Security Processing for High End Embedded System with Cryptographic Algorithms

M.Shankar[1], Dr.M.Sridar[2], Dr.M.Rajani[3]

[1]Associate professor,
Department of Electrical and Electronics Engineering,
Kuppam Engineering College, Kuppam, Andhra Pradesh (India)
Magaprajin@gmail.com

[2]Director International Relations
Bharath University,
Chennai, 600073, Tamilnadu, India,
deanrdinter@bharathuniv.ac.in

[3]Director of R & D,
Bharath University,
Chennai, 600073, Tamilnadu, India,
deanrd@bharathuniv.ac.in

Abstract

This paper is intended to introduce embedded system designers and design tool developers to the challenges involved in designing secure embedded systems. The challenges unique to embedded systems require new approaches to security covering all aspects of embedded system design from architecture to implementation. Security processing, which refers to the computations that must be performed in a system for the purpose of security, can easily overwhelm the computational capabilities of processors in both low- and high-end embedded systems. The paper also briefs on the security enforced in a device by the use of proprietary security technology and also discusses the security measures taken during the production of the device. We also survey solution techniques to address these challenges, drawing from both current practice and emerging research, and identify open research problems that will require innovations in embedded system architecture and design methodologies**.**

Keywords: *Security, Performance, Design, Reliability, Algorithms, Verification, architecture, hardware design, processing requirements, security protocols, cryptographic algorithms, encryption.*

1. INTRODUCTION

Today, security in one form or another is a requirement for an increasing number of embedded systems, ranging

from low-end systems such as PDAs, wireless handsets, networked sensors, and smart cards, to high-end systems such as routers, gateways, firewalls, storage servers, and web servers. Technological advances that have spurred the development of these electronic systems have also ushered in seemingly parallel trends in the sophistication of security attacks. Security has been the subject of intensive research in the context of general-purpose computing and communications systems. However, security is often misconstrued by embedded system designers as the addition of features, such as specific cryptographic algorithms and security protocols, to the system. In reality, it is a new dimension that designers should consider throughout the design process, along with other metrics such as cost, performance, and power. The secure data not only requires protection during data transfer but also while handling the data at the end user devices [1]. Vulnerability at the end user device, like easy access to the secret keys that are used to encrypt or decrypt the data, can easily turn down the entire security measures. The protocol involved for the secure transmission of either of the above mentioned contents through a public network uses more or less the same techniques but the handling of the user restricted data at the user's end involves much more care as the content is protected from the user itself [2]. This paper will introduce the embedded system designer to the importance of embedded system security, review evolving trends and standards, and illustrate how the security requirements translate into system design challenges.

2. SECURITY REQUIREMENTS OF AN EMBEDDED SYSTEM

The processing capabilities of many embedded systems are easily overwhelmed by the computational demands of security processing, leading to undesirable tradeoffs between security and cost, or security and performance. Battery-driven systems and small form-factor devices such as PDAs, cell phones and networked sensors often operate under stringent resource constraints (limited battery, storage and computation capacities). These

constraints only worsen when the device is subject to the demands of security. Battery-driven systems and small form-factor devices such as PDAs, cell phones, and networked sensors are often severely resource constrained. It is challenging to implement security in the face of limited battery capacities, limited memory, and so on. An ever increasing range of attack techniques for breaking security, such as software, physical, and side-channel attacks, require that the system be secure even when it can be logically or physically accessed by malicious entities. Countermeasures to these attacks need to be built in during system design. The data in a public network passes through a number of un trusted intermediate points. Therefore the secure data must be scrambled in such a way that the data will be useless or unintelligible for anyone who is having unauthorized access to the secure data. This can be achieved with the help of cryptographic methods such as Encryption/Decryption [3,4], Key Agreement, Digital Signatures and Digital Certificates. The use of these cryptographic methods in an embedded system to achieve data security is explained in the following sections.

