Introduction

My name is Chris L. Fleshner, aka http://fleshner.com, and I am a technology developer/consultant. Formerly, I've worked as a systems programmer at the Principal Financial Group, Des Moines, IA, later as a roving consultant, then a patent drafting business owner of Patent Perspectives, Inc. I currently manage my own LAN, servers and network for a newly formed consulting business – fleshner.com. Fleshner.com began operations first quarter of 2012. I have been involved with Backup and Restore technologies, in the form of a Patent Specification, since filing the provisional patent in 2007. I have written one SBIR/NSF proposal No. 10-546, which was not funded. The topic on 10-546 is in reference to the Patent Specification aforementioned, and is also topic of this proposal.

In this proposal I have devised a solution for securely handling mobile devices in a military setting. The selected devices can include consumer grade mobile devices, including, laptops, tablets, smart phones, and other computational and communication devices, each having a native file system, and/or modifiable resources internal to each device.

In this proposal, it is suggested that all modifiable resources on mobile devices be referenced using the IP6 addressing scheme (in a peer-to-peer connection), and are recorded in an Environment Storage Area, or ESA. The resources, in the ESA, should be defined as Modifiable Digital Device Entities (MDDE's), and function as pointers to the original resource. The resource is originally sourced from the resource owner, or resource licensee.

The ESA securely, links, ties and maps out command relationships between the subordinate users and their chain of command, and is one of the strengths of this proposal. It does this by implementing a customized file-based runtime firewall for each remote device dispatched in the field.

The ESA can be deployed as a software object, consisting of all MDDE's for a given remote device. The ESA can support many features concerning usage of remote devices in the field. It does this by keeping track of detail file-information for each MDDE.

The most important feature for the military, though, is that by referencing these MDDE's as Common Internet Resources (using the TCP/IP protocol) and using standard https (port 443) or http (port 80) requests, one can assert anonymity in packet streams on the web. Also, by dynamically encrypting these resources, at runtime, using Military grade encryption schemas, a secure and unnoticed stream of network traffic can occur between device user and the military base.

This proposal identifies many of the potential security problems occurring on the web today, and tries to anticipate problem areas of the future. Because I have integrated broad concepts and ideas, and have identified key areas most susceptible to attack, it is likely the structures described here will adapt, as older technology becomes obsolete, and newer technology is introduced.

As long as TCP/IP continues to be the primary protocol on the web, this design will remain relevant. The next sections will identify some of the problems in mobile device security, followed by a technical Glossary and discussion, feasibility considerations, a summary paragraph, an acknowledgements section, a supplemental illustration, and various notations.

This proposal should challenge you to find security vulnerabilities in the design and I invite you to email me at cfleshner@fleshner.com for your comments, concerns or suggestions or questions.

Problem Identification

When a device is in operation in the field, and that device is lost, stolen, compromised, or inoperative, situations may exist where that device needs to be recovered, controlled, disabled, or rebuilt (onto replacement hardware), or destroyed.

Some of these types of security breaches include, viruses, worms, phishing links, encryption compromise, traffic intercepts (such as packet analysis), reverse engineering, or component extraction/disassembly (by insider threats or outsider threats), and other less-technical intercepts such as overhearing critical operation credentials or parameters, codes or keywords etc.

When a device has been compromised in the field, there may be minimal time to react to the situation, and to disable or re-secure it before the data is compromised or intercepted, or valuable time is lost due to a compromised or hijacked device.

Encrypted device data, can sometimes be dependent on only a few layers of encryption, on an entire file system or volume, and if the cipher is broken, all is lost.

Bandwidth limitations can prevent mobile devices from receiving large quantities of data quickly. In the event a device has been compromised, or needs to be brought back online, securely and with minimal effort, a fast solution to recovery is desirable.

Another security breach is via services or applications planted on devices, which should be prohibited from accessing secure data on the device, or via the device.

IP4 addressing in most environments for a given WAN/LAN, have a relatively small set or scope of IP addresses. This can create security vulnerabilities and other issues, such as Denial of Service Attacks, open port discovery, etc. on military servers or military sanctioned remote devices, because the IP address for the server has been discovered through DNS or some other more sophisticated means of IP address discovery.

There are even “exploit kits”, available to less creative hackers/crackers, offering prefab tools to commit cyber-crime.

The above listed problems and many others, contribute daily to havoc on the web.

Identifying these problem areas, and others, all point to two critical conditions which must exist on the remote device, when securing that device.

1) Only authorized files “exist” on the remote device.
2) Only authorized “entry” to a device has been detected.

Number one is easily dealt with by monitoring the device for “non-authorized” files. Number two is dealt with by identifying the “requestor thread”.

Simply put, if there are any unauthorized (explicit or implicit) files on the device, the device cannot operate in a secure mode. In addition, when an activity or thread into the device has not gained access to that device via the Environment Storage Area, than the device is said to be requesting an unauthorized resource. In either case, the device will be downgraded to operate in a non-secure mode.

The Glossary and Definitions section follows. Please reference this section for further clarification of this proposal.

Glossary and Definitions

This section describes terms and definitions used in this proposal, and are best reviewed before reading the other sections. These definitions are in some cases loosely defined, but are integral to understanding the proposal.

Individual Device User (IDU)

The Individual Device User is the user of the remote device(s), or more commonly the end user.

Initial User Environment (IUE)

The Initial User Environment is a software object containing a variable number of MDDE Constructs, which represent all resources (for the initial device state) of a given RD. The Initial User Environment begins with all presets, applications, and so on before the device is operated in a non-secured area. When a device is confirmed to operate in the clear, an IUE is generated. During runtime, all activity after device boot is applied to the IUE and recorded in an ESA.

Modifiable Digital Device Entity (MDDE)

The MDDE defined in the Patent Application “Automatic Periodic Centralized Backup of User Environments”, was conceived by me, and converted into a Patent Specification by David S. Atkinson, formerly of Suiter Swantz, pc llo, Omaha, Nebraska, in 2007.

The MDDE is an object definition, and would best be described and implemented in software using object oriented programming procedures, compared to standard procedural programming methods.

Two MDDE constructs, describes the container objects (via IP6 association) whose sole functions are to contain MDDE’s, as encrypted objects in a peer-to-peer IP6 address scope, for a device pair (the virtual device and the field device). The MDDE’s for each device, reside in an Environment Storage Area, and are co-located on the remote device in a Virtual Private Server.

The purpose of the MDDE is to securely encapsulate files, in order to facilitate Dynamic Encryption and runtime processes on the file layer of the device, for both the “remote device” and the “virtual remote device”.

The MDDE is comprised of all entities that make up that particular device, and before that device is deployed (for the first time in the field), the device is said to have an “Initial Device Image”, which is at the core of the MDDE’s ability to rebuild a device.

The MDDE (collectively the MDDE object) is critical in supporting how the solutions, claimed in this proposal might be handled. I recommend that the Patent Application, and the Patent Drawings, be referenced as well, while digesting this proposal.

Glossary and Definitions (continued)

Pseudo Files Systems, (PFS) are file systems that are implemented in an OS. All I/O on a device must pass through the Pseudo File System. The Pseudo File System should be implemented as an interrupt request (user exit) whose function should occur during kernel access to file resources on a mobile device. A PFS, implemented in the native file system (for the OS), uses the original filenames and data, from the original locations, except they are encrypted and decrypted at run time, via an alternate location, local to the device, and identified as a Pseudo File System or PFS, having these general requirements.

It must have external (WAN) connectivity to the web
It must be able to manage dynamic ciphers
It must be able to run native applications
It must be able to intercept I/O requests

Mobile Connection (MC) is a TCP/IP pathway, available on any number of networks. For the purposes of this proposal, a general connectivity, using TCP/IP in the field is assumed, using an IP6 addressing scheme, in a peer-to-peer connection.

