On the Feasibility of Very Low Complexity Trust Modules Using PKPS Synergies

Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS 39762.
Email: ramkumar@cse.msstate.edu, Ph: 662 325 8435

***Abstract*— For many evolving application scenarios like ubiquitous and autonomic computing systems, trustworthy computing solutions are essential. However the fact that the autonomic elements that may take part in such networks may be 1) severely resource constrained, and that 2) the sheer scale of such devices may also place constraints on their *cost*, calls for inexpensive, low complexity (but nevertheless trustworthy) computing modules, or secure co-processors (ScP). We propose two synergistic strategies for the realization of very low complexity, inexpensive ScPs. The first is a simple security policy, decrypt only when necessary (DOWN). The second is the utilization of untrusted external resources to improve the security of very low complexity ScPs. We point out some very desirable properties of probabilistic key pre-distribution schemes (PKPS) that can take good advantage of the DOWN policy and simultaneously make use of external resources, to render the problem of their susceptibility to collusions irrelevant.**

I. Introduction

In evolving application scenarios based on the paradigms of ubiquitous and autonomic computing [1], devices are expected to co-operate with each other to realize synergistic benefits. The task entrusted to an autonomic element may have a very specific and limited scope. Some autonomic elements may be entrusted with the task of providing accurate temperature, time or position information. Some elements may be entrusted with the task of collecting information from numerous other elements, and relaying them to other elements that may need such information. While the task performed by each element may be trivial, the information supplied by such element may have high significance. For example, a single misbehaving sensor could possibly result in lage inaccuracies in weather forecasts. Thus it is important that such autonomic elements are highly trusted. The minimum requirement is to ensure that at least a part of the tasks performed by any autonomic element is carried out *inside* a trusted boundary - such as a secure co-processor (ScP).

Apart from their use in the military, trusted computing solutions employing cryptographic processors [2] have also seen widespread use in a variety of civilian application scenarios. Low end processors have been used in ATMs since the early eighties, and in smartcards and set top boxes like cable and satellite TV receivers since the early nineties. Higher end hardware security modules have found extensive use in securing high end servers that cannot be afforded proper physical protection. While the higher end ScPs are generally considered trustworthy, the extent of assurances offered by low end ScPs are questionable. For the feasibility of evolving ubiquitous and autonomic computing scenarios [1], ScPs bound to such autonomic elements have to be inexpensive, low complexity, and simultaneously capable of providing an "acceptable level" of assurances. While the simultaneous requirements of "inexpensiveness" and "trustworthiness" may seem mutually exclusive, we argue that there are indeed synergistic strategies to realize inexpensive ScPs.

The contributions of this paper are two-fold. The first is a simple security policy, decrypt only when necessary (DOWN), which is an algorithmic approach to protect secrets in tamper-responsive ScPs. The DOWN policy can significantly lower the costs of ScPs by reducing the complexity of circuitry required for countermeasures against tampering attempts. The second (contribution) is the discussion of factors contributing to a synergistic relationship between the DOWN policy and a class of key distribution schemes referred to as probabilistic key predistribution schemes (PKPS).

PKPSs, which have recently attracted much interest for securing highly resource constrained sensor networks, are seen as fragile schemes due to their susceptibility to collusions. A very desirable property of PKPSs is that it can permit very low complexity ScPs to take advantage of inexpensive *external* resources to improve their resistance to collusion. In the highly networked world of the future, while autonomic elements may themselves be resource constrained, it is reasonable to expect that they have access to (possibly untrusted) external storage and computational resources, over (possibly) untrusted networks. Furthermore, PKPSs lend themselves very well to efficient implementations of the DOWN policy. Synergistically, effective implementation of the DOWN policy can dramatically improve the security offered by PKPSs. In conjunction with the DOWN policy, and their ability to make use of external (untrusted or less trusted) resources, the collusion resistance of PKPSs can be increased to such extents so as to render their "fragility" *irrelevant* in practice.