2.1 Security defined in a system is to:

• Identify Threat

• Set Targets

• Assess Risks

• Devise Countermeasures (people, processes,

measures and procedures)

• Assure Countermeasures Remain Effective

A security protocol is a sequence of steps, followed by two or more parties, such that certain security objectives are satisfied. A security objective is formulated to either counter the threats or to ensure that interactions between legitimate parties satisfy some requirements. Following are the common security objectives which need to be satisfied by security protocols:

1. Confidentiality - Information is not disclosed to unauthorized entities.

2. Integrity - Any unauthorized manipulation of data can be detected.

3. Authentication - An unauthorized entity should not be able to pose as a legitimate entity.

2.2 Challenges in Secured Embedded Systems

Behind these visible applications there may also be several layers of back-end systems which must prevent fraud by distributors, network operators and other participants in the value chain. A good example is given by the prepayment electricity meters used to sell electric power to students in halls of residence, in the third world, and to poor customers to cause the system to ignore certain events. Imagine anti-aircraft radar that uses an embedded real-time operating system [6, 7]. Within the system are several Flash ROM chips. A virus is installed into one of these chips and it has trusted access to the entire bus. The virus has only one purpose to cause the radar to ignore certain types of radar signatures. Viruses have long since been detected" in the wild" that write themselves into the motherboard BIOS memory. In the late 90s, the so-called F00F bug was able to crash a laptop completely. Although the CIH (of Chernobyl) virus was widely popularized in the media, virus code that used the BIOS was published long before the release of CIH. Common, underlying challenge has to do with the central role of domain experts in embedded system design. It is common for embedded system development teams to be relatively small, and staffed more with domain experts than computing experts. This is often appropriate, because expert domain knowledge is crucial to success [5]. However, small teams and companies that are concerned mostly with an application domain rather than computer technology often don't have access to expertise in dependability.

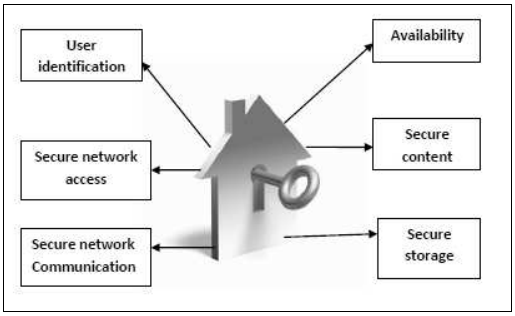


Fig.1. Common security requirements of embedded systems

User identification refers to the process of validating users before allowing them to use the system *Secure*

Network access provides a network connection or service access only if the device is authorized.

Secure communications functions include authenticating communicating peers, ensuring confidentiality and integrity of communicated data, preventing repudiation of a communication transaction, and protecting the identity of communicating entities.

Secure storage mandates confidentiality and integrity of sensitive information stored in the system.

Content security enforces the usage restrictions of the digital content stored or accessed by the system.

Availability ensures that the system can perform its intended function and service legitimate users at all times, without being disrupted by denial-of service attacks.

Symmetric ciphers require the sender and receiver to use the same secret key to encrypt and decrypt data. They are typically used for ensuring confidentiality of data, and can be chosen from two classes—*block* and *stream* ciphers [8]. Block ciphers operate on similar-sized blocks of plaintext (original data) and cipher text (encrypted data). Examples of block ciphers include DES, 3DES, AES, and so on.

Asymmetric ciphers (also called public-key algorithms), on the other hand, use a private (secret) key for decryption, and a related public (no secret) key for encryption or verification. They are typically used in security protocols for verifying certificates that identify communicating entities, generating and verifying digital signatures, and for exchanging symmetric cipher keys[9,10]. These algorithms rely on the use of computationally intensive mathematical functions, such as modular exponentiation, for encryption and decryption.

Hashing algorithms such as MD5 and SHA provide ways of mapping messages (with or without a key) into a fixed-length value, thereby providing "signatures" for messages

3. ATTACKS ON EMBEDDED SYSTEMS AND COUNTER MEASURES

Various attacks on electronic and computing systems have shown that hackers rarely take on the theoretical strength of well-designed cryptographic algorithms. Instead, they rely on exploiting security vulnerabilities in the software and hardware components of the implementation. In this section, we show that unless security is considered throughout the design cycle, embedded system implementation vulnerabilities can easily be exploited to bypass or weaken functional security measures. Technological advances that have spurred the development of these electronic systems have also ushered in seemingly parallel trends in the sophistication of attacks they face. An ever increasing range of attack techniques for breaking security, such as software, physical, and side-channel attacks, require that the system be secure even when it can be logically or physically accessed by malicious entities. Countermeasures to these attacks need to be built in during system design. If Root CA certificate can be modified, then the attacker can make the device to accept any certificate by substituting a fake root CA certificate and thus defeating the purpose certificate and secured communication. It is therefore also important that the security in the device is such that the data such as Root CA Certificates in the device is not subjected to unauthorized modification. First, there has been a good deal of work on verifying crypto protocols, which are typically sets of 3-5 transactions exchanged by two principals. But in many real systems, these techniques must be extended to the dozens or even hundreds of transactions supported by the actual cryptographic service provider (whether smartcard, crypto processor, or software library) [11]. Finally, a tamper-resistant device can be considered as just a high-quality implementation of an object that can only be invoked using its social methods, and whose internal variables remain inaccessible. Given that the object-oriented programming model is becoming popular, there may be more general lessons to be learned for robust programming.

