Trustworthy Computing under Resource Constraints with the DOWN Policy

Mahalingam Ramkumar, *Member*, *IEEE*

Abstract—Trustworthy computing modules like secure coprocessors (ScP) are already in extensive use today, albeit limited predominantly to scenarios where constraints on cost is not a serious limiting factor. However, inexpensive trustworthy computers are required for many evolving application scenarios. The problem of realizing inexpensive ScPs for large-scale networks consisting of low-complexity devices have not received adequate consideration thus far. We introduce two strategies toward realizing low-cost ScPs. The first is the decrypt only when necessary (DOWN) policy, which can substantially improve the ability of low-cost ScPs to protect their secrets. The DOWN policy relies on the ability to operate with *fractional* parts of secrets. Taking full advantage of the DOWN policy requires consideration of the nature of computations performed with secrets and even the mechanisms employed for distribution of secrets. We discuss the feasibility of extending the DOWN policy to various asymmetric and symmetric cryptographic primitives. The second is cryptographic authentication strategies which employ only symmetric cryptographic primitives, based on novel ID-based key predistribution schemes that demand very low complexity of operations to be performed by the ScP and can take good advantage of the DOWN policy.

Index Terms—Trustworthy computing, read-proofing, key predistribution.

1 Introduction

Many emerging applications will rely on extensive mutual co-operation among a highly interconnected network of computers [1]. A group of computers working together may decide the setting of a thermostat based on weather forecasts received directly from computers in the local weather station. Computers in a car, interacting with computers in cars nearby may decide the best course of action to avoid an impending collision. Sensors monitoring vital internal organ functions of a person on the road may relay early warning signs over multihop ad hoc networks to the nearest hospital to facilitate timely responses.

In such applications, each device is expected to perform some tasks for the *overall good* of the network. An obvious requirement in such scenarios is the ability to *trust* the devices. It does not take much imagination to see the consequences of an attacker's ability to impersonate a sensor to send a false alarm or a malicious course correction.

Realizing *widespread* adoption of such applications mandates sufficiently *trustworthy* computers that can be realized at *low cost*. Apart from facilitating deployment of futuristic applications, the ability to realize trustworthy computers at low cost can also address many of the security issues that plague our *existing* network infrastructure.

Trustworthy computers [2] provide assurances against 1) tampering of the software executed by such devices and 2) exposure of secrets used for *authentication* of the devices. Trustworthy computing modules like cryptographic coprocessors or more generally secure coprocessors (ScP) have seen widespread use in a variety of military and civilian application scenarios. Low-end ScPs have been used in ATMs since the early 1980s and in various smart cards and set-top boxes (like cable and satellite TV receivers) since the early 1990s [3]. Higher end ScPs (like IBM 4758/4764) have found extensive use in securing servers that cannot be afforded proper physical protection. Although the assurances provided by the high-end ScPs (which can range from few hundreds to many thousands of dollars) are considered acceptable, the assurances provided by low-end solutions thus far have been far from satisfactory [4].

Providing assurances of trustworthiness entails providing assurances of *reliability* and effective *shielding* of components inside a "trusted boundary" from intrusions (aimed at modifying software or exposing secrets). For this purpose, ScPs include passive and active shields and complex circuitry to trigger zeroisation (or deletion of all secrets) when intrusions are suspected.

Although, at first sight, "inexpensive" and "trustworthy" may seem mutually exclusive, a possible strategy is to reduce the *complexity* of the components inside the trusted boundary. The often heard statement that "complexity is the enemy of security" is far from dogmatic. For one, lower complexity implies better verifiability of compliance. Furthermore, keeping the complexity inside the trust boundary at low levels can obviate the need for proactive measures for heat dissipation. Strategies constrained to *simultaneously* facilitate shielding *and* heat dissipation tend to be expensive [5]. On the other hand, unconstrained shielding strategies can be reliable *and* inexpensive to facilitate.

1.1 Specific Contributions

Reducing the complexity of ScPs, particularly the complexity *inside* the trusted boundary, can thus lead to lowered cost and improved reliability of ScPs. In the rest of this paper, we do *not* try to justify this assumption any further. We propose two strategies for lowering the complexity of ScPs.