In Section II we provide a brief overview of issues related to protecting secrets in ScPs. Section III discusses the DOWN policy. In Section IV we discuss the synergistic features of PKPSs that bode well for practical realizations of very low complexity ScPs. Conclusions are offered in Section V.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

II. Secure Co-Processors

A secure co-processor (ScP) is a trusted computer which is expected to protect its secrets, and execute only software authenticated using the such secrets. Tamper *responsive* ScPs will "self-destruct" by *zeroizing* its secrets under suspicions of tampering attempts. ScPs typically include an autonomous processor, along with volatile and non volatile storage, and often dedicated hardware for the realization of various symmetric and asymmetric cryptographic computations, all enclosed in a tamper-proof casing. More often, all components are included in a single chip.

In most architectures for ScPs [4], [5], [6], [7], a single *master* secret is used to encrypt all other secrets that are indirectly protected by the ScP. The master secret is spontaneously generated inside the ScP, using dedicated hardware. The master secret will typically be stored in special battery backed volatile register (BBRAM) [5], [7]. The internal source of power also powers some protection circuits used exclusively for protecting the master secret. No software (including the ScP kernel) has access to the master secret [5], [7]. All cryptographic computations that employ the master secret, like encrypting / decrypting other secrets, will be realized using a special CPU instructions (which in turn employs a hardware block cipher). Secrets encrypted using the master secret can be safely stored *outside* the ScP.

Enforcing the trust boundary of an ScP involves providing assurances of components inside the boundary, through *countermeasures* that *shield* the components from intrusions that can 1) reveal secrets or 2) modify the code / data stored inside the ScP. The countermeasures take the form of active and passive shields. The primary purpose of passive shields is to block electromagnetic radiations: 1) radiations emanating from inside the chip from leaving the chip boundary, and 2) radiations originating outside from entering the chip. Radiations emanating from the chip can be used to reveal some information about the secrets used. External radiations aimed at the chip could be used for inducing faults which can in turn lead to compromise of cryptographic keys [2], [8].

Active shields detect intrusions and trigger circuitry for zeroizing. For instance, sophisticated attacks involving focused ion beam (FIB) techniques [9] can permit an attacker to drill fine holes and establish connections with the computer buses, and thereby have access to the bits that pass through the buses when the CPU is functioning. The active shields [6], [4] used as countermeasures typically take the form of a mesh (or layers of meshes) of non-intersecting conductors which will be open-circuited by microprobes and picoprobes [2] (FIBs), resulting in erasure of the master secret.

A. Remnance

Ideally, zeroizing should involve a single step, viz., erasing the master secret by removing power supply to the BBRAM, which renders all secrets / data encrypted using the master secret irrecoverable. Unfortunately the ability of attackers to exploit *remnance* in volatile memory, calls for some *additional* steps to achieve effective zeroisation.

Bits stored in volatile memory, especially for extended periods, can leave "footprints" that can be scavenged [10], [4] even after the power supply is removed. The ability of the attacker to "scavenge" bits from such footprints can be improved significantly by cooling the chip (say by immersing it in liquid nitrogen). Safe deletion of contents in magnetic and solid state memory may require many *repeated* overwriting operations.

The countermeasures against attacks that exploit remnance include periodic ones-complementing of the BBRAM (the master secret) using a dedicated circuitry for this purpose [10], [6] so that when the power supply is removed no footprints are left behind. However such levels of protection cannot be afforded to other volatile memory regions (like RAM and cache memory) where stored data may be actively in use in computations.

While it may seem that such attacks can be avoided if sensitive information is not stored in RAM / cache for *extended* periods, even storing secrets for *fleeting* durations in RAM can be risky. Very simple attacks are possible by inducing faults in memory [11] that could just cause the CPU to hang. Even with good passive shielding to ensure that the risks of such attacks are minimal, there may be numerous other reasons like hardware / software bugs which may result in the CPU hanging. If this occurs while a sensitive secret is stored in the RAM attackers could wait for some duration to ensure that the secret leaves a deep footprint before plunging it in liquid nitrogen, and scavenge the contents at liesure.