3.1 Known-key attack

The upshot was that most bank security modules had a transaction to generate a key share and print out its clear value on an attached security printer. It also returned this value to the calling program, encrypted under a master key (which we'll call KM) which was kept in the tamper-resistant hardware:
Host −*!* VSM : \Generate Key Share"
VSM −*!* printer: *KMTi*
VSM −*!* Host: *fKMTigKM*
The VSM had another transaction which combined two of the shares to produce a terminal key: While the above attack was found by inspection, the following one was found by formal methods { by writing a program that mapped the possible key and data transformations between deferent key types, computing the transitive closure under these, and scanning the composite operations for undesirable properties

3.2 Passive side channel attacks

Hiding: Break relation between processed value and power consumption
Masking / Blinding: Break relation between algorithmic value and processed value

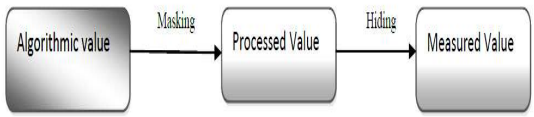


Fig.2. Channel attacks

3.3 A `two-time type' attack

While the above attack was found by inspection, the following one was found by formal methods { by writing a program that mapped the possible key and data transformations between different key types, computing the transitive closure under these, and scanning the composite operations for undesirable properties. It turns out that reusing a key type can be as dangerous as reusing a key in a one-time cryptosystem. Just as the Soviet re-use of key material during World War 2 led to what Bob Morris beautifully describes as the `two-time pad', so the re-use of the terminal master key type for PIN generation keys makes it into a `two-time type' that opens up another neat attack.

4. MEET-IN-THE-MIDDLE ATTACKS

The attack itself is straightforward. An identical test pattern is encrypted under each key, and the results recorded. The same test pattern is encrypted under each trial key and the result is then compared against all versions of the encrypted test pattern. Checking each key will now take slightly longer, but there will be many less to check12, 13, and 14]. It is much more efficient to perform a single encryption and compare the result against many different possibilities, than it is to perform an encryption for each comparison. Using a hash table, the comparison stage can be made almost free.

Organizational attacks (e.g., social engineering) can be prevented by well-thought security processes, secure infrastructures and organizational security policies

Logical attacks (e.g., cryptographic weaknesses or weak APIs) can be prevented by a secure well-thought security system design and adequate security protocols

Software attacks (e.g., weak OS mechanisms or malware) can be prevented by reliable software security mechanisms (e.g., secure unit, secure RTEs) and the application of hardware security mechanisms that protect & enforce security of software mechanisms

Hardware attacks (e.g., security artefacts manipulations/read-out, physical locks, side-channels etc.) can be prevented by hardware tamper-protection measures

5. MECHANISMS FOR EMBEDDED SECURITY

Trust SW security

Hardware security

System security

Organizational security

Table 1: relationship between security services and mechanisms

Security Service	Supporting Security Mechanisms
Peer entity authentication	Encipherment, digital signature authentication exchange
Data origin authentication	Encipherment, digital signature
Access control	Access control
Confidentiality	Encipherment, routing control
Traffic flow confidentiality	Encipherment, traffic padding, routing control
Data integrity	Encipherment, digital signature, data integrity
No repudiation	Digital signature, data integrity, notarization
Availability	Data integrity, authentication exchange

Secure Access Protocols

Security protocols are built using cryptographic algorithms to realize a combination of four security objectives confidentiality, integrity, authentication and non-repudiation, while availability is made possible through the use of access control security mechanisms The level of security provided is dependent upon many things such as the cryptographic methods used, the access to the transmitted data, algorithm key lengths, server and client implementations and most importantly, the human factor. Security protocols provide ways of ensuring secure communication channels to and from the embedded system [15]. To achieve data security, cryptographic methods such as Encryption/Decryption, Key Agreement, Digital Signatures and Digital Certificates are being used

