Executive Summary

The architecture of the Intel Itanium processor was specifically designed not only to provide unprecedented computational advantages, but also to enable development of systems with substantially greater security than is possible with other existing and evolving processor architectures. The security advantages offered by the Intel Itanium processor architecture are real, and are urgently needed to strengthen cyber infrastructure.

Secure64 Software Corporation has developed server software products that use this secure architecture, running on Hewlett Packard’s Intel Itanium processor-based Integrity servers. This paper explains, in minimally technical terms:

1. The characteristics of a secure system that can drastically reduce the likelihood of attacks that compromise system integrity with unauthorized code
2. The security capabilities provided by the Intel Itanium processor architecture that are not found in other mainstream processor architectures
3. How these security capabilities can be used to develop systems that prevent known forms of malware injection and rootkits

For those interested in more technical processor architecture details, the appendix explains the evolution and principal characteristics of the processor architectures that pre-dated the Intel Itanium processor architecture and elaborates the technical advances and advantages of the Intel Itanium processor architecture over these earlier processor architectures.

Secure System Requirements

To be trustworthy, an operating system must guard critical system properties against external attacks. The sections that follow describe internal system security properties of such a highly-secure system, through unique features of the Intel Itanium processor architecture that enable these properties.

Executable Code Integrity: No System/Library Code Infections or Injected Malware in Data Areas

It is vital to allow only authenticated, non-compromised system or application executable codes to run on the system. Malware must actually execute to inflict the damage it was designed to wreak. If it cannot be injected in executable form, it cannot do its dirty deeds. When booting a system, all executable system and library codes can be authenticated by cryptographic means. Once authenticated, Intel Itanium processor hardware can position, isolate, and protect executable codes in memory. This is done by a combination of completely independent memory mapping of data and executable codes, and access privileges that ensure all such codes are executable, but neither addressable, readable, nor writable.
by any other non-kernel application or system/library functions.

Once a system is booted, only an authenticated kernel module can load other authenticated, executable application or system/library codes. Once loaded, these codes are isolated and protected. An Intel Itanium processor can make it extremely unlikely for information read into a system as data ever to be executed as instructions. This closes a major attack path for injecting and then executing malware.

Executable Scripts Integrity: No Script Attack Paths for Malware

The system must allow only authenticated, non-compromised scripts to be interpreted. This is accomplished in several steps:

- **Executable code integrity methods** also protect the executable code of the interpreter itself.
- **Means can be designed to authenticate scripts** and, in all cases, to prevent unauthorized or other than read-only access to the script content.
- **Access to user and system data and files** while interpreting a script can be limited only to that for which the user invoking the script has access permissions.
- **Access to system functions** must be restricted to those for which the user and interpreter code have permission.

The executable code of the script interpreter can be protected (see Executable Code Integrity).

Setting read-only access to its containing memory can, in part, provide the integrity of the script itself. The Intel Itanium processor provides an additional degree of protection by permitting the containing memory to be tagged with a protection key (PK). This can also ensure that only the correct interpreter can access the script, and doubly ensure that the script can only be read. This can eliminate any malware access to script contents.

The Intel Itanium processor’s protection keys can also ensure that all memory access by a running interpreter is restricted to those memory areas for which such access is authorized.

Data Access Integrity: No Data Cross-Contamination by Buggy Codes

It is important to ensure that the only access to data is by executable software authorized to make such access. Software components executing at the same privilege level, in the same memory address space, must be prevented from accessing data belonging to different software components. This is especially important for system software, where such cross-contamination can result in system failures that can be extremely difficult, or even impossible, to diagnose.

This also is accomplished by Intel Itanium processor’s PKs, a unique feature that does not exist in other mainstream microprocessor architectures. A protection key contains a 24-bit ID and three control bits that can independently disable read, write, or execute permissions, respectively. If a memory area is tagged by a PK, access to that memory area, by executable code running at any hardware privilege level, is possible only if the processor hardware state contains a matching PK ID in one of 16 protection key registers (PKRs). This capability permits organizing a large memory area into subsections, called compartments, and limiting access to each compartment to executing codes with a PKR state that contains a matching PK ID and has no PK bit set to disable the type of access.

Control State Integrity: No Stack Code Injection or Control Hijacking

The stack is a memory area used to save information about the currently running program function before it transfers control to another function. Long ago, attackers recognized that if they could alter the contents of the stack, they could force the system to execute code of their choosing. This led to an entire class of highly publicized attacks called buffer overflow attacks. These
How Intel® Itanium® Processor Enables Superior System Security

buffer overflow techniques to inject malware into a running system.