Countermeasures to prevent scavenging of contents from RAM / cache include special sensors that respond to sudden changes in temperature [6], [10] and trigger clean erasure (by repeated overwriting) of volatile memory regions. Obviously, dedicated circuitry that work *independent* of the CPU are required for this purpose. In order to ensure that such countermeasures are executed completely (providing an adequate response time to execute countermeasures) it is also necessary to inhibit *rapid* cooling by increasing the *mass* of the ScP [4].

1) Multi-step Countermeasures: The need to address problems associated with remnance can substantially increase the complexity and cost of ScPs. Instead of a simple single-step countermeasure (erasing the master secret), effective zeroisation now calls for a *second* step, which requires

① sensors for detecting rapid changes in temperature;

② exclusive circuitry for erasing footprints (when active shields or temperature sensors are trigerred); and

③ increasing the mass of ScPs.

Apart from the obvious fact that the second step is *expensive* (increases the complexity of ScPs), *multi-step countermeasures are inherently vulnerable*. With complete knowledge of the layout of the components (which attackers can easily determine by tampering with a few chips / modules [2]), attackers can "force their way in" using FIBs to cut off circuitry (or power supply) responsible for undertaking the countermeasures. The attacker does *not* have to worry about triggering the first step to scavenge contents of RAM. The DOWN policy proposed in the next section *eliminates* the need

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

for multi-step countermeasures.

III. The DOWN Policy

The end result of attacks that exploit the inherent weakness of multi-step countermeasures is that it is likely for an attacker to get a "snapshot" of all contents of volatile memory (RAM / cache) at *any* instant of time. Even while the attacker is *restricted to a single* snapshot (as the ScP is irrevocably destroyed in the process of scavenging) such a snapshot obtained (say) when an ScP was computing an RSA signature, could reveal the *entire RSA private key*.

A. Elementary DOWN Operations

If attackers can get such a snapshot, a solution (to cut our losses) is to make sure that the RAM has very minimal sensitive information at *any point in time*. The DOWN policy comes out of the realization that many cryptographic operations can be split into *atomic parts*. In other words, observing the DOWN policy requires the ability to perform computations using *fractional parts of secrets.*

Let M be the master secret of an ScP, and $K(X)$ denote encryption of a value X using the secret K, employing a block cipher. Let $M_1 \cdots M_t$, where (say) $M_i = M(i)$ represent t secrets derived from the master secret. Let S be a private secret protected by the ScP, and used for decryption messages to / signing messages from the ScP.

With the DOWN policy, the secret S is split into t fractional parts $S_1 \cdots S_t$. The secrets are stored encrypted as $M_1(S_1) \cdots M_2(S_t)$, possibly outside the ScP. Computations that employ the secret S are broken down into t "elementary DOWN operations." In each such operation

1) an elementary secret $M_i(S_i)$ is fetched and decrypted (using a special CPU instruction) to obtain S_i;
2) the fractional secret S_i is used in some computation; and
3) the memory location where S_i was stored is flushed clear by repeated overwriting (before the next fractional part of secret is fetched and decrypted).

Thus, at *no* point in time, will a "snapshot" reveal more than one fraction S_i of the secret S.

1) DOWN with RSA: Let the secret S be an RSA private exponent d (say of size b bits). Let n be the (public) RSA modulus. Decryption of a some cipher text C, or computing $P = C^d \bmod n$ involves modular exponentiation of C with d. Modular exponentiation is often performed using the square-and-multiply ([12], Chapter 5) algorithm.

Let the binary representation of $d \in \mathbb{Z}_n$ is $\delta_1\delta_2 \cdots \delta_b$ (or $\delta_i, i = 1 \leq i \leq b$ are the b bits of d, where δ_1 represents the MSB and δ_b the LSB). The evaluation of $P = C^d \bmod n$ with the square-and-multiply algorithm proceeds as b steps,

$$z_i = \begin{cases} z_{i-1}^2 \bmod n & \text{if } \delta_i = 0 \\ z_{i-1}^2 C \bmod n & \text{if } \delta_i = 1 \end{cases} \quad (1)$$

with z_0 initialized to 1, and $z_b = P$. Note that in each step (loop) only one bit of the private key d is required. Thus with the DOWN policy, we could set $S_i = \delta_i$ - or each bit is a fractional part that can be used independently. No snapshot will reveal more than *one bit* of the private key.

