device (e.g., wirelessly) that an emergency has been declared, which would then (and only then) enable the user to "enter" the emergency partition. At this point, access to emergency information would require user identification and authentication and possibly session level negotiation. When the emergency is over, the central authority would signal the E-Device that the emergency has ended, which would once again disable access to the emergency partition, and hibernate all processes in that partition.

During the transition back to normal operations, emergency information created or collected in the field may be transmitted back to the central authority. To purge the emergency partition of information associated with a particular incident, and return the OS and applications to their initial state, the active emergency partition can be overwritten by the contents of the *passive* (i.e., non-executing and "read-only") emergency partition (see Figure 1) which contains a *baseline* image of the emergency partition. The TSL is responsible for re-initializing the emergency partition.

There are certainly scenarios for which the E-device would require additional protections. For example, if it were to be used in combat, then the architecture would be required to include mechanisms that would prevent compromise of emergency information should the E-device fall into the hands of an adversary.

5.4 Devices and Communications

The LPSK provides both synchronous and asynchronous device I/O interfaces and services. To meet high assurance requirements, the device management offered by the LPSK is simple and minimal. In our design, it is not necessary to incur the expense of a full context switch when a partition accesses a device, a possible bottleneck and performance challenge in other architectures [31].

Device management is characterized in three ways in our architecture. First, certain devices may be virtualized by the combined LPSK and TSL. In this case, as permitted by the configuration, the client operating systems and trusted executive may simultaneously share the services of the underlying device.

Second, a device may be assigned solely to a particular partition. For example, memory-mapped devices may be used so that they interact with processes through dedicated memory regions. Here device management is vectored by the LPSK to a client OS for its exclusive use.

Finally, the TSL supports *focus management* by assigning exclusive use of user I/O devices (viz., the keyboard, mouse, and

screen) to a designated partition until the user, via the TPA, chooses to associate the devices with a different partition. While user input is vectored to the selected partition, other partitions can continue to update their screen buffers. Additionally, the system supports configurations with tiled windowing, so that output from partitions with sensitivity levels dominated by the user's current session level can be simultaneously displayed. The TSL will save other partition screen output until the user establishes a session at a high enough level. Thus, instead of having to shut down activity at a particular session level to make the device available at a different level, e.g. [24][25], all partitions can continue to execute according to the predefined CPU-sharing schedule.

Each exported device has two labels associated with it: a read-class and a write class. [23] A device where both labels are equal is a *single-level device*; a device for which the two labels are different is a *multilevel device*.

For *serial-use* multilevel devices, the TSL exports an interface for administrators to change of the "current level" of the device during runtime, thereafter restricting access to the device to that security level. A *concurrent-use* multilevel device can be used by several single-level partitions at the same time. The device appears to (the processes in) each partition as a single level device, as: (1) subjects in a given partition can only read data from the device that is labeled at or below the level of that partition; and (2) data written to the device by those subjects is labeled by the TSL at the level of their partition.

Finally, a multilevel device can be used by a trusted application, which is trusted to apply the right labels to its data.

The TSL manages network devices to establish trusted communication channels between the E-Device and the central authority or trusted third parties. The TSL receives emergency "begin" and "end" signals from the central authority on a trusted channel, verifies the signal and, restricts or allows access to the emergency partition accordingly. It may also raise or lower alarms to other programs within the E-Device, as configured to do so. In this way, the device model is essential to achieving the overall transient trust capability of the architecture.

Partitions may host network-capable client operating systems. The TSL network devices can be used for communications between partitions and to external nodes while ensuring enforcement of the system's information flow policy. For example, since network communication protocols are two-way, in a partition-to-partition communication channel, both partitions must be the same sensitivity level.

5.5 Support for Client Operating Systems

In normal and emergency partitions, the E-Device hosts a client operating system that allows the user to use common applications and file types (e.g., for graphics display and word processing).