Authenticated System Calls: No System Help for Malware

An operating system provides a rich set of callable functions whose capabilities are also highly coveted by malware. The ability to call these functions greatly increases the weapons at the disposal of a malicious attacker. This is how, for example, root kits and other persistent malware can be planted onto the system disk. It is also how malicious contaminations can be made to system and library executable codes, both in memory and on disk. A secure system must ensure that calls to system functions are honored and executed only when made by authenticated, authorized software.

Another benefit of the Intel Itanium processor architecture is the capability for fine-grained control of which executable codes are allowed to call specific operating system or utility library functions. As explained previously, executable codes are set to execute only. When so mapped, the contents of these executable codes can be made completely inaccessible to other running codes. This is the first step in governing which executable codes can call specific system functions. The contents of the calling executable codes are, in effect, secret.

Step two comes from the fact that Intel Itanium processor instructions themselves are also able to directly load arbitrary 64-bit values into registers without having to read these values from data memory. These values can be set into code at load time to establish fresh 64-bit key values that cannot then be accessed or observed by other running codes.
When an operating system is booted, fresh random key values can be generated, or read from the non-deterministic random number generator of a trusted platform module (TPM). When authenticated system or application executable codes are loaded, the fresh key values can then be inserted into instructions of appropriate system and library functions and, where specified by authenticated load modules, into instructions within authenticated executable codes authorized to have such values. This is step three.

Each system or library function for which such a key is required checks for a matching key argument value when it is called. If the caller’s argument key matches the key embedded in the system or library function, the function completes its task and returns. If it does not match, the system or library function does not complete its function and raises an appropriate system alarm. This is step four.

In a system using such authenticated calls, malware, even if it could be injected, would no longer be able simply to use system calls to inflict damage, or to have the system long-term copy itself, or other malware, into memory or disk locations, where they later could contribute to an advanced persistent threat (APT) attack.

**Conclusion**

These Intel Itanium processor capabilities enabled Secure64 to develop a micro operating system and DNSSEC server product line that is highly resistant to known forms of malware, integrates built-in defenses for denial of service (DoS) and other network attacks, and provided the first fully automated DNSSEC Signer server.

Development of this DNSSEC Signer product was partially funded by the U.S. Department of Homeland Security (DHS), which realized that Secure64’s Intel Itanium processor-based server platform was the first that was secure enough to safely manage the cryptography, custody, and protection of the many cryptographic keys required to automate DNSSEC signing. The micro operating system architecture itself was reviewed by a commercial security firm that spent several days on-site at Secure64 with full access to Secure64 personnel and documentation and concluded that “the traditional methods used by malware to gain system access and obtain the privileges necessary to install themselves for continued infection do not appear viable” and also by a government-led team at Sandia National Laboratories before being recommended to critical government agencies.

The principles developed by Secure64 are generalizable to more complex operating systems and platforms. Intel Itanium processors are thus capable of hosting a wide range of physical and virtualized platforms with the unprecedented security capabilities currently available only in Secure64’s product line.

Secure64 believes that building such capabilities into basic system components is essential to secure critical infrastructure cyber security. We recently submitted the following comments to NIST’s “Framework to Improve Critical Infrastructure Cybersecurity.”
Unless critical cyber infrastructure is built from system components that integrate essential security and defense capabilities from the ground up, no body of policies, best practices, standards, and procedures will be able to protect it sufficiently. The vicious cycle of vulnerability-exploit-patch will persist.

Experience deploying massively complex COTS systems lacking such integrated security and defense capabilities, and then attempting to secure them with additional protective software and external bodyguard systems such as firewalls, intrusion detectors, etc., as well as operational standards, practices, and polices, has:

- **Shown itself unable** to stem the still-growing tide of bot formation, compromised systems, and successful cyber attacks. Systems continue to lose more and more cyber attack battles. It is clear that new thinking and approaches are needed.
- **Produced no solid evidence** that refinement of this current approach can be expected to succeed in the long run.
- **Led to increasingly complex and expensive** bodyguard systems, which are themselves often ineffective, vulnerable, and capable of being hijacked and used in attacks.
- **Confirmed Albert Einstein’s observation** that “Insanity is doing the same thing over and over and expecting different results.”