• *The author is with the Department of Computer Science and Engineering, Mississippi State University, Box 9637, Starkville, MS 39762. E-mail: ramkumar@cse.msstate.edu.*

Manuscript received 3 Aug. 2006; revised 8 June 2007; accepted 21 Aug. 2007; published online 31 Aug. 2007.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0114-0806.
Digital Object Identifier no. 10.1109/TDSC.2007.70227.

The first is the decrypt only when necessary (DOWN) policy, which reduces the complexity of circuitry for tamper responsiveness and simultaneously provides improved assurances against compromise of secrets. The second is low-complexity mutual authentication strategies suitable even for very large-scale deployments of trustworthy computers. The authentication strategies demand very low-computational overheads inside the trusted boundary.

1.1.1 The DOWN Policy

The DOWN policy is motivated by the fact that any countermeasure (against intrusions) that requires *multiple* steps is expensive *and* inherently vulnerable. One line of possible attacks that can exploit this weakness is a result of the property of remnance in volatile memory. Existing countermeasures against this attack are 1) expensive, 2) vulnerable, and 3) especially ill-suited for low complexity ScPs. The DOWN policy eliminates the need for multistep countermeasures and, in this process, obviates the need for complex circuitry required for facilitating (inherently vulnerable) multistep countermeasures. Although implementation of the DOWN policy relies on the ability to perform computations with *fractional* parts of secrets, most asymmetric cryptographic primitives meet this requirement. The DOWN policy imposes very low overheads and is thus well suited for use even in low-end ScPs.

1.1.2 Low Complexity Authentication Strategies

Although restricting ScPs to perform *only symmetric cipher operations* can significantly reduce the complexity and, hence, the cost of realizing ScPs, they do not, in general, *scale* as well as asymmetric schemes. Specifically, scalable authentication strategies that employ only symmetric cryptographic primitives require a trusted server for mediation (Kerberos-like schemes based on the symmetric Needham-Schroeder [6] protocol) *or* require guarantees that *collusions* of more than a certain number of nodes (say, n) is infeasible (key predistribution schemes).

Although the computational complexity inside the trust boundary needs to be lowered to the extent possible, reliance on storage (especially storage *outside* the ScP) does not in any way affect the cost of ScPs. Furthermore, storage is an increasingly abundant and inexpensive resource for almost any conceivable application scenario. Some of the desirable properties of the KPS presented in this paper, namely, multiple basic key distribution (MBK) and hashed MBK (HMBK), are that they

1. take good advantage of inexpensive external storage;
2. lend themselves well to the DOWN policy;
3. benefit substantially from the assurances provided by the DOWN policy; and
4. demand very low-computational complexity inside the ScP (only a few tens of symmetric block-cipher operations are mandated).

Thus, although KPSs are inherently susceptible to collusions, we show why the ability of MBK and HMBK to take advantage of external storage resources together with the assurances provided by the DOWN policy render this issue *irrelevant* in practice. For example, with 8 Mbytes of storage per ScP (which could be *outside* the ScP—for example, in a flash storage card or even storage accessed over an insecure network) HMBK can resist collusions of 8,000 nodes (an attacker has to expose all secrets from 8,000 of the deployed nodes, irrespective of the total number of nodes that are deployed). However, in conjunction with the assurances provided by the DOWN policy, the collusion resistance of HMBK can be increased to *8 billion* nodes. If 128 Mbytes of storage is feasible, the attainable collusion resistance is over *2 trillion* with the DOWN policy in effect.

The implication is that mutual authentication of *very large scale* networks are feasible using only low complexity symmetric cryptographic primitives, as long as an attacker cannot compromise secrets from *billions* or *trillions* of nodes!

1.1.3 Organization

In Section 2, we provide a brief overview of some of the salient features of practical approaches for realization of ScPs. In Section 3, the DOWN policy is investigated, and its suitability is explored for various conventional certificates based asymmetric schemes like RSA, El Gamal, and elliptic curve (ECC) schemes. In Section 4, we investigate the suitability of DOWN for identity-based encryption (IBE) and signature (IBS) schemes. We then motivate the need for low complexity ID-based authentication schemes for ScPs for evolving application scenarios. Section 4 includes an overview of some existing low-complexity ID-based KPS. In Section 5, we outline MBK and HMBK. Conclusions are offered in Section 6.