DOWN also extends itself readily to many other asymmetric schemes like

① encryption schemes like RSA, El Gamal
② Diffie-Helman key exchange,
③ RSA based signatures
④ DSA and other El Gamal - like signature schemes, and
⑤ encryption and signature schemes based on elliptic curves.

For ① - ③ the *operation performed with the private key* is exponentiation. For ④ the operation is *multiplication* [12]. Note that even for modular multiplication of $ab \bmod n$ (where a is a secret) only one bit of a is needed in every loop. For ECC schemes, each bit of the private key[1] dictates whether the group operation involves "group doubling" of "doubling and group addition" [12].

Computation of *multiplicative inverses* however, say $b = a^{-1} \bmod m$, where only one part of the secret a can be revealed at any time, can be challenging. While many asymmetric primitives require evaluation of multiplicative inverses, fortunately, inverses of *secrets* are not called for. Unfortunately, computing multiplicative inverses with *secrets* are required for identity based encryption (IBE) and signature (IBS) schemes [13] that are attracting substantial attention due to the fact that they eliminate the need for certificates. IBE and IBS schemes (as yet) do *not* lend themselves well to the DOWN implementations.

2) DOWN assurance and complexity: The *DOWN assurance* provides a guarantee that an attacker can expose no more than one elementary secret by tampering with an ScP, assuming that *the master secret cannot be compromised*. In other words, the DOWN assurance relies only on the *first*-step countermeasure (erasing the master secret). By simply *tolerating* the fact that the attacker cannot expose more than one bit (or fraction) of the private key, we can *eliminate* the need for the expensive (and vulnerable) second step.

The *complexity* imposed by DOWN is primarily dependent on the number of "elementary DOWN operations" into which the process of deriving a security association is split into. For 1024-bit RSA private exponent d, the DOWN complexity is 1024 DOWN operations. Each step (loop) calls for an additional symmetric cipher operation. Obviously it is also possible to say use just *two* DOWN operations for this purpose, in which case an attacker can determine up to 512 bits of the RSA private exponent from a snapshot.

3) Reducing ScP costs: Several factors contribute to the high cost of ScPs (compared to other generic processors), including manufacturing scale, need for extensive protection circuitry, and special shielding techniques that are required to simultaneously facilitate effective shielding and heat dissipation [6]. The DOWN policy can reduce costs of ScPs by eliminating expensive multi-step countermeasures.

[1]Just as each bit of the secret exponent determines whether the operation to be performed in a loop is "squaring" or "squaring and multiplication" for exponentiation using square and multiply algorithm.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

Another strategy to reduce the cost of ScPs is to restrict the ScPs to purely *symmetric cryptographic primitives*. More specifically, if the complexity of operations performed by the ScP is low enough to eliminate the need for proactive methods for heat dissipation, *unconstrained* shielding techniques can be used, which can be simultaneously inexpensive and effective.

However, in the envisioned application scenarios of the future, key distribution schemes used for authentication of devices should support *ad hoc* establishment of security associations. Thus schemes like Kerberos, which require ongoing involvement of a trusted server, are not well suited. However, key predistribution schemes (KPS) facilitate ad hoc authentication without the use of asymmetric primitives. Furthermore, like IBE and IBS schemes KPSs are inherently ID based, and do not require dissemination of certificates. Unfotunately KPSs are "fragile," due to their susceptibility to collusions.

In the next section we argue probabilistic KPSs (PKPS) have some very desirable properties that make them very well suited for use with *very low complexity*, inexpensive ScPs. Specifically, 1) they lend themselves very well to the DOWN policy, and 2) they can take advantage of resources outside the ScP to increase their resistance to collusions.

IV. PKPS Synergies

Most PKPSs exploit the property of "uniqueness of intersections" of subsets. Many subset allocation schemes using *deterministic* allocation strategies [14], have been proposed in literature [15], [16]. Dyer et al [17] (in 1995) were the first to point out the simplicity and effectiveness of *random* subset allocations. The idea of random subset allocation has received substantial attention in the recent past in the context of sensor and ad hoc networks [18], [19]. We refer to all such schemes ([17] - [19]) as RPS (random preloaded subsets).