For more information about Secure64’s line of “secure from the ground up” server software based on Intel Itanium processors, visit [www.secure64.com](http://www.secure64.com) or contact Secure64 at (303) 242-5890.
Appendix: CISC, RISC, VLIW, EPIC

Architecture Evolution

The foundation for the Intel Itanium processor architecture was developed in the early 1990s in Hewlett Packard’s HP Labs. A project was initiated to understand the strengths and weaknesses of the principal processor architectures embodied in the widely deployed computer system families at that time, and to determine whether it was feasible to define a new class of processor architecture that offered clear advances in functionality, price, performance, and system security over the then-current state of the art.

HP Labs’ conclusion was that such an advanced processor architecture was feasible. HP adopted this new class of architecture, shared the concepts with Intel, and proposed to work together to develop and offer processor products embodying these advances as industry-standard components available to all hardware system manufacturers. This led to the HP/Intel relationship that gave birth to Intel’s evolving family of processors based on the Intel Itanium processor architecture.

The basic price and performance advantages of the Intel Itanium processor result from the fact that active processing cores can be physically smaller and simpler, and that more operations can be executed during each processor cycle. These were the issues thoroughly debated by HP Labs and the HP product divisions, resulting in the conclusion that the advantages of this new class of processor architecture were compelling.

Actual chips subsequently showed that, using the same silicon technology, the active processing core of an Intel Itanium processor could fit in only half the silicon area needed by the active processing core of an earlier architecture. For processors of the same size, this meant that an Intel Itanium processor could provide much larger on-chip cache memories, resulting in higher performance due to faster access to the most frequently used data.

The Intel Itanium processor offers multiple functionality advantages: It:

- **Can execute** many more operations during each processor cycle
- **Reduces** software control complexity and overhead.
- **Introduced** innovative capabilities for managing a memory hierarchy, issuing long latency memory operations earlier than possible in other architectures, eliminating high entropy branch instructions, and for highly efficient software pipelining—all to reduce overall execution times.
- **Provided** a much larger set of computational resources for software to use, and hardware functions to manage, these resources without extra software complexity.
- **Has arithmetic units** to provide advanced capabilities for both floating point and big integer computations.

More would be needed to describe all of the Intel Itanium processor’s functional advantages. For this appendix, we will discuss only some of the key considerations that led to Intel Itanium processor’s functionality advantages.

For more comprehensive details about processor architects and evolving processor architectures during the decades preceding the turn of the century, see Professor Mark Smotherman’s outstanding Computer Architecture History site: [http://www.cs.clemson.edu/~mark/](http://www.cs.clemson.edu/~mark/).

Briefly, the HP Labs project leading to the Intel Itanium processor architecture began only a few years after the introduction in the 1980s of a new class of processor architectures called Reduced Instruction Set Computing (RISC). At the time the HP Labs effort began, conventional thinking was that out-of-order, superscalar RISC systems would be the best evolutionary direction for HP’s processor products. In fact, both superscalar and out-of-order superscalar PA-RISC systems were actually developed and sold by HP during the 1990s.

Consider just a few of the most widely deployed computer architectures of the 1960s, 1970s, and 1980s that set much of the background for RISC architectures. IBM, in 1964, introduced the first family of compatible systems, spanning a range of prices and performance levels, with the IBM System/360*. This was followed by IBM’s System/370* mainframes in the 1970s. Digital had weighed in with the VAX* family of compatible systems in 1978, and HP with the HP 3000* family in 1972. As microprocessors made their appearance in the 1970s, Intel introduced processors based upon the X86 architecture and Motorola introduced the Motorola 68000* chip family. IBM selected Intel’s X86 architecture for its highly influential IBM PC* product line, introduced in August 1981.

All of these architectures were of a class that became known as Complex Instruction Set Computing (CISC) architectures. They were characterized by:

- **A plethora of instructions** that required multiple cycles to complete
- **An internal micro engine** to control the multiple cycles
- **Internal memory** to hold the instructions needed to control the micro engine, called microcode
- **Processor instructions** that could operate with operands directly read from or written to memory

Dr. John Cocke of IBM had realized in the early 1970s that most of the instructions actually executed by most programs were just the simpler ones that:
ment program, christened Advanced Computer

During the 1960s, IBM had launched a develop-

with the basic RISC pipeline.

to be executed in parallel, this was not possible
be accomplished permitted multiple instructions
be accomplished permitted multiple instructions

If the computational work to

function of less complex instructions. This was
performance over CISC processors when built

HP (PA-RISC*), DEC (ALPHA*), and IBM

sors developed by MIPS (MIPS*), Sun (SPARC*),
sors, whose development was led by Dr. Denny

geometric, and in the IBM RS/6000 and Power
processors, all of which were introduced after

RISC architecture proponents advocated
RISC SS and OoOSS as the most promising

future architecture direction. (Find additional

information about the ACS effort on Professor

Smootherman's website).