Remote Devices (RD) for the purposes of this proposal are any devices that can travel and function without being tethered to power and can operate independently of any tether. The remote devices in this proposal requires a minimal system configuration containing a mobile transmitter/receiver, a Bluetooth transmitter/receiver, and local device storage, having a native file storage system, and external interfaces to remote storage.

Personal Restore Device (PRD) is a duplicate military issued RD, having identical or compatible hardware, which may be coupled to an ESA. The purpose of the PRD is to provide the IDU with an immediate duplicate device in the event the RD fails in the field. The PRD is effectively a runtime clone of the RD, and will use the ESA, to determine restore options up to the point of failure. The PRD may be carried on the IDU, or be held with the officer in charge of the deployed unit.

Virtual Remote Device (VRD) is a mirror of a remote device that resides on a VPS, having unique ciphers, on the PFS of the VRD, and on the PFS of the RD. The VRD/RD is contained in an ESA, on a Virtual Private Server (VPS), in an IP6 address scope, using a peer-to-peer connection, and referenced as a Virtual Host. The Virtual Host has two sole clients, the PRD and the RD.

Personal Authentication Device (PAD) is a proposed small electronic attachable device, which may be clipped or pinned to the lapel or elsewhere on the uniform of the IDU operator. The PAD is a separate entity from the remote device, and requires Bluetooth to operate. The PAD transmits IDU related information to the RD via Bluetooth.

Virtual Private Server (VPS) is a server that processes queries from a RD. These queries pass information in the form of a Validation Signature, and return an MK. A VPS hosts a single VRD device and each VRD can be deployed as single Virtual Host (VH) in a VPS. There is a 1:1 ratio for the DNS lookup, of the VPS and the corresponding RD PFS.

Virtual Host (VH) in the context of this proposal is a host that serves authentication content to an RD or a PRD, and uses a secure URL in the form, **https://vh.vps.com.** The "Virtual Host", is a hash function, to a string of concatenated values in the form "RD.FILEPATH". The file path is the resource being acted on in the RD/VRD during Run time events. The vps domain is a dedicated device-level server, which processes the RTE's, as they occur on the RD, and the PRD.

Glossary and Definitions (continued)

Runtime Event (RTE) is an event (occurring on an RD or PRD), which is also mirrored in a VRD. An RTE is characterized as an OS layer I/O interrupt, in the kernel portion of the OS. The RTE can trigger a write operation, of the output buffer, using an encryption dynamic cipher string (DCS), and writes the data, encrypted to the native file system, in the form of a PFS. Conversely, an RTE can trigger a read buffer operation, of the PFS, using a decryption dynamic cipher string (DCS), and returns that data, unencrypted back to the input buffer, for the requestor thread.

Dynamic Personal Authentication Key (DPAK) is a dynamic authentication key defined for a VPS, administered by the security personnel, IT staff, and/or the IDU. The key couples directly with a PAD Device, which is coupled to the IDU. The key accounts for "locality" type parameters, which ensure a device is being operated in authorized zones, and under authorized conditions, by authorized personnel. The locality parameters are embedded in the ESA, on the VPS of a military base.

Dynamic Cipher Strings (DCS) are defined as encrypt/decrypt strings, coupled to military grade cipher algorithms, whose values change dynamically, based on RTE's occurring in a remote device. These strings are provided, in the kernel process, and are supplied during fetch operations of microprocessor level instruction sequences, relating to input and output. The strings are sourced from the ESA.

A continuous loop referenced, has a variable span of time, related to the "on time" of a given remote device, and terminating with the "off time", of the same remote device, and can be tracked using device to base communications. Various events which initiate at device boot time, and terminate with device shutdown or system "freeze", are processed, and characterized by the series, 0,1,2... [Off time].

Each device generates its own unique set of operational sequence numbers, and these sequence numbers change, while "looping" (fetching instructions), in the kernel of the mobile device. A register, that correlates to each sequence number accrues, and provides an index into the MDDE constructs, embedded in a native file system, and also mirrored as a PFS in an ESA.

Specific RTE's occurring during this "loop" implement the authentication checks, using the "last state" indicators, thus providing the kernel further "authority" to continue processing instructions, or to drop the RD back into a less secure mode of operation.

The MDDE construct may contain important details about the origin of the file because it is defined in an ESA. Generally the MDDE tracks who sourced the data, and which application sourced it, what type of thread sourced it, etc. Basically, anything known in the environment when the file was created can be known in an MDDE, and extracted and posted in an ESA.

This is of particular import when rouge system files are introduced to a device. Creation of the ESA (MDDE sources/mirrors), are beyond the scope of this proposal, suffice to say that such a construct is well defined in the patent specification "Automatic Periodic Centralized Backup of User Environments". Therefore an "accountability" of resources is made available through organizing these resources, and tracking their points of origin, in an ESA.

Validation Signatures (VS), consist of 3 states, which are determined in the field during operation of an RD. The states are identified by the source for each state; PAD state, the ESA state and an OP state. The PAD state is sourced from the PAD, the ESA state is sourced from the ESA, and the OP state is sourced from the RD. Each state, sent collectively, as the VS, to the VPS represents a return state, from the ESA, and is identified via the return hops as an MK.

Glossary and Definitions (continued)

Match Keys (MK), consist of the sum of the states VS, from a PADM, VPS or RD. The returned MK is in the form of a "nominal state" or an "error condition". The "nominal state", indicates to the RD, that it can continue to run in secure mode. The "error condition" drops the operation mode of the RD to non- secure. The nominal state, zero (0), and the error state one (1), for each of the sources above, are summed, and collectively are identified as the MK. Thus any non-zero MK, represents the RD operating in non-secure mode.

Non-Secure Mode (NSM) consists of a mode of operation of RD or PRD devices. The NSM mode is dynamically detected. Resources defined as "unauthorized", based on a flag embedded on the ESA at the military base, for a particular MDDE, will drop the IDU device to a Non-Secure Mode of operation. The Non-Secure Mode of operational mode may restrict things such as Wi-Fi access, application access etc. or even initiate device shutdown.

Process Stack Restart Logs (PSRL) are logs that enable restart of a device to a secure mode, and are characterized by "N" seconds back, to the point of device compromise. These logs are necessary, in order to allow reboot of a device that dropped out of a secure operation mode for one reason or another. That is they provide a pointer "n" MDDE's units *outside* of the location of the Initial User Environment. See Fig. 3, of the Patent Specification.

Dynamic Encryption Engine (DEE) handles all encryption for file level input using dynamic ciphers that are supplied to the engine via TCP/IP level packet queries to the VPS. The VPS, queries the ESA, to obtain the file-level crypto information for write operations, and logs these to the PSRL.

Dynamic Decryption Engine (DDE) handles all decryption for file level output using dynamic ciphers that are supplied to the engine via TCP/IP level packet queries to the VPS. The VPS, queries the ESA, to obtain the file-level crypto information for read operations, and logs these to the PSRL.

OP Module is an OS augmentation, on a RD, that intercepts I/O functions. These functions are redirected to a DEE or a DDE, for encryption and decryption type processing. Depending on the OPCODE of the requestor thread, the OP Module will initiate either a READ PFS or a WRITE PFS operation.

A Requestor Thread (RT) can be represented as a unique number tied to a given RD/MDDE in time. A requestor thread can be authenticated using the MK authentication procedures. The requestor thread is used to authenticate an MDDE at runtime. When an application running on an RD encounters an MDDE resource, it must determine if the application doing the request for the resource is an authenticated application, and it does this by ensuring the application doing the requesting is on the local device, and is an authentic source for that resource.

A Runtime Firewall examines MDDE data and authenticates it using the ESA. In the ESA, the application and the process thread of the application must be verified. The thread of the application is a number that is correlated with the external validation processes (using the MK). Thus if conditions were such that a device status had changed from secure to non-secure, the RD would cease to function in secure mode, even though the application itself agrees with the authorized applications.

