Loss-Tolerant and Secure Embedded Computing via Inscrutable Instruction-Set Architectures (I^2SA)

Technical Report GIT-CC-04-12

Vincent J. Mooney III, Krishna V. Palem and Richard B. Wunderlich

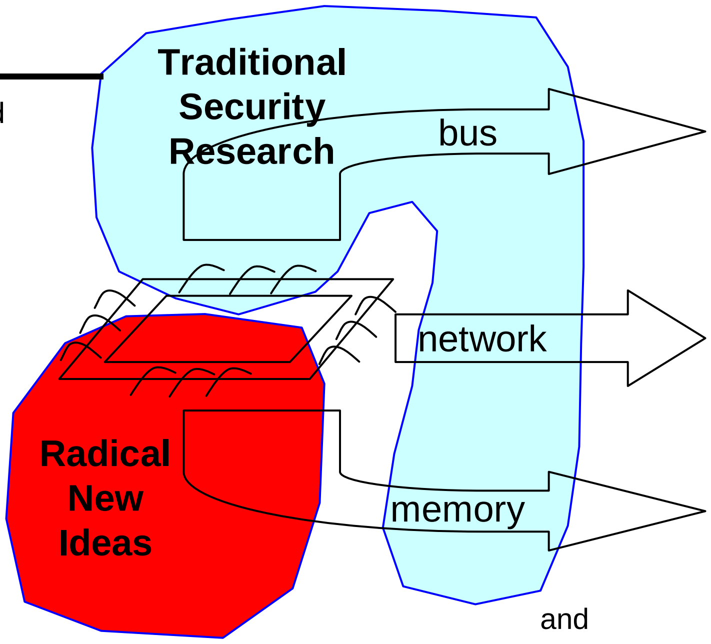
Center for Research on Embedded Systems and Technology

School of Electrical and Computer Engineering

Georgia Institute of Technology, Atlanta, Georgia, U.S.A.

17 November 2004

1. INNOVATIVE CLAIMS



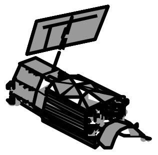
As a contrast to prior work and as shown in the red area in the picture to the right, we propose an Instruction-Set Architecture (ISA) and associated processor hardware design which implements an *Inscrutable* ISA (I^2SA). An I^2SA operates on non-standard numbers such that the data are inscrutable or nearly impossible to fully recover with possession only of the hardware chips implementing an I^2SA programmed with some application(s). In short, we examine secure computing where a microprocessor reads, writes and executes operations ideally *in an inscrutable domain*. The goal is to provide methods and implementations for computation where data never leave the inscrutable domain in which they reside; the intended effect is twofold. The first intended effect is that any transmissions between computing media using our approach would be unbreakable in any reasonable amount of time; i.e., intercepted instructions and/or data would be meaningless to the unauthorized interceptor. The second intended effect is that, as a result of embedded computing platforms realized using inscrutable computing elements, any loss of equipment utilizing such computing hardware would not be very meaningful to the recoverer: we refer to this as *loss tolerance*. In other words, if the computing hardware – and associated instructions, data and microchips – were to fall into hostile hands, no data could be fully understood (all bits uncovered) in a reasonable amount of time (e.g., less than one year). Furthermore, we also investigate degrees of inscrutability where portions of the data are inscrutable and other portions are not inscrutable (i.e., not easy to understand).

Shown on the previous page is a juxtaposition of research proposed here versus "traditional" security research regarding processors and ISA. Traditional security processor and ISA research to date has focused on providing protections and encryption support for bus, network and memory contents and traffic. However, in all prior research, encrypted data must be unencrypted to plain, standard text (i.e., typical Boolean numbers implemented with straightforward bit patterns such as two's complement for number representation) prior to execution with the ISA.

We realize such loss-tolerant computing using an Instruction-Set Architecture (ISA) we call an *Inscrutable* ISA (I^{2}SA). To our knowledge, we are the first such group to ever investigate this approach.

2. Example Scenario and Domain Impact

Consider the following embedded computing scenario as an example. Wireless transmission hardware shown to the right is placed in the theatre. This hardware receives, transmits and generates data.



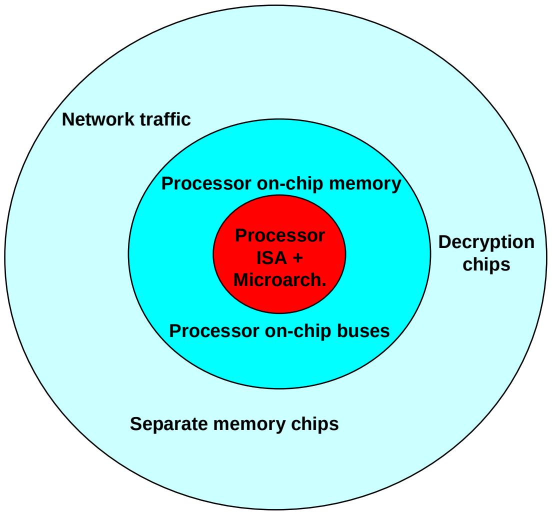
To further clarify the situation, assume that the wireless transmission hardware is unmanned and falls into malicious hands *with all microchips fully functional*. The result is that engineers in hostile territory end up with all chips in hand.