The HP Labs effort also benefitted heavily from
earlier work done at Cydrome, led by Dr. Bob
Rau and Mike Schiansker, and the work done at
Multiflow, led by Dr. Josh Fisher. Both of these

efforts had developed processor architectures and
system products christened Very Long
Instruction Word (VLIW). (You can also find addi-
tional information about these efforts on
Professor Smootherman's website.) Both Bob Rau
and Josh Fisher were members of HP Labs, later
HP Fellows. They and Mike Schiansker were key

contributors to the development of the Intel

Itanium processor architecture. Their previous
experience included deep studies of instruction
sets designed to issue multiple instructions to
the hardware on each cycle and the scaling of
such architectures. They understood the inner
details of predicated execution, software
pipelining, scaling do's and don'ts, speculative
execution, and parallel pipelines. They heavily
influenced how such capabilities were provided
by the Intel Itanium processor architecture,
because the evolved concepts derived from their
early VLIW work were demonstrably powerful.

Also contributing significantly to the HP Labs

The advantages of the processor architecture
direction defined by the HP Labs work, which
became christened as Explicitly Parallel
Instruction Computing (EPIC), can be further
appreciated by the following considerations, first
articulated by Dr. Rajiv Gupta, the HP Labs effort
architecture manager. A compiler can see the

• Could be directly implemented in hardware
• Did not require an internal micro engine
• Could more efficiently operate on operands
contained in hardware registers
• Need access memory only with load and
store instructions

When Professor Don Knuth published his com-
pelling analysis of actual compiled Fortran* pro-
grams in 1971, John Cocke's insights were fur-
ther substantiated and, in 1975, he initiated
work that led IBM Yorktown Research to build
the IBM 801, the world's first RISC research
prototype. This effort was led by Dr. Joel
Birnbaum, later director of HP Labs, and staffed
by an exceptionally talented and distinguished
team from IBM's Yorktown Research Labs.
Details about this outstanding team can be
found on Professor Smootherman's website.

John Cocke was the visionary genius who under-
stood that RISC architecture would enable very
powerful high-performance processors to be
possible on single silicon chips. He was truly the
father of RISC architectures.

As a result, many computer processors based on
RISC architectures made their debuts during the
1980s and early 1990s. These included proces-
sors developed by MIPS (MIPS*), Sun (SPARC*),
HP (PA-RISC*), DEC (ALPHA*), and IBM
(RS/6000*, Power*). These systems were eventu-
ally acknowledged to offer superior price and
performance over CISC processors when built
using the same silicon technologies.

RISC architectures enabled simply pipelined exe-
cution of less complex instructions. This was
their strength and, at the same time, their fun-
damental limitation. If the computational work to
be accomplished permitted multiple instructions
to be executed in parallel, this was not possible
with the basic RISC pipeline.

During the 1960s, IBM had launched a develop-
ment program, christened Advanced Computer
System (ACS), led by IBM's Dr. Herb Schorr and
the company's vice president, Jack ("Blackjack")
Bertram. This effort invented the design of a
processor capable of executing multiple simple
instructions in parallel, not necessarily in the
architectured one-at-a-time instruction sequence
defined by a processor architecture. Such execu-
tion became known as Super Scalar (SS) and Out
of Order Super Scalar (OoOSS). This technology
was offered in the HP PA-RISC Snakes* proces-
sors, whose development was led by Dr. Denny
Georg, and in the IBM RS/6000 and Power
processors, all of which were introduced after
1990. RISC architecture proponents advocated
RISC SS and OoOSS as the most promising
future architecture direction. (Find additional
information about the ACS effort on Professor
Smootherman's website).

The HP Labs effort also benefitted heavily from
earlier work done at Cydrome, led by Dr. Bob
Rau and Mike Schiansker, and the work done at
Multiflow, led by Dr. Josh Fisher. Both of these
efforts had developed processor architectures and
system products christened Very Long
Instruction Word (VLIW). (You can also find addi-
tional information about these efforts on
Professor Smootherman's website.) Both Bob Rau
and Josh Fisher were members of HP Labs, later
HP Fellows. They and Mike Schiansker were key
contributors to the development of the Intel
Itanium processor architecture. Their previous
experience included deep studies of instruction
sets designed to issue multiple instructions to
the hardware on each cycle and the scaling of
such architectures. They understood the inner
details of predicated execution, software
pipelining, scaling do's and don'ts, speculative
execution, and parallel pipelines. They heavily
influenced how such capabilities were provided
by the Intel Itanium processor architecture,
because the evolved concepts derived from their
early VLIW work were demonstrably powerful.