2 Secure Coprocessors

Practical realization of ScPs calls for two fundamental assurances: *read proofing* of secrets and *tamper proofing* of software. Tamper proofing (or write proofing) of software ensures that the software controlling the functions of ScP cannot be modified by unauthorized entities. Read proofing is necessary to ensure that secrets protected by a ScP, which will be used for authentication of the ScP, cannot be exposed.

The two requirements are however *not* independent. With the ability to modify software at will, an attacker can force the ScP to reveal its secrets (for example, by inserting a set of commands to write the secret bytes out to the serial port). On the other hand, secrets that are protected can be used to authenticate software that will be executed by the computer, using (for example) key-based hashed message authentication codes (HMAC). Without the knowledge of the secret used for computing the HMAC, the attacker cannot modify the software.

In practice, read proofing is seen as a stepping stone to the more elusive goal of tamper proofing of software. Attacks aimed at modifying software *to reveal secrets* can be prevented by ensuring that the software does not have access to at least some of the secrets that are protected. Some secrets may be *generated, stored, and used* by dedicated hardware [5], [7]. However, authenticating software with the secrets provides a boot-strapping problem [8]. After all, some software should be loaded (typically the BIOS), which includes instructions to load the secret and perform the authentication. Recently, Gennaro et al. [9] have argued that providing assurances that software cannot be modified entails assurances of read proofing *and* the additional assurance of a write protected nonvolatile counter.

2.1 Secure Coprocessor Design

The problem of practical realization of ScPs has received significant attention over the last two decades especially since the development of the ABYSS coprocessor [10] in the late eighties. Even with substantial changes in semiconductor technology and the capabilities of tools that can be utilized by attackers, the core principles behind possible attacks and countermeasures have not changed significantly.

Most solutions for tamper responsive ScPs like ABYSS [10], Citadel [11], Dyad [12], IBM 4758 [5], and Cerium [13] consist of a tamper-resistant package that includes the CPU, DRAM, battery-backed RAM (BBRAM), and flash ROM. Tamper attempts will result in *zeroizing* or erasure of secrets stored. In almost all approaches, a secure public-private key pair is generated inside the device, and only the public key is exported. The private key (typically a private RSA exponent) is stored in BBRAM and is protected at all times—even when the CPU is off.

Most ScPs will also generate a private symmetric *master* secret that can be used to encrypt all other secrets that need to be protected. The master secret can be used to encrypt even the private key (say RSA decryption exponent) when the CPU is off. The encrypted private RSA exponent can then be stored in nonvolatile memory (NVM) that is not afforded any protection. Thus, only the master secret stored in BBRAM needs to be protected when the device is off. However, when the device is in the on state, other physical areas of the ScP (like DRAM, special cache memories, etc.) are also extended protection. Such protection measures take the form of active and passive shields and circuitry that execute countermeasures for zeroizing when active sensors are triggered by intrusions.

2.2 Active and Passive Shields

Passive shields block inbound and outbound electromagnetic radiations. Outbound radiations (emanating from inside the chip) can be used to reveal some information about the secrets used. Inbound radiations can be used for inducing faults, which can in turn lead to compromise of cryptographic keys [14], [15]. For example, differential power analysis (DPA) [16] can be used for gaining clues about secrets based on instantaneous power consumption by the processor. Countermeasures against DPA include introducing redundant steps in cryptographic computations [17] and the use of self-timed circuits [18].

Active shields strive to identify intrusions and activate circuitry for zeroizing. Thus, active shields are also sensors that can trigger various countermeasures. For instance, sophisticated attacks involving focused ion beam (FIB) techniques [19] can permit an attacker to drill fine holes and establish connections with the computer buses. With such taps, the attacker can gain access to the bits that pass through the buses when the CPU is functioning. The active shields used as countermeasures typically take the form of a mesh (or many layers of meshes) of nonintersecting conductors [5], [12]. They can prevent microprobes and picoprobes [3] from gaining line-of-sight access to the buses. Even if one line of the mesh is cut, the resulting open circuit will trigger circuitry for zeroisation. Even if we can ensure that only a fraction of the lines can be tapped, it may be possible to use *private circuits* [20] to ensure that the attacker gains no knowledge (by tapping a few lines).