A. Random Preloaded Subsets

RPS (as defined in [19]) is defined by two parameters P and k, where $\xi = k/P < 1$, and a public random function $F()$. The KDC chooses an indexed set of P secrets $\mathbb{S}$. For node A (or node with ID A) the *public* function $F(A)$ (a random sequence generator, seeded by the ID) generates k *indices* between 1 and P, or $F(A) = \mathbb{I}_A$. The k indices assigned to node A determines the k secrets $\mathbb{S}_A$ provided to A. Thus,

$$\mathbb{S} = \{K_1, \ldots, K_P\} \qquad \mathbb{S}_A = \{K_{A_1}, \ldots, K_{A_k}\}$$
$$\mathbb{I}_A = F(A) = \{A_1, \ldots, A_k\} \quad \mathbb{I}_{AB} = F(A) \cap F(B)$$

Any two nodes A and B will *share* (on an average) $m = \xi k = \xi^2 P$ indices represented by the set $\mathbb{I}_{AB}$. Note that any entity can discover the set of indices $\mathbb{I}_{AB}$ as the node IDs and the function $F()$ are public. However only[2] nodes A and B have access to *all* m secrets $\mathbb{S}_{AB}$ corresponding to the indices $\mathbb{I}_{AB}$. Thus A and B can discover a shared secret K_{AB} derived from all m secrets in $\mathbb{S}_{AB}$.

An attacker who has exposed all secrets from n nodes, say $M_1 \cdots M_n$ can determine K_{AB} (or *all* m secrets in $\mathbb{S}_{AB}$) if $F(A) \cap F(B) \in \{F(M_1) \cup F(M_2) \cup \cdots \cup F(M_n)\}$. The *probability* $p(n)$ of this event, the optimal choice of ξ that minimizes p (or maximizes n), the minimum value of k, and the average value m (of the number of intersecting indices) are [20]

$$p(n) = (1 - \xi(1-\xi)^n)^k \qquad \xi = \tfrac{1}{n+1}$$
$$k_{min} = (n+1)\mathrm{e}\log(1/\mathrm{p}) \quad m = k\xi = \mathrm{e}\log(1/\mathrm{p})$$

In other words, RPS is (n, p)-secure, as an attacker who has exposed all secrets from n nodes can discover only a fraction p of all possible pairwise secrets. As a numerical example, RPS with $P = 2^{21}$ and $k = 2^{14}$ (or $\xi \approx 1/2^7 = 1/128$), is $(n = 128, p \approx 3.7 \times 10^{-21})$-secure. By exposing *all* secrets from 128 nodes, the attacker can expose one in 2.7×10^{20} pair-wise secrets. The number of *symmetric cipher operations*, $m \approx 128$, is *independent* of n.

For example, for increasing n by a factor 4, we need to 1) increase k by a factor 4, and 2) increase P by a factor 4^2 (or decrease ξ by a factor 4). Such a scheme (with $P = 2^{25}, k = 2^{16}$) is $(512, 3.7\times10^{-21})$-secure, while still requiring the *same* number $m = 128$ of symmetric cipher operations.

B. PKPS with DOWN

With the DOWN policy, only *one* of the k secrets assigned to any ScP will be exposed during each DOWN operation. Thus with the DOWN assurance, an attacker can expose only one secret from each node, as long as the first-step (and only step) countermeasure *cannot* be circumvented.

If only a fraction ρ of the k secrets can be exposed from any node, an (n, p)-secure PKPS is rendered *at least* $(n/\rho, p)$-secure [21]. With the DOWN assurance, $\rho = 1/k$. Thus with the DOWN assurance, an (n, p)-secure PKPS is rendered (nk, p)-secure

1) DOWN complexity: The DOWN complexity of *both* $(P = 2^{21}, k = 2^{14})$ and $P = 2^{25}, k = 2^{16})$ RPS are the same - $m = 128$, as only m secrets are *used* for evaluation of any pairwise secret. With the DOWN assurance the former is $(128 \times 2^{14}, p)$-secure (or 2-million-secure), and the latter is $(512 \times 2^{16}, p)$-secure (32-million-secure). In other words, with the DOWN assurance, the number of nodes that the attacker has to destroy, increases as k^2.