Also contributing significantly to the HP Labs
effort were University of Chicago Professor
Clemens C. J. Roothaan and Dr. Alan Karp.

Professor Roothaan, a member of the
International Academy of Quantum Molecular
Science, is the father of computational methods
for a-priori, large-scale scientific computations of
atomic and molecular wave functions, including
the groundbreaking Hartree-Fock-Roothaan
method that bears his name. Dr. Karp is an astro-
physicist who joined HP Labs after 15 years of
work on large-scale scientific computations at
IBM. Both Clemens and Alan are fluent, hands-on,
assembly-level programmers whose knowledge
and experience enabled them to wisely guide
the HP Labs work with respect to the architec-
tural capabilities needed for large-scale scientific
computations, and their highly parallel and
pipelined implementations. The resulting Intel
Itanium processor computational resources were
not arbitrary; they were based upon the detailed
analyses and codes of these superb scientists.

The highly pipelined vector math function library
for Intel Itanium processor, developed by HP and
CERN, set new records for speed and accuracy.

The HP Labs effort thoroughly investigated SS,
OoOSS, and VLIW architectures. It was able to
prove that the SS and OoOSS processor architec-
ture directions were not as effective as that
eventually adopted by the Intel Itanium proces-
sor family. First, the actual level of parallelism
that could be achieved by RISC OoOSS turned
out to be quite limited. Second, the silicon area
required for determining which RISC instructions
to execute on each cycle turned out to be huge.
The first HP OoOSS PA-RISC, for example,
required as much silicon area for just this
instruction scheduling function as was required
for the entire previous-generation PA-RISC
processor chip.

The advantages of the processor architecture
direction defined by the HP Labs work, which
became christened as Explicitly Parallel
Instruction Computing (EPIC), can be further
appreciated by the following considerations, first
articulated by Dr. Rajiv Gupta, the HP Labs effort
architecture manager. A compiler can see the
source code for an entire set of functions. From this, it can discern the opportunities for parallelism, and for other optimizations such as overlapping memory latencies by speculative early execution of memory loads, eliminating branch instructions by predicated execution, software pipelining using the RSE renaming functions, etc.

For OoOSS processors, the compiler then must generate a sequence of instructions to correctly perform the computation if those instructions were to be executed one after the other in successive processor cycles, as specified by the RISC processor architecture. At execution time, the OoOSS hardware then must, on every cycle, examine a limited window of the generated instructions and determine which of them can be executed out of order and/or in parallel. This task is extremely complex, requiring long pipelines and coordinating the results of thousands of comparisons. It is not surprising that this requires a huge silicon area.

For architectures such as the Intel Itanium processor’s, the compiler can organize and generate groups of executable instructions that are to be executed in parallel. Large silicon areas are no longer required to determine what to execute in parallel. The Intel Itanium processor architecture added additional capabilities to overlap memory access latencies to:

- **Simplify** software pipelining
- **Minimize** the need for branch instructions by predicated execution
- **Concurrently drive** multiple pipelined memory, integer, conditional and unconditional branch, and floating-point functional components

RISC architectures typically offered 32 architecturally visible general-purpose registers. The Intel Itanium processor offers four times that many, and provides hardware that automatically renames, saves, and restores physical registers as needed by programs. This eliminates the pervasive CISC and RISC need to frequently execute single or multiple cycle housekeeping instructions to save and restore the contents of registers.

Finally, but not least importantly, capabilities for enabling construction of highly secure systems were an explicit objective for the HP Labs work. This objective drove, among other capabilities:

- **The evolution** of early security capabilities offered in PA-RISC.
- **The evolved** virtual memory mapping and access privileges.
- **Fully separating memory mapping** for executable codes and data.
- **Fine-grained memory compartmentalization** and access controls.
- **RSE register management** for critical control state.
- **Higher access privilege level** RSE save/restore areas.
- **Software pipelining** integrated with RSE register renaming.
- **Preserving virtual addressing** while handling processor exceptions.
- **Full 64-bit immediate constants** in executable instructions.
- **A repertoire** of synchronizing instructions.

2Ibid.

Copyright © 2013 Intel Corporation. All rights reserved.

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL’S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS’ FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined”. Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.