The distinction between traditional firewalls and the runtime firewall described in this system is clarified here. The RD will not continue to run in authorized mode, when it encounters conditions (external to the device) which contraindicate a secure environment. This occurs by using the construct identified as a requester thread. This insures a file having the same signature as valid resource, doesn't run as being authorized on the target device, unless the request to run is authenticated outside the local scope and inside the local scope.

The "OS Augmentation"

/*

While the RD is operating in the field, a buffered area of binary digits identified as The Dynamically Expandable Bit Area (DEBA), grows and shrinks as I/O processing is occurring on the RD. For each file read, a series of three bits are generated. Each bit indicates either a 0 or a 1, depending on authentication queries made locally and remotely (to the ESA). The shrink/growth rate of this buffer depends on the rate at which run time events (RTE's) are processed on both the RD and the VRD. As this buffer grows and shrinks, a user exit placed in the remote device on the kernel layer and will check the DEBA, to ensure all values are zero. The placement of this check (in the kernel) will be for every clock tick, as opposed to the read and write op codes processing for a given RD. In this way, the RD isn't dependent on a response from the RTE, during the instruction fetch cycle. Also, since only a single check for zeros needs to be in place in the kernel, at runtime, the processor on the RD will not be overly burdened, when determining if the device is operating securely.

/*

Click here for The Code (modeled in PHP for demonstrative purposes only)

Technical Overview

The ESA, or Environment Storage Area, is an area which can be administered by the military, and explicitly defines resources that may be used on a remote device.

These resources are in the form of links, linking back to the creator of the resource. In this proposal we call these links MDDE's. The MDDE, for a device, runs the gamut of information on a device, and is not limited to the file system. As long as there are methods, from vendors, which allow areas other than File System Areas, to be modified, flashed, etc., then the MDDE can be defined for that "entity" of the device. Specifically, where any type of data on an RD can be modified using a program, utility or interface, then that data can be defined as an MDDE.

As we see, device data can exist in an MDDE structure, but this data should be encrypted. The answer to this is to incorporate a dynamic cipher system (DCS). The DCS can help to keep data local to a device secure, in the event a device is lost in the field, or taken from the Individual Device User (IDU).

We assume military IP traffic can be masked on the global internet (because of IP6) and the mammoth range of addressability defined therein.

For the purposes of this proposal, we assume a huge scope of IP6 Dynamic IP's, to point to MDDE resources, and these resources are encrypted and decrypted using the DCS. What's more is that we intend on encrypted and decrypted processes, running at runtime to keep the transmissions secure over the wire.

An additional security measure is the use of the PRD. The RDRD can remain secure even if it falls into the wrong hands, by using the DCS, by requiring cross-references to external military resources.

The proposed DCS does not require specialized hardware, or software, and can make use of existing hardware and the existing device-level military grade encryption techniques, already developed.

Instead the DCS targets all stored data or newly generated data on the device, while it is operating in the field, and manages securing the ciphers for the encryption. The encryption algorithms in this case are treated as components to the overall solution.

Using the ESA, I also describe how a mobile device, could be tested for compromise, occurring as the result of virus, worms, phishing links, type code being introduced into the device environment. This is done by checking the device for the existence of things that are allowed, rather than for a list of things that are not allowed.

These problems and many others can be dealt with programmatically, when using an ESA, coupled to a RD mirrored on a virtual host, using file-level run time events (RTE's), on the kernel layer, because of the logical arrangement of MDDE's with respect to the Initial Device Environment structure.

The technical discussion that follows, offers more details on how these components can work together to solve the problems related to security, in particular by using popular mobile platforms and software.

Technical Discussion

This section discusses how the high level components in this proposal, interact with the mid-level and low level components of the system. These sections address security issues with mobile devices, as they exist in business today. This approach is taken because app compatibility is the main goal of this research.

The other goal, the most important one, is security of these devices and networks in the field. In addition, we analyze the topic from the perspective of current RD's vendors, used in the private sector, and how a change to "cloud administration", may be necessary, in order to support military owned RD's.

This proposal embraces the device environments of the main companies in the mobile world; Apple, Google and Microsoft. But it does this knowing that these corporations have a vested interest in marketing to corporate entities and government.

A problem with this is that corporate autonomy may conflict with the military chain of command. Therefore, it is suggested that military administration of cloud based resources working in conjunction with RD providers might be the best alternative to security management.

Each company listed above, effectively has a closed environment, with respect to handling devices in that environment, especially when it comes to authentication aspects and particulars of their devices. For example Authentication for device functions using domain level accounts is commonly placed under the domain of the "App store" for that particular company (i.e. devuser@gmail.com).

Fortunately, this proposal works with the building blocks of these environments, but ensures that a military based environment remains secure, and is not easily exposed to internal tampering.

The distinction between using an ESA and using a cloud environment, is that cloud based resources have an *assumed authority* to delegate data, within the device environment, whereas with an ESA, cloud based resources have an *explicit authority*. With explicit authority, devices can be shut down by military IT personnel, quickly and easily. It is proposed, therefore, that the military use an ESA, in partnership with one of the above mobile device companies, using an Environment Storage Area, referencing a Cloud Storage Provider.

In this way, storage of the cloud data remains independent to the authority in which it is allowed. Thus cloud based storage paradigms are not disrupted, and can continue to provide their benefits and purposes in already deployed environments (military or civilian). But the resources accessed, and the authentications of those resources are brought down to the device layer. The environments, in which manufacturers of these remote devices were developed, must continue to function as normal using their various cloud implementations. Since this is a security concern, I propose this is best handled by an intercept of cloud and resource based activity over the web, on the device layer!

One main reason for this is to avoid duplication of code and systems that are adequate for both military consumption and civilian consumption. Another reason, is by having alternate cloud sources for identical resources, versioning issues, cache issues, etc. may be harder to track down. And finally, the work required to adapt these environments to military based clouds, could be extensive, and expensive. Thus, it is proposed that if consumer grade devices are purchased, for the military, and those consumer devices and their "apps" are expected to run securely on those devices, then it is best to allow those RD's to continue to work natively, but place the authority of file access directly within the chain of command, and the RD, via an ESA! Internal "app" development, at the military can continue to thrive, within the private sector or within the military.

New Constructs Assist in Authentication

The Environment Storage Area (ESA), the Operating Environment and the Vendor Repository, are all areas which may be used to track files used on devices. Each of these areas is defined to help manage large quantities of devices that share same or similar device configurations and resources.

Thus, the purpose of these areas is to enable the network containing these devices, an ability to track resources, and authenticate them where necessary, when encountered in a field device.

Additionally, these areas (or objects) can expose many details, otherwise not known about the file, in its form on the file system. Details such as who created the file, platform specific details about where the file was last written, and even firmware or BIOS information, about the thread that wrote it or from what file system the file last resided on. In some cases EXIF type data can be used to populate an ESA.

Thus these details are embedded in an object defined as a Modifiable Digital Device Entity, or MDDE. In the life of an MDDE, these details can accrue in time, all-the-while tracking helpful details that could assist a computer OP module, on the kernel layer, in authenticating files at runtime.

Please see "Automatic Periodic Centralized Backup of User Environments, for the full specifications on the MDDE and what can be contained in it and how those areas are generalized, such that a system can track these parameters.

I acknowledge that this proposal is an adaptation of the MDDE construct to a security subsystem and that this is the first such adaption since authoring/defining the MDDE object in 2007. No current software defines the object, acts on the object or otherwise handles these objects; however a clear definition of these objects is available in the patent specification.

The ESA can reside on a portable device, at a remote location, and/or on a military base, so replacement images can be easily populated, when necessary, using and ESA on a PRD or using an authenticated wired/wireless connection to the device mirror, on the ESA.

Processes and procedures for the initial build of an ESA object, are beyond the scope of this proposal, and project planning of this phase, depend in part, on the types of devices chosen to be in the field, and the "kernel" layer research aforementioned. A list of authorized applications, and authorized websites would need to be built using the "Vendor Repository" and the "Operating Environment" sections of the ESA.