6. DATA ENCRYPTION

This paper offers two contributions. First, a survey investigating the computational requirements for e a number of common cryptographic algorithms and embedded architectures is presented. The objective of this work is to cover a wide class of commonly used encryption algorithms and to determine the impact of embedded architectures on their performance. This will help designers predict a system's performance for cryptographic tasks. Second, methods to derive the computational overhead of embedded architectures in general for encryption algorithms are developed. This allows one to project computational limitations and determine the threshold of feasible encryption schemes under a set of the constraints for an embedded architecture. But when message authentication is required in addition to encryption, hash or block ciphers, such as RC5, have the advantage of providing support for both authentication and encryption

6.1 Public-key Key Agreement Algorithm

In Public Key Agreement (PKA) algorithms two interlocutors *A* and *B* produce a secret shared key (SSK) by exchanging public information and combining it with private one. Such cryptographic algorithms are called *asymmetric* because the private information's possessed by *A* and *B* are different and not shared [16, 17]. In the present talk a new method to construct PKA algorithms is discussed in which this residual form of symmetry is eliminated, hence the name: *strongly asymmetric PKA Algorithms* The splitting of the public information into multiple public keys implies levels of: Security Variety of concrete realizations which cannot be found in the standard PKA algorithms. The construction of these algorithms does not depend on sophisticated mathematical structures (e.g. groups associated to elliptic curves or complex theorems of number theory).

This implies a drastic decrease in implementation complexity and increase in velocity

6.2 Digital Signature

APPLICATIONS SUCH AS banking, stock trading, and the sale and purchase of merchandise are increasingly emphasizing electronic transactions to minimize operational costs and provide enhanced services. This has led to phenomenal increases in the amounts of electronic documents that are generated, processed, and stored in computers and transmitted over networks. This electronic information handled in these applications is valuable and sensitive and must be protected against tampering by malicious third parties (who are neither the senders nor the recipients of the information) [18]. Sometimes, there is a need to prevent the information or items related to it (such as date/time it was created, sent, and received) from being tampered with by the sender (originator) and/or the recipient. Traditionally, paper documents are validated and certified by written signatures, which work fairly well as a means of providing authenticity. For electronic documents, a similar mechanism is necessary. Digital signatures, which are nothing but a string of ones and zeroes generated by using a digital signature algorithm, serve the purpose of validation and authentication of electronic documents. Validation refers to the process of certifying the contents of the document, while authentication refers to the process of certifying the sender of the document. A simple generic scheme for creating and verifying a digital signature is shown in Figs. 1 and 2, respectively. A hash function is applied to the message that yields a fixed-size message digest. The signature function uses the message digest and the sender's private key to generate the digital signature. A very simple form of the digital signature is obtained by encrypting the message digest using the sender's private key. The message and the signature can now be sent to the recipient. The message is unencrypted and can be read by anyone. However, the signature ensures authenticity of the sender (something similar to a circular sent by a proper authority to be read by many people, with the signature attesting to the authenticity of the message). At the receiver, the inverse signature function is applied to the digital signature to recover the original message digest. The received message is subjected to the same hash function to which the original message was subjected.