2.3 State Transitions, Trust Boundaries, and Remnance

Any trusted computer defines a clear trust boundary. For example, for a single chip ScP all components inside the chip may fall under such a trust boundary. *Enforcing* the trust boundary is by proactive measures for protection of components within the boundary. However, the regions inside a trust boundary that are physically protected can change dynamically, depending on the *state* of the ScP. As discussed earlier in Section 2.1, when the CPU is off, there is no need to extend protection to all regions. However, when the CPU is on, the scope of protection will need to be wider.

Even when the CPU is on, the scope of protection may be restricted to very small regions inside the ScP. For example, operations involving secrets may be permitted only in a *concealed execution mode* [21], during which many buses may be disconnected from the processor. Even buses to a general purpose RAM can be disconnected. Only some special cache memory regions may be used as work benches for cryptographic computations. Other data/code may be encrypted/authenticated and paged to areas [7] that may not be extended protection.[1]

2.3.1 Remnance

One of the hidden problems associated with the dynamic scope of countermeasures takes the form of *remnance* in volatile memory [22], [23]. Bits stored in volatile memory (especially for extended periods) can leave "footprints" that can be "scavenged" *even after the power supply is removed*. The ability of the attacker to scavenge bits from footprints can be improved by cooling the chip (say, by immersing it in liquid nitrogen). Safe deletion [22] of contents in magnetic and solid state memory may require many *repeated* overwriting operations. Thus, even after volatile memory regions like cache/RAM have been powered off (in the off state, where only the BBRAM is extended protection), secrets that *were* stored in RAM/cache regions (while the device was on) can be exposed.

The countermeasures against attacks that can exploit this weakness include:

1. clean erasure (by repeated overwriting) of contents of volatile memory (like RAM/cache memory, except contents of BBRAM) before turning the power off, and as part of zeroisation;
2. ensuring that secrets are not stored for long durations in RAM;
3. the use of special sensors that respond to sudden changes in temperature and trigger clean erasure [12] of volatile memory regions;
4. increasing the mass of the modules to inhibit *rapid* cooling [12] to provide an adequate response time to execute countermeasures; and
5. periodic ones-complementing of some highly sensitive secrets [5], [22], (for example, contents of the BBRAM) with dedicated circuitry for this purpose, so that when the power supply is removed, no footprints are left behind.

1. As mentioned in Section 1, the need for limiting the scope of countermeasures to the smallest possible extent lies at the heart of every trusted computing effort. Tangible reasons include reducing complexity (of) and reducing heat dissipated (by) the components within the scope of countermeasures.

3 The DOWN Policy

For many of the application scenarios that is of interest to us, increasing the mass of ScPs may simply not be a viable option. Furthermore, although periodic ones-complementing may be possible for a limited number of secrets (for example, contents of the BBRAM), extending such protection to all volatile memory regions like cache memory and RAM, where the values stored may be *actively* used in computations, may not be practical.

At first sight, it may seem that such issues are of no concern as long as secrets are not stored in the same location for long periods. Unfortunately, even storing secrets for fleeting durations in RAM can be risky, as very simple attacks are possible by inducing faults in memory [23] that could cause the CPU to hang. Even with good shielding to ensure that the risks of such attacks are minimal, there may be numerous other reasons, including hardware/software bugs, which may result in the CPU hanging. If the CPU hangs while a sensitive secret is stored in the RAM, an attacker can wait for some duration to ensure that the secret leaves a deep footprint before plunging it in liquid nitrogen.

Although sensors that can detect rapid changes in temperature (which obviously should work independent of the CPU and are well protected by active shields) can erase contents of the RAM, the repeated overwriting operations mandated for *clean erasure* of all sensitive information in the RAM may not be possible. Even repeated overwriting may not be a satisfactory solution for DRAMs [22] for which the only option may be to ensure that sensitive values are not stored for extended durations[2] (even a few tens of seconds).