Now consider this scenario with current state-of-the-art encrypted buses, memory and network traffic. Let us assume the best of published state-of-the-art; thus, decryption is not performed on a separate coprocessor chip but instead is inside the single microprocessor chip. Furthermore, various hardware protections have been added to make sections of memory readable only by certain software processes with proper keys. Unfortunately, known techniques (such as electrical signal monitoring, power analysis, differential fault analysis and reverse engineering) can potentially read the hardware logic, thereby uncovering how the internal hardware carries out decryption, including any keys stored in internal register locations. Given this information, the memory can be decrypted and all bits recovered.

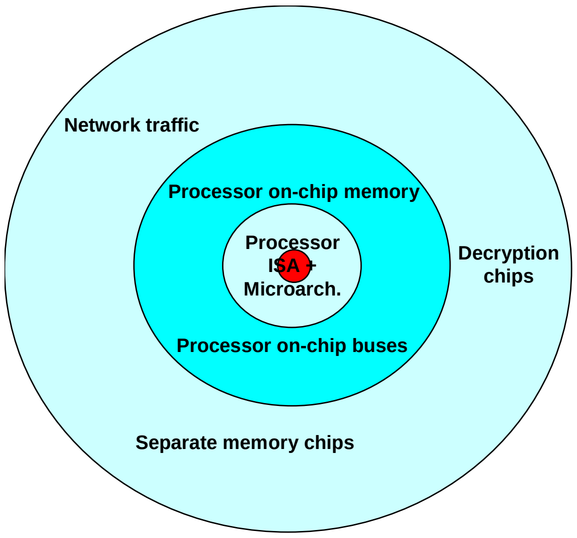
As an alternative, take all of the above plus one additional fact: the microprocessor uses an I^{2}SA. If the hardware were to fall into malicious hands, the hardware logic could potentially be reverse engineered. However, the raw data and instructions operate in an *inscrutable* domain. Even with expensive large-scale equipment for reverse engineering available, say, to a nation-state, nonetheless the hostile engineering staff have an additional complicating factor: all computation is carried out with an I^{2}SA. In short, even in this scenario, the possibility that data could be recovered in any reasonable amount of time (say, less than a year) is essentially zero.

3. Technical Approach

To the right is shown increasing levels of use of encrypted data. At the outermost level are network traffic and external, separate memory chips. Also, in the outermost level are separate decryption chips which take incoming encrypted data and output unencrypted data. Recent work has maintained data in encrypted format in parts of on-chip memory and on-chip buses (on the microprocessor chip itself). This is shown to the right in the blue-green middle circle. At the innermost circle in red to the right, the processor ISA is shown the processor ISA with the associated microarchitecture. Currently, all prior work known to the authors implements the ISA in an unencrypted microarchitecture, namely, everything in the red circle is non-inscrutable. As a result, none of the processor hardware implementing the assembly instructions operates on anything other than unencrypted, raw data.



To approximate an inscrutable ISA we shrink the red inner circle shown above. For example, we could consider operating on encrypted instructions except for branch instructions. For a branch instruction alone, we would unencrypt in order to execute the branch. The result is shown on the right where the red (unencrypted) domain of operation is much smaller.



Homomorphisms

In one interpretation, the problem of realizing a fully inscrutable ISA (or I^{2}SA) is intimately related to that of *homomorphic* encryption schemes. While such schemes are sufficient for our needs, it is not at all evident that they are necessary. Much of our effort along the foundational domain in the context of realizing I^{2}SA has been devoted to innovating “weaker” notions of realizations than those required either by complete homomorphisms on the one hand, or by the need for a completely secure system following the rigorous characterizations based on *semantic security*. Clearly, an auxiliary goal will be to understand the feasibility of realizing such systems where the notion of weakness, relative to a fixed ISA, will be quantified. Finally, based on these foundations, an analysis of the I^{2}SA frameworks innovated can reconcile the *quality of security* afforded by them against the theoretically feasible limits.

4. Target Environment

To demonstrate feasibility, we port the I^{2}SA to a well known compiler infrastructure, Trimaran. Trimaran is in use worldwide with over 30 sites active (e.g., publishing results and dissertations using Trimaran). Furthermore, the Center for Research on Embedded Systems and Technology (CREST) at Georgia Tech has an active and vibrant research program in customization of embedded systems. Research performed at CREST is leveraged in this research, especially for power modeling, analysis and optimization for design of the I^{2}SA embedded processor.

We evaluate the research along three dimensions: speed, energy and increase in security. As a baseline case, we consider a microprocessor with the same attributes (chip technology, sizes of memories, etc.) running the same applications. Thus, reported simulation results comparing metrics for instructions per second, instructions per Joule, and increase in security should always have a baseline case so that clear and precise quantitative statements can be made.

5. Conclusion

In conclusion, we propose an Instruction-Set Architecture (ISA) and associated processor hardware design which implements an *Inscrutable* ISA (I^{2}SA). An I^{2}SA operates on non-standard numbers such that the data are inscrutable or nearly impossible to fully recover with possession only of the hardware chips implementing an I^{2}SA programmed with some application(s). In short, we examine secure computing where a microprocessor reads, writes and executes operations ideally *in an inscrutable domain*. The goal is to provide methods and implementations for computation where data never leave the inscrutable domain in which they reside. Finally, we also investigate degrees of inscrutability where portions of the data are inscrutable and other portions are not inscrutable.