Explicitly and by deduction, the MDDE object, via an ESA, can know other details about file resources, including its' origin. Thus, runtime decisions relating to the "resources' authenticity" can be made. This extra information could be encapsulated in an MDDE object.

The MDDE, using a concept where each device can be represented by a secure tree of IP6 addressees, in an ESA, an Operating Environment and a Vendor Repository, is the crux of this security solution. See the Patent Drawings, FIG. 1, and FIG. 3, for a conceptual diagram of the ESA and MDDE respectively.

Vulnerabilities of Remote Devices in the Field

The security layers discussed here are not intended to supplant any security techniques in operation today, but rather to supplement them. I believe that the most logical way of securing a remote device, where compatibility of existing applications is of import, is to attack the problem by identifying these vulnerable areas of mobile device operation.

The file system(s) on the device (native or custom)
The file names on the device (read or write)
The file data on the device (read or write)
The Internet Resources accessed (e.g. Universal Resource Locators)
The Applications Programs and OS Code
The device operator (or IDU)
The encryption schemes used (internal/external)
The IP protocol layer (TCP/IP)
The internal/external interface layer (e.g. SATA III/USB)

A Secure Mode of Operation using Discreet Queries

This proposal handles secure operations of the RD, by defining a "mode of operation", using separate layers of query activity. The security mode can either be "nominal" or "error". By using separate communication channels, and by combining these communication channels into a single entity, the MK can be asserted.
When MK is zero, the device is said to be operating in a secure mode. When the MK is not zero, the device is operating in non-secure mode.

Each query layer collectively provides "kernel level authentication". This is accomplished using data from the RD and the VPS, the PADM and the RD, and the OP and the PFS.

There are several types of queries, and their sources vary depending on the query type. At any time, the queries will have a last state, and this last state is always combined with all of the "last states", into an MK from a previous check. That is to say an MK is a return code comprised of source-queries that have a variable amount of time they may take to answer the query. This is because there are unique conditions, for each query, but the "last state" is always combined with an MK, from previous state changes.

Said another way, Each MK "sub-state", is a separate network query, or RTE, so the combined state is a dynamic state where the last "combined" state is determined based on queries that are occurring in real-time, and whose response times vary depending on the connection speeds (for the DPAK), server loads (on the VPS), and clock speeds (of the RD).

Obviously, the last state, when 0, and combined with other 0 states, will always yield zero. But if any one of the communication channels yields a 1 (error condition), then collectively the device would then drop back to non-secure-mode.

Authenticating by Cross Referencing Various Sources

I believe authentication of a remote device operating in a secure mode should be dependent on several areas, external to the device itself. Authentication states, for a mobile device, are identified and retrieved from these areas in the secure system. The PAD state authenticates the DPAK, the ESA state authenticates the ESA (either PRD or RD), and the OP state authenticates runtime resources on the PAD

The last PAD state is a return condition from the PADM either consisting of a nominal return, or an error return, state. The PAD sends information to the remote device in the form of a DPAK. This is a paired connection between two Bluetooth devices. The PAD transmits the DPAK, and the remote device receives the DPAK.

The last ESA state, is a return condition from the VPS (coupled to the field device), and also consists of conditions, nominal and error. The ESA state, acknowledges communications occurring between the VPS and the RD, and is communicated through a peer-to-peer IP6 connection between VPS and RD.

The last OP state consists of nominal and error state as well. The OP state insures the file and memory on the local scope of the device has not been compromised. This state is tightly coupled to dynamic encryption and the PFS, and to the device OS, and to the ESA, on the VPS.

In order to validate that the right person is operating a remote device we use a Biometric Identifier (BIOD) or a Carrier Identifier (CID), coupled to a Personal Authentication Device (PAD), which communicates with a mobile device via Bluetooth. A BIOD is comprised of information for devices being operated by military personnel and a CID is comprised of information about devices operated by non- military personnel, or in general non-biological entities.

The PAD is defined as a wireless device (containing a Bluetooth transmitter) which is worn on the device operator or “frame entity”, and also may contains an embedded GPS, for proximity checks relative to the frame entity controlling the device. Other localized information depending on sensors embedded in the PAD, could cross-reference mission files, thus further validating device usage in the field. “Flexible Plastics” is one research area at the NSF, where such a “wearable” device could be solicited.

A PAD transmits information about the IDU in the field by associating that device with an officer in a military base, and contains a IDU profile at the base, and other relevant profile information, (coupled to the IDU/device pair), using the VPS.

The PAD device, and its’ corresponding RD, are two device-elements dependent on another to operate. With this dependency, an additional layer of security is present (if the RD is separated from the PAD). Bluetooth, being a short proximity wireless technology, fits the bill nicely, as the mechanism to communicate DPAK data to the PADM of the RD.

Personal Authentication Device Module (PADM)

The Personal Authentication Device Module, (PADM) running on the PAD, has a primary function to create a DPAK. The DPAK is sent via Bluetooth, to the mobile device. The mobile device combines the DPAK key with the IP6 address of the VPS, corresponding to the mobile device, and establishes a secure connection, to the VPS. The VPS returns a Match Key (MK), which is checked in the RD, for a nominal or error condition. The RD continues to run in secure operation mode while the MK is nominal; otherwise the RD will run in non-secure mode.

The VPS mirrors the activity in a VRD, in the event a rebuild is necessary. This mirror is represented in a secure ESA object on the VPS.

The local storage on both mirror and device are independently/dynamically encrypted, and synchronized using messages sent during operation of the PADM. Therefore the DPAK is continually encrypted during secure mode of the device, and the DPAK is integral to secure operation mode.

The VPS can be brought online (using VPS server commands) or similarly brought offline, and this would only effect the one IDU, assigned to the IDU Security Personnel, because of the 1:1 correlation between device and VPS (IP6 address).

I propose using Dynamic DNS servers to handle IP6 name resolution between the VPS and the RD, in the form **https://virtualhostid.dyndns.org**, for example, for a feasibility study.

The Pseudo File System (PFS)

The PFS could be implemented in the OS, where I/O requests to media devices storage or RAM (internal or external) occurs within the operating system. The location in software where these changes would be made, is generalized in this document as “the kernel layer”, but may or may not be on that layer depending on what OS is installed on a particular remote device.

A PFS resides in the actual file system, native to the remote device, but is continually encrypted and can’t be interpreted by any OS, without wireless access to a secure VPS.

Successfully accessing a PFS can only occur, with wireless connectivity to a military server, whose operator is in close physical proximity to the mobile device, and who has obtained VS, from a military server, and whose VPS is up and running.

The PFS can be read and written to using a kernel layer interrupt functions, for an actual file system, on a given device, and requires no additional parameter requirements from the requestor thread (i.e. the application or service running on the device).

While using a PFS, device resources are read from and written to, using unencrypted filenames or paths, via the requestor interrupt (application program), but these paths are available only at the instance where the kernel has encrypted or decrypted it.

The Pseudo File System (PFS) (continued)

In the case of reads, the return data is returned unencrypted via the data bus to the application which requested the data. For other write operations, to the file system, the I/O buffer is encrypted, using a dynamic cipher. In both cases, after the operation is complete, a new dynamic cipher is generated, so for each I/O operation for a given file, the filename and data are encrypted on the native file system, and only at runtime would they use the new cipher.

At write time, the Pseudo File System, will receive the unencrypted write buffer, (via the OS), but only when the data is actually written, will the PFS perform the encryption. Data and files/filenames being written to the actual media on the device (micro-sd, internal device memory, or even RAM) will always be dynamically encrypted

The data, when written, will be in the form of an ESA/MDDE construct on the actual device, and a wireless transmission will be sent to the VPS, which will mirror the cipher operations.