One of the appealing features of RPS is that the we can increase n (and k) *without* increasing the DOWN complexity, which simultaneously provides a return proportional to $nk \propto k^2$. On the other hand, for deterministic KPSs like Blom's scheme [22] or the scheme by Matsumoto et al [23], the DOWN complexity is also k - as evaluation of any pairwise secret will call for the use of *all* k secrets. Thus while Blom's scheme with k keys is roughly $(k-1)$-secure without DOWN assurance and $k(k-1)$-secure *with* the DOWN assurance, k *cannot* be increased substantially without increasing the DOWN complexity. For a DOWN complexity of 128, Blom's scheme is just $16,256$-secure (with the DOWN assurance).

2) Making Use of External Resources: As the k secrets assigned to each node are stored encrypted, they could be stored unprotected - for example in flash memory *outside* the ScP. With ever increasing storage capabilities, the extent to

[2]With a high probability.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

which k can be increased is *not* likely to be limited by storage concerns. However, while increasing k does not increase the "DOWN complexity" m, it increases the complexity of evaluation of the *public functions*, or more specifically, evaluation of $\mathbb{I}_{AB} = F(A) \cap F(B)$ to determine the m intersecting indexes (the complexity is $\mathbb{O}(k)$).

Firtunately, the public functions (like $F(A) \cap F(B)$) can be evaluated by resources *outside* the ScP as they *do not require operations with secrets*. Alternately, a special *unprotected* (outside protective shields) circuitry could be used for this purpose, for which heat dissipation is *not* and issue. As the one-way function $F()$ need not be cryptographic [20], even (many parallel banks of) linear feedback shift registers (LFSB) can be used for this purpose, where the initial state of the LFSB is determine by the node IDs.

Only the m symmetric cipher operations need to be performed inside the trusted boundary. ScPs with a single AES block cipher in hardware, and a processor equivalent in capabilities to the processors a quarter century ago can be more than adequate for this purpose. With such low complexity ScPs heat dissipation is not likely to be an issue, thus facilitating unconstrained shielding techniques, which can be *inexpensive* and *effective*. Synergistically, the low complexity inside the trust boundary (which translates to unconstrained shielding techniques) can provide a very high level of confidence in the first-step (and only step with the DOWN policy) countermeasure. Note that the DOWN assurance relies on the fact that the first step countermeasure cannot be circumvented.

With ever increasing capabilities of storage, even 256 MB for storage of secrets may not be an issue, as long as storage does not have to be *inside* the ScP. By increasing k from 2^{16} to 2^{25} (and thereby calling for 256 MB of storage for keys), and increasing[3] P to 2^{43}, we can increase n to 2^{18}. Such a scheme (with $P = 2^{43}, k = 2^{25}$) is $(2^{18}, 3.7 \times 10^{-21})$-secure without the DOWN assurance, and $2^{18} \times 2^{25}$- secure (or **over 8 trillion secure**) with the DOWN assurance. Even with 64 MB (or one fourth of 256 MB) of storage, the collusion resistance is decreased by a factor 16 (which still over half a trillion). Obviously, under such scenarios the "susceptibility of KPSs to collusions" is not a practical concern.

V. Conclusions

We introduced a simple algorithmic approach for protecting secrets in secure co-processors which can reduce the cost of practical realizations of ScPs. While the DOWN policy relies on the ability to perform computation with fractional parts of secrets, most asymmetric primitives readily lend themselves to this requirement.

We then argued that probabilistic key predistribution schemes are very amenable to efficient implementations of the DOWN policy. This DOWN-PKPS synergy results in dramatic improvements in the efficiency or security-to-complexity ratio of PKPSs. The feasibility of very low complexity and hence inexpensive ScPs can help towards accelerating the wide spread use of such ScPs, which are required for the practicality of many evolving application scenarios based on the paradigms of ubiquitous and autonomic computing.