3.1 Vulnerability of Multistep Countermeasures

Countermeasures that involve more than a single step to be effective are *inherently vulnerable*. With complete knowledge of the layout of the components (which attackers can easily determine by tampering with a few chips/modules [3]), attackers can "force their way in" using FIBs to cut-off circuitry (or power supply) that is responsible for undertaking the countermeasures. Even with good shielding and assurances that it is not possible for intrusive attempts to evade active shields, with such attacks, the attacker does *not* have to worry about triggering active shields (which leads to erasure of master secret). As long as the circuitry for taking *additional* countermeasures (like clean erasure of contents of RAM) can be cut-off, the attacker can still scavenge bits from RAM.

3.1.1 *The Snapshot*

The end result of such attacks is that it is possible for an attacker to get a *snapshot* of all contents of all volatile memory regions (except the BBRAM) at any instant of time. However, the attacker is limited to a *single* snapshot as the ScP is irrevocably destroyed in this process, and the master secret used for encrypting all other secrets is erased. Nevertheless, the attacker gains knowledge of secrets/data that may have been stored unencrypted in RAM, even temporarily, *when* the attack was carried out.

A solution to minimize the damages is to make sure that such snapshots reveal as little useful information as possible for the attackers. The DOWN policy is motivated by the realization that many cryptographic operations can be performed with fractional parts of secrets. At any point in time, only a small part of a secret may be necessary for cryptographic computations. Thus, while multistep countermeasures require extensive circuitry to (strive to) ensure that such snapshots will reveal no information, the DOWN policy *obviates* the need for such measures by ensuring that there is very little useful information to be gained from a snapshot in the first place.

For example, *without* the DOWN policy, if the attack was carried out when the ScP was computing an RSA signature, the entire private key can be exposed from RAM. With the DOWN policy however (as we shall see), the attacker will be limited to exposing *no more than one bit* of the RSA private key.

3.2 DOWN Enabled Trust Modules

For DOWN enabled ScPs, the master secret K_M is generated spontaneously inside the ScP. The master secret is the only key that is *directly* protected by the ScP. The master secret is stored in a special volatile register (a BBRAM). The CPU also has exclusive access to a hardware block cipher. The BBRAM and the hardware block cipher are *hidden* from the OS kernel [21]. The processor exposes dedicated CPU instructions for using the hardware cipher in conjunction with the master secret K_M for encrypting/decrypting secondary secrets that are *indirectly* protected by the ScP (for example, an RSA private exponent).

The ScP draws power from external devices for its operation. Even when the CPU is off, battery backups power the BBRAM and minimal active circuitry required for protecting the master secret. The active circuits include mechanisms for periodically ones complementing the master key so that even if the battery backups are cut-off, no decipherable footprints of the master secret are left behind. The battery or power lines from the battery do *not* need to be protected. *For any countermeasure, the **only** step is the erasure of the master secret by removing power supply to the BBRAM.*

Observing the DOWN policy requires the ability to perform computations using fractional parts of secrets. Let

1. K_M be the master secret of an ScP;
2. $K(X)$ denote encryption of a value X using the secret K, employing a block cipher;
3. $M_1 \cdots M_t$, where (say) $M_i = K_M(i)$ represent t secrets derived from the master secret; and
4. S be a secret indirectly protected by the ScP (for example, RSA private key).

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 the ith DOWN operation,

1. the elementary secret $M_i(S_i)$ is fetched, decrypted (using a special CPU instruction) to obtain S_i, and S_i is used in computations; and
2. the memory location where S_i was stored is flushed clear by repeated overwriting before the next fractional part of the secret (or S_{i+1})) is fetched and decrypted. Alternately, the secret S_{i+1} may be used to *overwrite* S_i to avoid overheads required for "flushing clean" the footprints left behind by S_i.

2. For clean erasure of contents stored for long durations in DRAM, the only option (apart from heating) may be to store some random value for a long duration to "dilute the stress" [22] imposed on the oxide layer by the old data.

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

3.3 Protecting Private Keys with Down

We shall now see how the DOWN policy can be applied for protecting the private keys of various asymmetric schemes.