Application layer, RAM based I/O, in mobile devices is only as vulnerable as the applications defining it. But security concerns with apps using RAM data, to emulate shared file systems, for example virtual hard drives (with external SAMBA shares), may be introduced as applications on the mobile device.

Also, browser based cached data, emails, or email attachments, etc. can all be alternate sources of virus code being introduced in a remote device. For this reason, I believe tracking the source of the data is paramount to a successful security implementation. The MDDE construct can assist in handling situations such as this.

A secure PFS and a dynamically encrypted file coupled to the MDDE construct may augment other security solutions, already in place at the military. The PFS is described, and ensures compatibility with the militaries approved remote applications, and thus apps and other software can continue to function without modification, on compatible devices.

I propose, by focusing on the security of the data on the mobile devices in the field, and not on just the computer hardware in the field, decision makers will end up with better ROI, on projects taking this approach, especially due to the constant influx of new and improved remote hardware devices.

The private sector is more than adequate in developing hardware devices that can serve as secure military devices, provided the right changes are made to the software controlling these devices, and that the appropriate network protocols and Internet Network Infrastructure, and environments are in place.

Security is more than just a military concern; it is also a national concern, for the American Public. And because this solution requires that sources of data be secure, and that the device user has waived their right to privacy, as a military sanctioned end user, it can be asserted that the sources of data introduced to the device, are the responsibility of the end user, and the IT officer in charge of that device in the field.

If an ESA is implemented, it won't be a mystery any more, as to what was introduced to a device that caused problems later. The sources of data in the environment all map back to the Environment Storage Area (presumably authorized by the military), so resource accountability will be in place if anything goes wrong. The single device single server idea, may even guarantees ongoing issues with Dos attacks are a thing of the past, because RD traffic on the IP6 layer (using DHCP6) is like finding a needle in a haystack!

A Dynamically Encrypted/Decrypted Native File System

As mentioned earlier, in order to ensure secure operation of mobile devices, the native file system must remain a functional part of the device. However, external connectivity to a military base is required for decryption.

The native file system is best kept intact, because that way app compatibility is ensured, so no recompiling, porting or changes to the already deployed application is necessary.

Further, the information on the VPS will contain a mirror of the device, essentially duplicate file paths and data, mirrored on the VPS, in the form of MDDE constructs in an Environment Storage Area (ESA), however their ciphers are different, and require authentication by the IDU, and the DPAK in the field!

During Dynamic Encryption, a tiny data transfer (i.e. a file path & cipher string) between server and device is all that is required for the encrypted file path, to be transmitted.   The MDDE construct (see FIG. 1 for MDDE [x1...xn] can be embedded in the native file system and the offsite virtual copy.

Said another way, while the device is in secure state operation mode, and the MDDE transform module is running, an "ordered set of MDDE constructs" are actively coupled to a VPS. All the while, the I/O requests being processed at the kernel layer, while resources are being read or written (encrypted and decrypted) using the dynamic ciphers, the device remains secure due to "Dynamic Encryption".

Dynamic Decryption decrypts "file paths" and the "file data" located on the remote devices local file system, or data resident in RAM, and is coupled to the remote device as a kernel level function.
As implemented, an example file name (or resource name) might be identified as FABCXC23, and the file data identified as KK332211, both encrypted forms on the local file system or I/O buffer. During runtime, and during a read operation, this data may only be decrypted using offsite (VPS Hosted Ciphers'. Similarly, at write time, these files are encrypted using off-site ciphers, and written to the VPS (ESA) SERVER. In either case, if a single cipher is broken, only one file in the file system is compromised. But, by the time the cipher has been solved, the device would likely have jumped back to non-secure mode.

Averting Security Risks using a Native File System

Clearly as each I/O operation occurs (on the mobile device) on the kernel layer, the I/O handler, for file resources, or storage requests, are encrypted/decrypted at runtime on the device, and queued to the VPS mirror, as well.

These I/O augmentations are directed at the native OS, so files and file names being encrypted and decrypted, at run time could run the gamut, (i.e. system files, data files, etc.).

Encryption is not limited to secure areas, but is provided on every file on the device, and ciphers for each file are variable, and depend upon external military resources built into an ESA.

Thus if a device is stolen, and somehow a hacker can emulate an dummy ESA, serving decryption keys, garnered from an intercept process, it is likely these ciphers would no longer remain solutions to the encryption used. This is because time would have passed when the original ciphers were intercepted, (presumably while building rouge ESA).

Averting Security Risks using a Native File System (continued)

In that case, it would thus be too late (or invalid at that point), due to the microkernel clock, having operated on subsequent resources, on the remote device, after the device jumped out of secure mode. Thus cipher keys would already have been applied to the native file system, so remote access, or other TCP/IP layer service access would no longer work, on the device, because internet access to those resources would be shutdown at that point. Not to mention the other two layers of security that would have to be solved. 1) Creating a dummy DPAK, and 2) creating a dummy ESA.

Communications Proliferation and Remote Device Operation Enhances Crypto

What is known is that encryption techniques based on a single cipher or cumulative ciphers, may be vulnerable if the keys are somehow acquired. But requiring external resources to provide ciphers, and requiring part of that cipher to be a function of time, representing clock ticks, coupled to a real I/O interrupt, at device "boot time, could result in an encryption system that cannot be subverted.

Not, because of the short durations between clock ticks, but because of the *relatively* long duration between packet intercept and ESA compromise.

Even if all iterations were presented to a compromised database having the ciphers, it would no longer be relevant, because it would have changed by then (the cipher only works on the device at the time the resource is needed), and this is assured, based on the communications between the 3 elements of the security schema, VPS, DEVICE and ESA, each having their own clocks, whose value changes at intervals which can likely, never be simulated by another system, or the system itself.

Additionally, using false information, embedded in the native file system, or the file itself, may provide other layers of security, in particular if the device fails while in operation.

Targeting these fundamental properties of a file, (the file system, the file data itself, and the file system path), one can assert an impervious source; from which to develop reverse ciphers (i.e. a decrypted source appended with additional false data, but encrypted using the same cipher).

Although the length of the file recorded in the original file system, might still offer clues to a hacker as to what the file contents are, then again the lengths may not. Since the PFS doesn't require that the folder structure maps the actual file length, to the file system, (potentially attainable in the MDDE construct, and only provided to the requestor thread), false data for these properties, may serve as a rue to the hacker, and could be substituted, and tracked by the PFS via the PADM. By using False Time Stamps, False File Lengths, False File Names and Types, etc., on the native file system, added security for this data in the file system (NTFS/Ext4), could be asserted, because secure access to the PFS would be required after the device was no longer operational.

In this context, the actual file entity properties (MDDE), would be returned, (from the secure ESA), but would only be relevant during the real-time operation of the remote device. Since one of the parameters of the solution is tied to the system clock on the remote device, it is unlikely a corresponding "on time" could ever be "produced", and even if it was, it would be too late, because the ESA would have changed by then.

Executing Instructions on a Secure Mobile Device, using an I/O Augmentation

While a RD is operating nominally, it will continually post an "OP" state (VS) to the VPS. That signature, is identified as a MK computation, and is comprised of three discreet parts.

1) An address summation algorithm (via the VPS)

2) A BIOID/CID indicator via Bluetooth (via PAD Device [lapel pin, for e.g.])

3) A sequence identifier (via the instruction clock on the local device)

These 3 discreet parts are combined into a single MK, and are used to authenticate kernel layer code, while processing I/O requests. This combined arrangement is identified as the Secure Nominal State (MK). The MK could be represented by an internal register, based on a sum of these values. Each of these individual parts of the MK is further described below.

The address summation algorithm is comprised of a tree of IP6 addresses pointing to a pool of authorized device files, located as a mirrored device image, on a secure military server, in an Environment Storage Area or ESA.