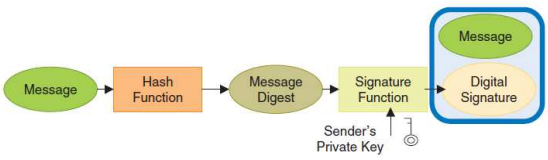


Fig. 3 creating a digital signature

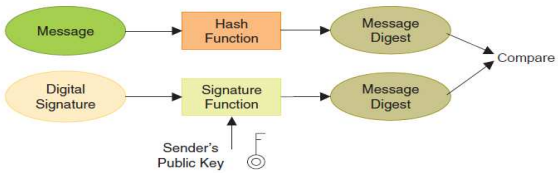


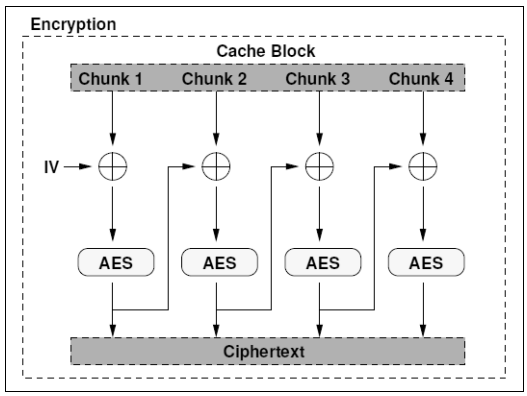
Fig. 4 verifying a digital signature

6.3 Digital Certificate

An attachment to an electronic message used for security purposes. The most common use of a digital certificate is to verify that a user sending a message is who he or she claims to be, and to provide the receiver with the means to encode a reply. An individual wishing to send an encrypted message applies for a digital certificate from a *Certificate Authority (CA)*. The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information. The CA makes its own public key readily available through print publicity or perhaps on the Internet. The recipient of an encrypted message uses the CA's public key to decode the digital certificate attached to the message, verifies it as issued by the CA and then obtains the sender's public key and identification information held within the certificate. With this information, the recipient can send an encrypted reply.

7. SECURE PROCESSING ARCHITECTURES

A secure computing architecture provides a solid foundation for secure software applications. Hardware structures built with secure computing in mind can add significantly to the performance of secure computation. All domains of computer security share a common set of primitives like encryption and hashing and the performance of secure computing solutions are greatly enhanced if these primitives can be implemented in hardware instead of software.



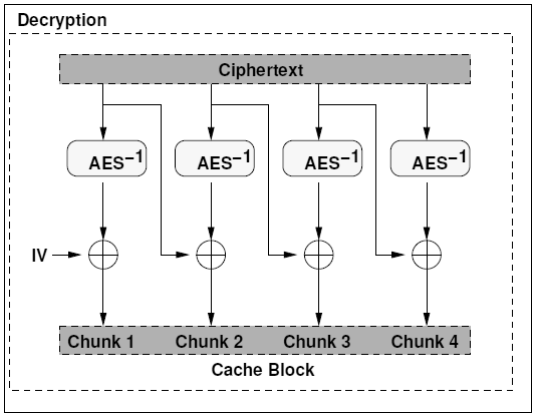


Fig 5. Encryption and Decryption

Secure SoC

In the last four decades of the 20th century, many information and communication technologies have been developed and also introduced in several social infrastructures, which are supporting our daily lives. Since the information technologies have progressed very rapidly, the basic structure of each social infrastructure, which was mostly designed in the 19th or the beginning of 20th centuries with few information technologies, should be redesigned under the assumption of the existence of the advanced information technologies. Based on the high performance SoCs (System-on-a-Chip) connected by wideband networks, we can design next generation of social systems, which are directly related with quality of our society including individual rights and national security SoC technology is now one of the most fundamental information technologies for the social infrastructure as well as network technology and embedded software technology. Since the rapid progress of these information technologies causes the drastic reduction of time and space of information transfer, processing and storage, new scheme of social infrastructure are redesigned under the assumption of the utilization of these information technologies.

Secure ROM

One method for storing the device secret keys securely in the persistent storage of a device is to encrypt the secret keys before storing. Thus even if anyone managed to get the data out of the persistent storage he/she will never be able to understand the secret keys. To encrypt any data generally two things are required, an encryption algorithm and a key for encryption. If any well-known algorithm like AES is used for encryption of the secret keys, then the strength of the encryption is only as strong as the secrecy of the key that used for the encryption. Thus the same problem faced for the storage of the secret keys is faced again for the storage of the key that is used for encrypting the secret keys. This problem is repeated unless an encryption algorithm is used that is known only to the device manufacturer. If the device proprietary algorithm is used for the encryption and storage of the secret keys, the security of the secret keys are only as strong as the secrecy of the algorithm

Table.2.Secure Boot-Loader and Code Signing

Security levels for boot loader

	Security Features					Ease of Management
	Software			Hardware		
	CRC ECC	Hash	Signature	Write Protected Bootloader	TPM	
Normal Boot	O	-	-	-	-	Easy, but no protection
Secure Boot (by digest)		O		Root of Trust (Reference Value)		Bad
Secure Boot (by signature)		O	O	Root of Trust (Signer's public key)		Good + Easy to update OS image without modifying Bootloader
Trusted Boot		O		Root of Trust	Root of Trust (Secure Storage)	Good (for connected device) + Device Authentication + Integrity Protection + Integrity Report

Any attempt on overriding the firmware components of the device thus must be turned down. The presence of secure Boot loader can ensure this. On start-up before loading the firmware code, the Secure Boot loader checks whether the firmware is genuine or not and prevents the device from booting up if the device firmware is modified or replaced.