3.3.1 Down with RSA

In RSA, two large secret primes p and q are chosen, and the RSA domain $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$ is computed as $n = pq$. A value e, $3 \leq e \leq \Phi(n) = (p-1)(q-1)$ is chosen as the *public exponent* subject to the constraint that e and $\Phi(n)$ are relatively prime. The inverse d of e in the modular domain of $\Phi(n)$, or $d \equiv e^{-1} \mod \Phi(n)$ is the *private* exponent. The values p, q and $\Phi(n)$ are then destroyed.

For encryption, the ciphertext $C \in \mathbb{Z}_n$ corresponding a plain text $P \in \mathbb{Z}_n$ is computed as $C \equiv P^e \bmod n$. The decryption of C is performed as $P \equiv C^d \bmod n$. Similarly, signing a hash H of a message is performed as $S \equiv H^d \bmod n$, and verification of the signature S is achieved by computing $H = S^e \bmod n$.

The intent of the DOWN policy is to protect the *private* exponent d from being scavenged from memory. Thus, computations performed for encryption and verification of signatures (which do *not* use the private key) are not influenced by the DOWN policy. The private exponent d is used for decryption and signing. More specifically, the private exponent needs to be stored in RAM for performing computations like $P \equiv C^d \bmod n$ (decryption) and $S \equiv H^d \bmod n$ (signing).

Modular exponentiation is often performed using the square-and-multiply ([24, Chapter 5]) algorithm. Let the binary representation of d be $\delta_1\delta_2\cdots\delta_b$ (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 in b steps:

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

with z_0 initialized to 1. The value $z_b = P$ is the output of the last step. Note that in each step (loop), *only one bit of the private key* d is required.

Thus, the private exponent d can be stored as b independent encryptions of each bit. For each step in the evaluation of the square-and-multiply algorithm, one encrypted bit is fetched, decrypted, and used in modular computations. Thus, no snapshot will reveal more than one bit of the private key. Recall that *without* the DOWN policy, the *entire* private key could be exposed by a snapshot.

3.3.2 DOWN with Other Asymmetric Schemes

The effectiveness of the DOWN policy is intricately tied to the nature of cryptographic computations that have to be performed *using the private key*. Such computations may involve different types of finite field (or ring or group) operations like exponentiation, multiplication, and computation of multiplicative inverses. As seen earlier, modular exponentiation (where the exponent is a secret to be protected) is naturally facilitated. Modular multiplication of two quantities (one of which is a secret to be protected) can also be facilitated in the same way. Just as exponentiation involves "squaring" or "squaring and multiplication" in every loop (depending on whether the particular bit of the private key is a 0 or a 1), for multiplication, each loop involves "doubling" (left shift) or "doubling and addition," (left shift followed by addition) depending on the particular bit of the private key.

3.3.3 DOWN with Exponential Ciphers

For example, in the El Gamal cryptosystem ([24], Section 6) over $\mathbb{Z}_p$, using a primitive element $g \in \mathbb{Z}_p$, private key $a \in \mathbb{Z}_p$ and public key $\alpha = g^a \bmod p$, encryption of a message x using the public key and decryption using the private key are carried out as follows:

$$e_K(x, k) = (y_1, y_2), \begin{cases} y_1 = g^k \bmod p \\ y_2 = x\alpha^k \bmod p \end{cases} \tag{2}$$
$$d_K(y_1, y_2) = x = y_2(\{y_1^a\})^{-1} \bmod p,$$

where $k \in \mathbb{Z}_{p-1}$ is randomly chosen by the entity encrypting the message. The scope of the DOWN policy in this case is to ensure that no more than a small fraction of the private key a is decrypted and stored in RAM at any point in time during the computation of y_1^a. As in the case of RSA, exponentiation with a requires only one bit of a in each loop of the square and multiply algorithm.