The trusted partition hosts a very simple Trusted Executive (TE) for the purpose of running high assurance, high integrity, custom applications. We anticipate that trusted applications will be very compact and present a character-based interface, as the evaluation of graphics-based applications is costly. The Trusted Executive provides a necessary and sufficient set of services to support the Trusted Path Application and possible other trusted applications. These include application management, user management such as identification and authentication support, and a small set of common operating system services.

5.6 Trusted Applications

As discussed, the Trusted Path Application (TPA) presents the user with an interface for high assurance identification and authentication, invocation of trusted applications, emergency notification, and logout, as well as partition selection. Because of the criticality of the trusted path to the correct operation of the E-device, we allocate it to a separate privilege level than that occupied by other trusted applications.

While the E-device evaluation and certification will include the LPSK TSL, TE and TPA, additional trusted applications can be installed by system integrators and other customers to provide customized high integrity services. The layered hardware separation provided by the privilege level mechanism ensures that any additional trusted applications will not affect the trustworthiness of the E-device's evaluated base.

6. DISCUSSION

6.1 Security Policy and User Session Level

To ensure effective enforcement of the security policy, the interpretation of security labels must be consistent across all E-devices in a given emergency response network. While each partition of each E-device may be assigned a different sensitivity level, multiple partitions may have the same label, forming an *equivalence class* of partitions with respect to the security policy, which extends to all E-devices and other trusted computers in the emergency network.

A user request to change partitions results in a "session level negotiation" based on various user and partition attributes (e.g., sensitivity level, temporal restrictions, and roles) and the emergency response network security policy. In a typical E-device configuration, changing partitions effectively alters the user's session level, within the constraints of the user's clearance.

The LPSK's information flow policy can be configured to allow inter-partition information sharing. Thus, during an emergency the user might be able to read information from normal (non-emergency) partitions, or, for example, the user at a SECRET:LOW session level could read information from a partition that is at the UNCLASS:LOW level.

6.2 Transient Trust Assurance

Access to all information is enforced globally across the emergency response network and persistently during data transport, processing and storage. Emergency information is isolated both temporally and spatially. Emergency information is only stored in the emergency partition, where it is confined. Communication between the emergency partition and a remote entity with the same label (e.g., the authority) is allowed, as such an entity is in the same policy equivalence class as the emergency partition. Users are only able to enter the emergency partition during an emergency, and only following appropriate identification and authentication via the TPA. After the emergency, emergency partition processes are hibernated (except when activated to receive updated information), and any data created or collected in the emergency partition can be uploaded to the central authority, and purged from the E-device.

One might posit the application of the architecture presented here for use in embedded devices. Without a requirement for a user interface, the TPA would be unnecessary; however, since organizational policies regarding the handling of emergency indicators are likely to differ, the TE and a trusted handler, though

simpler than the TPA, would still be needed to manage transient trust with high integrity.

6.3 Hardware-Supported Privilege Domains

The Transient Trust architecture uses hardware-based *execution domains,* or *privilege levels* (PLs) to separate the LPSK, TSL, TE and TPA.

In traditional commodity processing environments, the applications and the operating system (OS) are separated into two execution domains. For an Intel x86 CPU, the OS is located in privilege level 0 and the applications are in the least-privileged domain (privilege level 3). However, commercial OSes today are large, complex and lack the structural modularity required to isolate and protect the security policy enforcement mechanism: all system services reside in a single hardware privilege level, where non-critical functions can affect critical policy enforcement mechanisms.

Over thirty years ago, Anderson concluded that, for high assurance of security policy enforcement, the critical enforcement mechanisms must: enforce the policy at all times, protect themselves from unauthorized modification and be sufficiently small and simple to permit analysis of their correctness.[3] Subsequently, no plausible alternatives to this *reference monitor* approach have been proposed. Thus, today's commercial OSes cannot provide a high degree of confidence in their correctness, or in the absence of malicious code. In contrast, the Transient Trust architecture fully utilizes the hardware privilege level mechanism to separate programs according to their relative privilege.

6.4 Validation and Future Work

A prototype of the architecture was developed as the basis for experimentation with different design approaches and for performance testing. The prototype runs on a four-privilege-level x86, and supports multiple partitions, keyboard and screen devices, a trusted path, and least privilege capabilities.