The BIOID/CID indicator is comprised of a secure IP6 address, located on the VPS. The BIOID/CID correlates to a static IP6 address, and is issued to device equipment in the field. This IP address points to server data which describes additional operational contingencies and parameters of these contingencies, related to the device operator.

The last part of the MK computation, should include a sequence identifier, and is required, in order to keep the authentication synchronized, between the VRD and the RD.

Since clock ticks (on the instruction counter), on the local device, have a *fixed* duration, and communications with a server has a variable duration (latency), a sequence identifier is necessary to account for this latency, thus averting delays in handing i/o requests, buffering packets.

It should be obvious to the reader that wireless speeds compared to clock speeds are disparate. Thus, one solution to this issue is to use a process where this latency, with respect to clock ticks is handled by allowing multiple cipher strings to apply to more than one particular file and filename pair.

For simplicity in communicating I have authored a PHP script for steps required to accomplish this. This script would need to be implemented in the language the kernel is written in on the remote device for example C++ for android based devices (under Linux).

Secure State Operation Mode

At boot time, all three category states, PAD/SERVER and OP, are set to nominal. While all three states are set to nominal, the device is said to be operating in a Secure State Operation Mode.

At any other time, once the device powers up, any one of the three category-states can change and can determine if the device continues to function or not in a secure state.. Also any of these states may be queried at any time within the scope of this subsystem, assuming conditions exist to support a query to the network or service that provides the state information. Otherwise, the state persists, in time, with values obtained from the initial state, or the last successful state query.

The device kernel (on the mobile device) operates continuously, only executing remote device instructions in secure mode, while these three states are set to nominal.

The various conditions triggering these states can be sent from a remote or a local scope, with the deterministic processing. Anything shared from SCADA, or contained in SCADA, could be shared, if the security clearance of the device operator was sufficient to make a query. And these states could be set by virtue of existing system currently embedded in SCADA, should it be required.

A non-secure operation mode would in general be a mode where no “outside” resources could be introduced to the device (for example internet browsing).

Dynamic Cipher Strings and the I/O Augmentation

For any device, the file system is the core of the device functionality, and also therein lay its’ vulnerability. Using Dynamic Cipher Strings is a complex proposition, but one that is not easily subverted.

This section attempts to describe the concept of Dynamic Ciphers, in the context of this proposal, and the implications administrating these ciphers, using off-site VPSs. Due to latency issues, and other criterion, an overview of mechanics required to develop Dynamic Ciphers is offered here.

On the “kernel” layer, of a mobile device, I propose integrating access to the file system, with augmented OS functions, pairing file names and file data with military grade encryption/decryption algorithms, while indirectly using the native file system on the device.

By intercepting I/O requests and receiving decrypt strings (for each resource or resource series), via a wireless connection, MDDE constructs (encrypted on the field device media) could be located and decrypted using the file-based dynamic cipher strings, at runtime.

Using the MDDE constructs, as an IP6 resource for these file/filename pairs, one can assume established communications protocols, and methods, such as using https for access to both local and remote communications, between RD and VRD, would be more than adequate, especially since their observance in transmissions, is not crucial to the ability of a hacker to do much damage.

Dynamic Cipher Strings and the I/O Augmentation (continued)

Although real-time encrypt/decrypt processing – coupled to local and off-site resources (while loading, writing, and reading data in the RD/VRD) is critical, observation of these packets, by a packet sniffer, reveals nothing about what is happening in either device, because of dynamic encryption. In order to address the time differential, between RD/VRD, and the kernel layer interrupt requests, buffering of File paths. These storage requirements, for Dynamic Encryption, would be part of a feasibility study, and I cannot be adequately estimated, as of this writing.

Alternatively, instead of using the I/O augmentation, resources, on the storage device, resources could be re-shuffled, or re- encrypted only at boot time, such that static file locations, and file names, on the native file system, could be cloaked, by using a sub- service-level boot process, initiated at boot time, instead of a clock dependent implementation. Using this type of implementation, would obviate the need for an MDDE database, and would render most of this proposal moot, but using Dynamic Encryption, alone makes for a robust, secure and compatible environment having many complex layers, not easily foiled, by hackers.

Ciphers Coupled to Dynamic Encryption

In order to insure the Dynamic Ciphers remain paired with the resource, I propose two techniques – having a rather subtle difference.

The first technique would involve using an encryption algorithm, having multiple cipher-solutions. That is more than one key could decrypt a given resource. The key would only change when the remote device would establish a subsequent connection to the server.

The second technique would involve using a single cipher-solution, and would apply to each individual file. By randomly assigning ciphers, local to the device, latency issues would not present a problem. However, ciphers would remain in memory until they could be transmitted, and only then would they be deleted.

Both techniques could be used on either the remote device or the VPS "mirror device".

In either case, the concept, I've identified as "Dynamic Encryption", can provide an extremely complex Encryption technique and is not easily subverted, because decryption details are stored off-site, and change in time. Also, each cipher solution is unique for each file, for each device and their device mirror.

Powerful Devices/Runtime Validation

As an added layer of security, the server could send *additional* unique runtime ciphers, coordinated with the remote server, depending on the resources available on both server and device.

If processing power, onboard storage sizes, etc. all increase in time (as we expect) for mobile devices, then buffering, and other time contingent limitations related to dynamic encryption and kernel level processes, will easily adapt as these predictable technological advances take place. Put simply the idea is scalable.

In general, this technology can adapt with minimal changes, as new storage sizes and connection speeds are deployed in remote devices - such as the recent micro SD upgrades on some devices.

Considering these possibilities, kernel modifications could be avoided, using an alternative design. But I believe by tying the data to the kernel I/O process, and quantifying the scope of file usage, on the execution layer, notable advances in security can be asserted.

Device Mirror/Shutdown/Restart

The offsite mirror of the device (the virtual device) would have unique cryptography applied to it. This virtual copy would be a functional object representing the actual device in the field (however with different crypto solutions).

Kernel level shutdown, of the device, could occur when VS not equal MK (device compromised), or an alternate less secure mode of operation could be initiated, if the device had been compromised. Therefore *detection* of device compromise is a requirement of the system for secure operations, especially in a secure military setting.

During reboot, or device malfunction, or if wireless connectivity or authentication was lost and then re- established, backtracking using process stack restart logs could assist in restarting an application whose sync data wasn't acknowledged on the USB mirror.

RD, PRD and Restore Options

As mentioned, the purpose of the PRD is to allow restoration of the mobile device if it is compromised. It should be noted however that a copy of the MDDE structures, in the ESA is available from two sources. These sources are via the VPS, over the internet, or via the PRD, over a common interface.

If mobile connectivity is functional, the optimum restore situation would be the method of choice. If not, the backup Hard Disk (the PRD) would be used, to restore the RD. If no PRD or no mobile connection available, then a final method would available this would be to cradle the device upon return to base, where a rebuild could be done, using the VPS (local link). The options to return the device to a secure state are listed below.

1. Via the VPS (bootable RD, via VPS, WAN link)
2. Via the PRD in the field (non-boot restore, emergency restore, no connectivity) + VPS
3. Via the VPS, (local link cradle)

The VPS will always contain an image copy of the remote device. The only difference between the RD and the VRD is that they are encrypted differently.

When there is no mobile connection available to the RD, the mobile device in this case would be restored to the point of when the device was originally deployed to the field, assuming the PRD had been cradled at base.

For efficiency, when the ESA on the local device is referenced, then cross-checking of resources will occur, and the appropriate cipher keys would be used while rebuilding the remote device.

Resources that comprise the RD environment, but have not changed, remain on the device. Resources that comprise the RD environment but have changed are updated using the VRD image.

Once a device goes off “state”, meaning it is no longer running in secure mode, than the VPS server will shut down. This is a security measure to avoid unauthorized equipment and users who have access to the RD and the PRD, from being able to restore their device. If the device is lost, then the finder would likely be unable to retrieve any information off of the RD or the PRD, because of the encrypted PFS.