[3]The KDC does not actually have to store P values as it can *generate* any value form a single or a few highly protected secrets.

References

[1] D. M. Chess, C. C. Palmer, S. R. White, "Security in an autonomic computing environment," IBM Systems Journal, **42** (1), 2003.

[2] R. Anderson, M. Bond, J. Clulow, S. Skorobogatov, "Cryptographic Processors - a survey," University of Cambridge, Computer Laboratory Technical Report, UCAM-CL-TR-641, Aug 2005.

[3] A. Pfitzmann, B. Pfitzmann, M. Schunter, and M. Waidner, "Mobile User Devices and Security Modules: Design for Trustworthiness"; IBM Research Report RZ 2784 (#89262) 02/05/96, IBM Research Division, Zurich, Feb. 1996.

[4] J.D Tygar, B. Yee, "Dyad: A system for Using Physically Secure Coprocessors," Technological Strategies for the Protection of Intellectual Property in the Networked Multimedia Environment, pp 121–152, 1994.

[5] D. Lie, C. A. Thekkath, M. Horowitz, "Implementing an Untrusted Operating System on Trusted Hardware," Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 178-192. ACM Press, October 2003.

[6] S.W. Smith, S. Weingart, "Building a High-Performance Programmable Secure Coprocessor," IBM Technical Report RC21102, Feb 1998.

[7] J.P. McGregor, R. B. Lee, "Protecting Cryptographic Keys and Computations via Virtual Secure Coprocessing," ACM SIGARCH Computer Architecture News archive, **33**(1), March 2005.

[8] E. Biham, A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," Lecture Notes in Computer Science, **1294**, 1997.

[9] O. Kommerling, M. Kuhn, "Design principles for tamper-resistant smartcard processors," Proceedings of the Usenix Workshop on Smartcard Technology, pp. 9–20, 1999.

[10] P. Gutman, "Secure Deletion of Data from Magnetic and Solid-State Memory," Sixth USENIX Security Symposium, San Jose, California, July 1996.

[11] R. Anderson, M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," IWSP: International Workshop on Security Protocols, Paris, April 1997.

[12] D. R. Stinson, *Cryptography, Theory and Practice,* Second Edition, Chapman and Hall CRC, 2002.

[13] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing," Advances in Cryptology – Crypto'2001, Lecture Notes on Computer Science 2139, Springer-Verlag (2001), pp. 213–229.

[14] P. Erdos, P. Frankl, Z. Furedi, "Families of Finite Sets in which no Set is Covered by the union of 2 Others," *Journal of Combinatorial Theory, Series A,* **33**, pp 158–166, 1982.

[15] L. Gong, D.J. Wheeler, "A Matrix Key Distribution Scheme," *Journal of Cryptology*, **2**(2), pp 51-59, 1990.

[16] C.J. Mitchell, F.C. Piper, "Key Storage in Secure Networks," *Discrete Applied Mathematics,* **21** pp 215–228, 1995.

[17] M. Dyer, T. Fenner, A. Frieze and A. Thomason, "On Key Storage in Secure Networks," *Journal of Cryptology,* **8**, 189–200, 1995.

[18] L. Eschenauer, V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," Proceedings of the Ninth ACM Conference on Computer and Communications Security, Washington DC, pp 41-47, Nov 2002.

[19] M. Ramkumar, N. Memon, R. Simha, "Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks," Globecom-2003.

[20] M. Ramkumar, N. Memon, "An Efficient Random Key Pre-distribution Scheme for MANET Security," IEEE Journal on Selected Areas of Communication, March 2005.

[21] M. Ramkumar, N. Memon, "On the Security of Random Key Predistribution Schemes," 5th Annual IEEE Information Assurance Workshop, West Point, NY, June 2004.

[22] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Advances in Cryptology: Proc. of Eurocrypt 84*, Lecture Notes in Computer Science, **209**, Springer-Verlag, Berlin, pp. 335-338, 1984.

[23] T. Matsumoto, H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem. CRYPTO 1987: 185-193.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.