The completed initial prototype demonstrates the effective use of the hardware protection mechanisms built into the Intel x86 processor family [16]. The prototype was constructed using the Open Watcom Version 1.7 C compiler and linker [32] configured to generate 32-bit x86 code in the large memory model, and using the LX file format.[14]. The linker was modified to generate multi-segment data and executables per program (viz., the *large* memory model). The run-time kernel supports multiple hardware tasks; and all four hardware privilege levels, where the state of each privilege level within a task is maintained within the Task State Segment (TSS); hardware supported gates to permit calls to and returns from more privileged hardware domains within a task; and keyboard input and screen output associated with the execution of a selected partition.

While the goal of the Transient Trust Architecture is to provide the user with access to common productivity applications, it is not the intent of this research to develop new techniques for (e.g., Type 1 [10] virtual machine monitoring. The TSL makes use of available hardware support, and para-virtualization [5] where necessary to support the TE and client OSes. In the future, we expect to extend the design to take advantage of new virtualization techniques, e.g., for hosting more complex and heterogeneous OSes. For example, the LPSK and TSL could occupy "negative" privilege levels (e.g., PL0 and PL1 in Intel VMX root operation), while the client operating systems and their applications utilize the classical four hardware privilege levels.

We are working on incorporating a framework into the Transient Trust architecture for objectively calibrating and interpreting the risks associated with exposing sensitive data to users with insufficient clearance. This will provide a means for understanding which objects can be placed into an emergency partition. We expect to experiment with different grades of emergency as well.

7. RELATED WORK

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][40] Our framework for management of emergency information advances the GIG vision 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.

PACER [18] is the DHS *National Center for Study of Preparedness and Catastrophic Event Response*, lead by Johns Hopkins University. Pacer is conducting research into emergency preparedness and response, including enhanced situational awareness and critical decision-making, as well as the causes for the many past failures in emergency response, communication and data sharing. While it is within the scope of the PACER program to develop policy and legal analyses concerning constitutionality, autonomy, and privacy issues, our work differs in its emphasis on providing theory and technology to enable the secure sharing of information in an emergency.

OASIS provides an EDXL standard [30] for information exchange during emergencies, such as confidentiality of payloads, and encryption of messages. The architecture presented here can provide a trusted context for the management of EDXL data.

Few commercial PC-based OSes have used more than two Intel PLs, for reasons that may include: e.g., 1) potential performance penalties; 2) additional engineering costs; 3) lack of understanding of the security benefits; etc. There have been several secure architectures that have used multiple PLs on an Intel CPU.[36][35][24][25] These architectures, however, were built to provide only an application-execution environment. The Transient Trust architecture provides both an application-execution environment and, a virtual machine environment, where guest operating systems may also execute. The architecture provides a device that will not only execute specialized applications in a trustworthy environment, but will also support commodity applications that users are familiar with, while still being constrained by a mandatory security policy.

The Turaya [1][34] and MILS [2] [37] [39]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 architecture presented here differs from these efforts (see also [22]) in that it does not rely on application domain programs for security policy enforcement, and it provides an interface for the enforcement of intra-OS least privilege policies as well as inter-OS sharing policies. Additionally, the Tuyaya and MILS efforts do not address the temporal confinement, revocation, and distributed state-change issues inherent to emergency management of information.

8. SUMMARY

This paper describes the Transient Trust Architecture and related operational concepts for securely managing sensitive information during emergencies. Key software components of the architecture are the Least Privilege Separation Kernel (LPSK), a Trusted Security Services layer, a Trusted Executive and a Trusted path application. The target platform for design and experimentation is a hand-held device, called the E-device.

The key innovative properties of the Transient Trust Architecture are: a means of providing secure, transient access to sensitive information which can also completely revoke the information after the emergency; a dedicated environment for the use of high-integrity applications such as for the trusted signing of documents; a means of directly utilizing the high assurance capabilities of the separation kernel to control communication and other interactions between partitions; and an efficient, flexible device model that allows direct access by guest operating systems to dedicated devices, and secure, kernel-mediated access to shared devices, the latter of which supports both serialized access per security level and concurrent multiplexing without the need for process-level context switching.