You may recall the ciphers are not local to the device, and the VS/MK authentication must occur on the remote device, before a device can book into secure mode.

The restore process can be initiated by powering on the device, but all of the other security conditions must be in place in order for the rebuild to work.

Bandwidth Limitations and Handling Large File Updates to Field Devices

For updates to existing files (on the local device) and for handling changes to large-footprint files on both the local device and the remote VPS, I proposes handling these files similar to how CICS/INTERCOMM type Teleprocessing environments handle screen I/O, (i.e. only passing the changed portion), that is by alternatively treating the i/o buffer as screen data.

The technology, eluded to here, (and also described in my Patent Specification) may or may not port to this system, but nonetheless is offered here for consideration. The Teleprocessing environments are costly investments, but it is likely the Military has similar product, that could be integrated to handle these types of I/O.

My ideas with handling I/O updates to these types of files, can be developed from the ground up (or integrated from existing solutions) in an ESA, and these solutions are highly documented and addressed in various embodiments of “Automatic Periodic Centralized Backups of User Environments”.

Secured RAM Caveats

It should be clarified that, un-encrypted data is passed back and forth on the RD, between the applications (Data bus), and the OS, (and memory), but after that logical link is severed, (i.e. the file is properly closed); this data would likely, no longer exist. Also, when garbage collection processes are complete, the exposed device data should be purged from RAM.

For buffer overrun conditions, or other Internet based snooping activity on the TCP/IP layer, it is suggested to use a requester thread. Using the requester thread one can assert the file accessing resources on the device is an authenticated file.

Due to the native file system being represented as a PFS embedded with MDDE constructs, and the original associations having pointed to the existing file structure, application’s requesting the I/O services, can continue to do their thing. That’s a good thing, because data changes in closed, I/O buffers, having no interfaces to speak of, on the application layer, effectively removes any chance of the data ever being intercepted, examined or otherwise extracted during secure device operations.

Leveraging our Nations Technology for the Military

The military has the “exclusivity” factor, which is best suited for this technology, and the ESA provides a sensible and pragmatic approach, that uses our nation’s resources and ingenuity, collectively, to build a strong secure model around proven infrastructure.

An ESA can provide an extremely secure exclusive feed of data, with many potential benefits, including an ability to trace and intercept resources accessed from remote devices.

By working with device vendors, and using Small Business Innovative Research, work on both fronts can come together in ways never before considered.

The “melting pot” of technology is here. If we work together we can build a secure military network of RD’s around existing technology, so military can focus on their jobs using familiar tools and interfaces.

Feasibility Considerations

- ✓ Assemble teams of technology leaders, to analyze feasibility, in the associated disciplines.
- ✓ Learn about the SCADA system (where possible) for overlaps in technology, and how it relates to features and definitions of the system and processes described in this document.
- ✓ Determine the feasibility to the OS kernel modifications.
- ✓ Determine the feasibility of real-time crypt/decrypt, of file-system layer entities using a PFS.
- ✓ Define the Pseudo File System, and its' component parts.
- ✓ Research a Personal Authentication Device (PAD), considering alternative media upon which to build these into, for example using "flexible plastics".
- ✓ Create a secure military data center containing test services, (e.g. ESA accessibility parameters), for the RD authentication requirements, and populate these servers with MDDE Constructs, correlating to the field devices.
- ✓ Generate and distribute Match Keys (MK), and Validation Signature (VS) data, used for authentication purposes, in a test environment, and determine feasibility of testing these entities, at runtime in a device kernel.
- ✓ Study the impact of time dependent contingencies, relevant to RTE's. For example packets sent to/from mobile devices, (mobile connections and/or Bluetooth).
- ✓ Consider implementing the RTE, at another layer of the OS, other than the kernel.
- ✓ Consider the impact of creating a special file system driver, as part of the Linux kernel, or using an existing specialized driver for custom file systems, already developed for the Linux kernel.
- ✓ Study the Impact of the Pseudo File System on Remote Device Operation, as implemented.
- ✓ Determine the minimum hardware requirements, for RD's for this proposal.
- ✓ Determine the best secure software environment for PADM development and deployment.
- ✓ Determine best means to install the PADM on the device.
- ✓ Analyze various security vulnerabilities to PADM on the BT, RD, and the PRD.
- ✓ Evaluate Modes of Operation (Secure and Non-Secure)
- ✓ Determine the VPS deployment requirements.
- ✓ Determine PAD, DPAK and Dynamic Encryption Techniques, and search for holes or vulnerabilities in handling Ciphers.

Feasibility Considerations (continued)

- ✓ Plug holes, or consider alternate higher level modifications to the design, to address any weaknesses discovered.
- ✓ Consider alternate special override modes of operation and determine methods of implementing counter intelligence, on the remote device, in the event the device has been compromised.
- ✓ Define counter intelligence options, as in the form of rouge data, being intentionally planted or invoked on the remote device, by the system. Such definitions could include, altering segments of the files system or replacing the entire file system with alternate file system, or even using low-level utilities intended to scrub or initialize media on RD.
- ✓ Consider how using alternate sources of, BIOID and CID information (likely requisitioned from existing structures in the military), and unknown to the author at this time, would integrate with this security system, and the impact it has on functionality of this system.
- ✓ Administrative Interfaces, for this technology should be developed, with emphasis on the layers of security and how they are administered and distributed to officers and the IDU – that is the chain of command as referenced.
- ✓ Consider the use of superior systems and/or alternate systems and methods, which can be integrated with ideas expressed herein.

Patent Status, Office Actions and Intellectual Property

The Status at the USPTO of the Patent Specification referenced in is proposal, is “Abandon” due in part to a lack of funding for additional legal counsel, and my desire to keep the broad claims intact, and to avoid compromising the essence of the application.

Follow-up amendments to office actions, with the PTO, did not serve to clarify the original concepts, I Envisioned, and are therefore suggested to be ignored.

I recommend using the original Patent Specification, submitted to the USPTO in 2007, as the resource to draw supplemental information from, regarding the Environment Storage Area and the MDDE constructs referenced in this proposal. Some of the original Patent Drawings, have been corrected, since the original filing, and are included in the Resources section, on my website.

The Patent Application, referenced in this proposal, is used solely to communicate the high level concepts, and in no way assigns or relinquishes my claim to this work or any derivative work, resulting from the application.

Summary

I have carefully considered the current issues in mobile technology, and how these issues affect good design. After having isolated those areas, I focused on a cost-effective solution, which targets R&D in the most critical area of security, which is the data on the TCP/IP layer. Other systems running on the device, such as JAVA, are best left alone, and the focus placed on the kernel I/O layer that these functions access. Thus the I/O augmentation is critical, both to the full understanding of this proposal and to the likelihood of it’s’ adoption.

By identifying the weakest links concerning security in mobile devices, I feel I have defined a practical solution that will remain secure, and useable well into the future. Hopefully I have shown you why using a subset of well-tested, current technology (networks, hardware and interfaces, and indexing these entities in an ESA) can actually be used for future tech, having the same base parameters - those parameters being Specifically broad and specifically narrow. By incorporating the MDDE, and its broad object definition of mobile devices and device entities, I believe it is well worth the effort to invest in this schema, and optimally to build on it.

In my opinion this system is both cost effective, and powerful, and could be implemented with a minimum amount of programming, compared with other top down designs.

Also, since the administrative responsibilities are distributed at the IDU layer, the proper channels of communication, between IDU and IT security personnel, regarding secure device usage, can be assured. The military is a perfect fit for this technology, since privacy rights are waived by IDU's.

The IP6 specification and offsite authentication requirements, coupled with the I/O process, can help to ensure resources on a remote device are not compromised, and/or used in an unauthorized setting. Therefore the MDDE’s, as described in my patent application and the ESA are the high/low level points to analyze when considering this proposal. A file is only relevant to the device, in that a system can read it. Thus the MDDE takes this simple premise and builds on it, from access point, through usage, to storage and finally to destruction, protecting it from prying eyes!