8. CONCLUSION

In this paper we analyzed the various ways in which the attacks can be performed on the embedded systems. Any security function implemented in an embedded system must be considered in both hardware and software, at all design abstraction levels, in communications between components, and in the manufacturing phase. The good news is that unlike the problem of providing security in cyberspace (where the scope is very large), securing the application-limited world of embedded systems is more likely to succeed in the near term. However, the constrained resources of embedded devices pose significant new challenges to achieving the desired levels of security. We believe that a combination of advances in architectures and design methodologies would enable us to scale the next frontier of embedded system design, wherein, embedded systems will be "secure" to the extent required by the application and the environment.

9. REFERENCES

[1] *Counterpane Internet Security, nc.http://www.counterpane.c* http://www.counterpane.com.

[2] *Paynews - Mobile Commerce Statistics*. http://www.epaynews.com/statistics/mcommstats.html.

[3] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1998.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code inC*. John Wiley and Sons, 1996

[5] Anoop MS, Elliptic Curve Cryptography - An implementation guide, May 2007, Available at http://msitbox.blogspot.com/2008/03/elliptic-curve- cryptography.html

[6] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M.Kirkup, A. Menezes, "PGP in Constrained Wireless Devices", in Proceedings of the 9th USENIX Security Symposium, Denver Colorado, pp. 247-261, Aug. 2000.

[7] D. Carman, P. Kruus, B. Matt, "Constraints and approaches for distributed sensor network security", NAI Labs technical report#00-010, Sept 2000,

[8] W. Dai, "Crypto++ 4.0 Benchmarks", http://www.eskimo.com/~weidai/benchmarks.html

[9] CH Meyer, SM Matyas, `Cryptography: A New Dimension in Computer Data Security', Wiley, 1982

[10] SM Matyas, `Key Handling with Control Vectors', IBM Systems Journal v 30 no 2, 1991

[11] SM Matyas, AV Le, DG Abraham, `A Key Management Scheme Based on Control Vectors', IBM Systems Journal v 30 no 2, 1991, pp 175{191

[12] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, D. Culler, "SPINS: Security Protocols for Sensor Networks", Proc. 7th Ann. Intl. Conf. Mobile Computing and Networking (MobiCom 2001), pp. 189-199, 2001.

[13] "LAN MAN Standards of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specification IEEE Standaard 802.11, 1997 Edition," 1997.

[14] O. S. Elkeelany, M. M. Matalgah, K. P. Sheikh, M. Thaker, G. Chaudhry, D. Medhi, and J. Qaddour, "Performance Analysis of IPSec Protocol: Encryption and Authentication", IEEE Communications Conference (ICC 2002), pp. 1164- 1168, 2002

[15] National Institute of Standards and Technology. *Advanced Encryption Standard*. FIPS Publication 197. NTIS, Nov. 2001.

[16] P. R. Schaumont, H. Kuo, and I. M. Verbauwhede. Unlocking the design secrets of a 2.29 gb/s Rijndael processor. In *DesignAutomation Conference 2002*, June 2002.

[17] W. Shi, H.-H. S. Lee, M. Ghosh, C. Lu, and A. Boldyreva. High efficiency counter mode security architecture via prediction and pre computation. In *ISCA '05: Proceedings of the32nd Annual International Symposium on Computer Architecture*, pages 14.24, Washington, DC, USA, 2005. IEEE Computer Society.

[18] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Ef_cient memory integrity veri_cation and encryption for secure processors. In *MICRO 36: Proceedings of the 36th annual IEEE/ACMInternational Symposium on Microarchitecture*, page 339, Washington, DC, USA, 2003. IEEE Computer Society.

[19] G. E. Suh, C.W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. *SIGARCH Comput. Archit. News*, 33(2):25.36, 2005.

[20] J. Yang, Y. Zhang, and L. Gao. Fast secure processor for inhibiting software piracy and tampering. In *MICRO 36: Proceedingsof the 36th annual IEEE/ACM International Symposium on Microarchitecture*, page 351, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Y. Zhang, L. Gao, J. Yang, X. Zhang, and R. Gupta. SENSS:Security enhancement to symmetric shared memory multiprocessors. In *HPCA11*. IEEE Computer Society, 2005.