The architecture provides a partitioned environment for separating information with different sensitivities. The strictly layered architecture supports commercial operating systems and applications in both normal and emergency contexts, so that emergency workers can be familiar with the E-device through frequent use, and the device might be readily available for emergencies. The LPSK is the nexus of security policy enforcement, and is designed to meet both the basic and the least privilege requirements of the NSA's *Protection Profile for Separation Kernels in Environments Requiring High Robustness.*

9. ACKNOWLEDGEMENTS

We wish to thank Terry Benzel and Ganesha Bhaskara for many helpful discussions. This material is based upon work supported by the National Science Foundation under Grant Number CNS-0430566, 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 of DARPA ATO.

10. REFERENCES

[1] A. Alkassar, M. Scheibel, A-R. Sadeghi, C. Stüble, and M. Winandy, Security Architecture for Device Encryption and VPN. *Information Security Solution Europe*, 2006.

[2] J. Alves-Foss, et. al., A Multi layered Approach to Security in High Assurance Systems. *Proc. 37th Hawaii International Conference on System Sciences*. 2004

[3] J. P. Anderson, Computer Security Technology Planning Study, ESD-TR-73-51, vol. I, ESD/AFSC, Hanscom AFB, Bedford, Mass., October 1972 (NTIS AD-758206).

[4] Assured Sharing Tiger Team. Access Control Concepts for Assured Sharing. National security agency, information assurance directorate 1.0 May 2004

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 20th ACM Symposium on Operating System Principles*, (Bolton Landing, NY), pp. 164–177, October 2003.

[6] D. E. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations and model," Tech. Rep. M74-244, MITRE Corp., Bedford, MA, 1973.

[7] Common Criteria for Information Technology Security Evaluation. CCMB-2006-09-001 September 2006

[8] D. E. Denning, "A Lattice Model of Secure Information Flow," Communications of the A.C.M., vol. 19, no. 5, pp. 236–243, 1976.

[9] *Department of Defense Trusted Computer System Evaluation Criteria*. No. DoD 5200.28-STD, National Computer Security Center, December 1985.

[10] R. Goldberg, Architectural Principles for Virtual Computer Systems, Ph.D. Dissertation, Harvard University, Cambridge, Massachusetts, October 1972.

[11] D. Grawrock, The Intel Safer Computing Initiative. Hillsboro, OR: Intel Press, 2006.

[12] D. G. Holmberg, W. D. David, S. J. Treado, and K. A. Reed, Building Tactical Information System for Public Safety Officials: Intelligent Building Response, NISTIR 7314, National Institute of Standard and Technology, January 2006.

[13] W.-M. Hu, "Reducing Timing Channels with Fuzzy Time," in Proceedings of the IEEE Symposium on Research in Security and Privacy, (Oakland, CA), pp. 8–20, IEEE Computer Society Press, May 1991.

[14] IBM, IBM OS/2 16/32-bit Object Module Format (OMF) and Linear Executable Module Format, Revision 10, October 1996. http://www.openwatcom.org/ftp/devel/docs/lxomf.pdf. (Last viewed: 09 August 2008).

[15] Intel Corp., "Intel® Virtualization Technology Specification for the IA-32 Intel® Architecture", 2005. http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/pentium4/reference/197666.htm (Last viewed: 09 August 2008).

[16] Intel Corp., "Intel 64 and IA-32 Architectures Software Developer's Manual", Volume 3A: System Programming Guide, Part 1, November 2006. http://download.intel.com/design/processor/manuals/253668.pdf (Last viewed: 09 August 2008).

[17] C. E. Irvine, T. Acheson, and M. F. Thompson, "Building trust into a multilevel file system," in *Proceedings 13th National Computer Security Conference*, (Washington, DC), pp. 450–459, October 1990.

[18] Johns Hopkins University, National Center for Study of Preparedness and Catastrophic Event Response. http://www.pacercenter.org/. The Johns Hopkins University, Baltimore, MD, Last viewed 6/23/08.