Acknowledgements

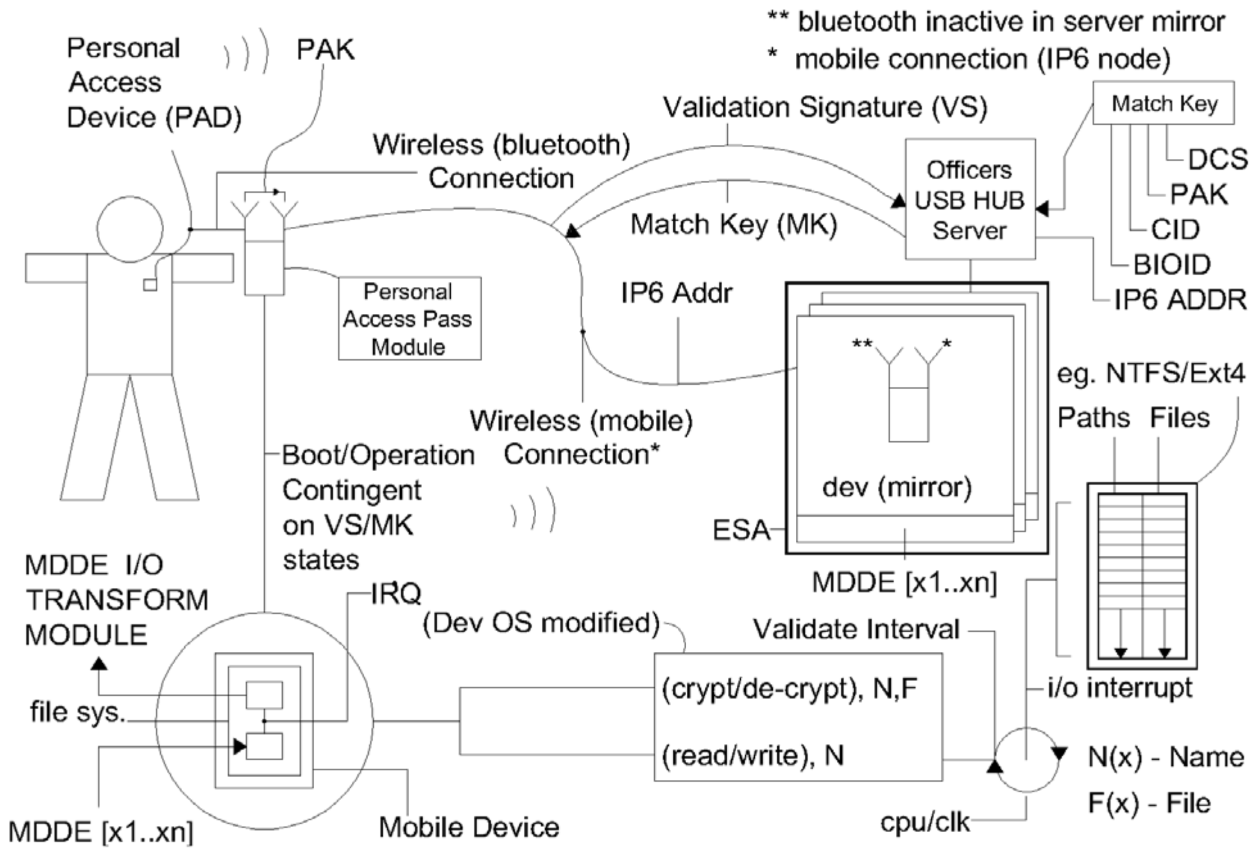
I would like to acknowledge my Attorney, David S. Atkinson, formerly of Suiter Swantz, Omaha Nebraska, for the excellent work he did when writing the patent application “Automatic Periodic Centralized Backup of User Environments”. Although no claims were allowed during examination of the patent, I believe the value of the specification is relevant to those who take time to assimilate the broad explanations, definitions, and relationships cast therein.

I also would like to thank various individuals within the federal, state and private sectors, who have helped and encouraged me to continue documenting my ultimate goal of developing a “Global Backup System and Method”.

In particular, I would like to thank the Service Core of Retired Executives (S.C.O.R.E) and Dr. Jim Karam PhD, the Nebraska Business Development Center (NBDC) and Veronica Doga and Marisol Uribe Rodriguez, the Nebraska Procurement Technical Assistance Program (PTAC) and Jason Bosquet, the Small Business Association (SBA) and Lisa Tedesco, for helping me to understand the nature of research in a Small Business Setting.

Also, I acknowledge the Greenwood Consulting Group and Jim Greenwood, and Dr. Joe Iseman PhD. for their kind assistance, in helping me with past proposal preparation for the Small Business Innovative Research Program (SBIR).

Finally, I would like to thank Dr. Murali Nair, at the National Science Foundation (NSF), for offering pragmatic encouragement and a second chance, to write on this topic.



SUPPLEMENTAL ILLUSTRATION 1

This illustration depicts authentication interactions between high level components in the security subsystem, described in this proposal. In particular, this drawing shows the IP6 communication, and Bluetooth communication, occurring between the mobile or "field" device and the VPS (or HUB Server as depicted here), and the Personal Access Module.

This illustration is intended to assist the reader *while* reading this document. Please refer to the Patent Drawings for additional visual aids.

Notes

I. The algorithm used to communicate and compute the VS, uses the MDDE construct, in the form of a "tree" of IP6 addresses, each branch represented also, as an MDDE construct. For each buffer of an I/O block, to the file system(s), on the field device, the interrupt would access an encrypted path and unencrypted file name, describing the encrypted resource data, located on the "native" file system. For each file or file-set, unique cipher keys are assigned, and accessible on the remote server, and the RD/VRD would be dynamically modified at runtime. Since part of the calculation, for device authenticity may require other local operational parameters (such as GPS/Temp, etc.), then in theory it can be said that authentication formulations for RD usage, could be described in the form of the questions; Who is using the device, What are the conditions regarding the device, When is the device authorized to be used, Where is the device being used and Why is the device being used?

II. Wireless jamming, communications issues, and other measures preventing good links between RD and base, are very important, although not required for this system to offer help. An optimal implementation would likely occur while communications are nominal, available and advisable. However, by using the PRD, as described in this proposal, one can assert a "Plan B", in the event of communications disruption
In the field, and at least offer a restored environment to the point of dispatch, of the RD. Getting the device online, and bootable, with the applications intact (as of dispatch), would still be a viable alternative to a "bricked" device.

III. Since the schema operates the remote device in a virtual environment (at the same time the actual device is dispatched and subsequently operating), data remains secure (in the event the RD or the VRD is compromised). Due to discreet pairs of cipher strings (the VRD and the RD), coupled with the RD, RTE's, and RD hardware, an extremely logical approach to security can be asserted.

IV. Since the secure layer is placed in an ESA (which is a "mobile cloud"), and has control to these resources in and out of this area, presumably on the application layer, then obtaining the originating thread, and historical thread of these resources could be asserted upon build of the ESA.

V. Further research on intercepting I/O, from the memory bus, needs to be determined, and is dependent on the device manufacturer. However, hacker techniques such as applications snooping RAM, (i.e. buffer overrun conditions) for example, might be averted by querying the MDDE, for an identifier defined as a "requestor thread". In particular, each resource requestor must be authenticated to read resources sourced from, and created in, certain environments.

VI. The peer-to-peer idea, using DHCP (IP6) in a large mega-subset of IP6 addresses, helps to hide the traffic on the web, and may help to lessen Denial of Service attacks, which require the IP address of the target site to be known. Also, since the server and device is at the single IP address layer, incoming unauthorized traffic can be mitigated, because attacks on the system are on a global (WAN) scope, and each IP address allocated represents a closed system. Also, if compromise is detected, then that IP is shutdown immediately, without affecting other devices operating in the field.