[19] R. Lee, C. Irvine, T. Benzel, T. Levin, G. Bhaskara, J. Dwoskin. P. Clark, T. Nguyen. "Securing the Dissemination of Emergency Response Data with an Integrated Hardware-Software Architecture," submitted to ANCS.

[20] T. E. Levin, C. E., Irvine, and T. D. Nguyen, Least Privilege in Separation Kernels, *Proceedings International Conference on Security and Cryptography*, Setubal, Portugal, August 2006, pp. 355-362.

[21] T. E. Levin, C. E. Irvine, T. V. Benzel, G. Bhaskara, P. C. Clark, and T. D. Nguyen, Design Principles and Guidelines for Security, NPS-CS-08-001, Naval Postgraduate School, November 2007.

[22] T. E. Levin, C. E. Irvine, C. Weissman and T. D. Nguyen. Analysis of Three Multilevel Security Architectures. Proceedings of the computer security architecture workshop, Fairfax, Virginia, USA November 2 2007.

[23] T. F. Lunt, P. G. Neumann, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley, "Secure distributed data views security policy and interpretation for DMBS for a Class A1 DBMS," Tech. Rep. RADC-TR-89-313, Vol I, Rome Air Development Center, Griffiss, Air Force Base, NY, December 1989.

[24] Final Evaluation Report, Wang Federal Incorporated, XTS-300, National Computer Security Center, CSC-EPL-92/003.B, July 11, 1995.

[25] Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, National Computer Security Center, 34-94, June 28, 1995.

[26] Glossary of Computer Security Terms, NCSC-TG-004, Version 1, National Computer Security Center, October 21, 1988.

[27] John W. Milton. Security Spillage Procedures Defense Message System (DMS). Interim Procedure 21-V02 October 2006

[28] National Information Assurance Partnership, *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness.* version 1.03 ed., 29 June 2007.

[29] National Security Agency, Executive Summary of the End-to-End IA Component of the GIG Integrated Architecture, National Security Agency Information Assurance Directorate, Version 1.0 April 2005.

[30] OASIS, Emergency Data Exchange Language (EDXL) Distribution Element v1.0, OASIS, May 2006. http://docs.oasis-open.org/emergency/edxlde/ v1.0/EDXL-DE_Spec_v1.0.pdf

[31] Objective Interface, MILS: Multiple Independent Levels of Security, http://www.ois.com/Products/MILS-Technical-Primer.html#question6. (Last viewed: 18 July 2008).

[32] Open Watcom, Main Page, http://www.openwatcom.org/index.php/Main_Page. (Last viewed: 09 August 2008).

[33] D. Reed, R. Kanodia, Synchronization with eventcounts and sequencers, *Communications of the ACM*, 22(2):115--123, Feb 1979.

[34] A-R. Sadeghi, et. al., European Multilateral Secure Computing Base - Open Trusted Computing for You and Me. Datenschutz und Datensicherheit (DUD) 9/2004, Vieweg Verlag, pp. 548-554, 2004.

[35] D. D. Schnackenberg. "Development of a Multilevel Secure Local Area Network," Proc. 8th National Computer Security Conference, October 1985, pp. 97 - 101.

[36] O. Sibert, P. A. Porras, and R. Lindell, "The Intel 80x86 processor architecture: Pitfalls for secure systems," in Proceedings 1995 IEEE Symposium on Security and Privacy, (Oakland, CA), pp. 211–222, IEEE Computer Society Press, May 1995.

[37] U.S. Department of the Navy. Information Assurance Remanence Security Publication. Department of the navy IA Pub-5239-26 May 2000

[38] U.S. Department of the Navy. Information Security Program. Department of the navy SECNAV M-5510.36 June 2006

[39] W. M. Vanfleet, et. al., "MILS: Architecture for high assurance embedded computing," CrossTalk, 18, pp. 12–16, August 2005.

[40] P. Wolfowitz, Global Information Grid (GIG) Overarching Policy. U.S. Department of Defense, directive number 8100.1, September 19 2002