In the El Gamal signature scheme ([24, Section 7.3]), $\{p, g, a, \alpha\}$ over $\mathbb{Z}_p$, where $g \in \mathbb{Z}_p$ is a primitive element, and $a \in \mathbb{Z}_p$ is the private key of the signer, and $\alpha = g^a \bmod p$ is the corresponding public key, the signature $\mathbf{sig}(x, k)$ for a value $x \in \mathbb{Z}_p$, and a random $k \in \mathbb{Z}_{p-1}$ is

$$\mathbf{sig}(x, k) = (\gamma, \delta), \begin{cases} \gamma = g^k \bmod p \\ \delta = (x - \{a\gamma\})k^{-1} \bmod p - 1. \end{cases} \tag{3}$$

The operation performed with the secret a (during signing of a message) is evaluation of $a\gamma$. Multiplication with a can also be trivially performed using only one bit of a in each loop.

The popular Diffie-Helman (DH) key exchange algorithm also lends itself readily to DOWN. In the DH key exchange algorithm, two nodes A and B agree on some prime p and $g\mathbb{Z}_p$, choose secrets a and b, respectively, and make public $\alpha = g^a \bmod p$ and $\beta = g^b \bmod p$, respectively, to establish a shared secret $K_{AB} = \alpha^b = \beta^a$.

3.3.4 Elliptic Curves

DOWN also readily extends itself to protecting the private keys of ECC (ECC, [24, Section 6.5])-based systems. ECCs form an additive group $\mathcal{G} \in \mathbb{Z}_p \times \mathbb{Z}_p$ defined over a finite field $\mathbb{Z}_p$. For example, if points like $P = (x_P, y_P), Q = (x_Q, y_Q) \in \mathcal{G}$ (where x_p, y_P, x_Q, $y_Q \in \mathbb{Z}_p$) lie on the elliptic curve, then points like $P + P$, $P + Q$, $P - Q$ also lie on the curve (see Appendix A.2).

For ECC schemes, the private key is a randomly chosen value $a \in \mathbb{Z}_p$. The security of ECC schemes rely on the assumption that if $P' = aP$, where $P, P' \in \mathcal{G}$, even with the knowledge of P *and* P', it is infeasible to evaluate a. The operation performed with the secret (private key) a in all ECC schemes involves computation of a value aP. Multiplication of a point $P \in \mathcal{G}$ by a value $a \in \mathbb{Z}_p$ is carried out as $\log_2(a)$ doubling/"doubling and addition" group operations, where only one bit of a need to be used at any time. Thus, ECC schemes also easily lend themselves well to the DOWN policy.

3.3.5 Generation of Private Keys

A *strict* implementation of the DOWN policy mandates that the DOWN policy should be observed *throughout the life cycle* of the ScP. Although for most public key schemes observing the DOWN policy for *using* the secrets is trivial, for RSA, it may be very difficult to observe the DOWN policy during the *generation* of primes p and q. Furthermore, many of the optimizations employed for speeding up exponentiation [25] in RSA can render observing the DOWN policy even for using[3] the private key difficult.

However, generation of secrets is not an issue for El Gamal (and variants) and ECC schemes, where the private key can be *randomly* chosen from $\mathbb{Z}_p$. Thus, for such schemes, it is possible to generate, encrypt, and store each *bit* of the private key a independently.

3.4 DOWN Assurance and Complexity

The *DOWN assurance* provides a guarantee that an attacker can expose no more than one elementary fraction of the secret (private key) by tampering with an ScP, *if the master secret cannot be compromised*. In other words, the DOWN assurance relies *only* on the *first*-step countermeasure (erasing the master secret by removing power supply to the BBRAM). By simply *tolerating* the fact that the attacker cannot expose more than one bit (or fraction) of the private key, the DOWN policy eliminates the need for the expensive and inherently vulnerable multistep measures.

Without the DOWN policy, we saw that several *additional* countermeasures are mandated to address the problem of remnance. Specifically, such countermeasures mandated 1) sensors for detecting rapid changes in temperature, 2) exclusive circuitry for erasing footprints (when active shields or temperature sensors are triggered), and 3) increasing the mass of ScPs. Furthermore, such expensive countermeasures are *still* vulnerable as an attacker with complete knowledge of the layout of an ScP can still expose an entire private key from RAM. With the DOWN policy, the attacker is restricted to exposing at most one bit of the private key.

The complexity imposed by DOWN depends on the number of elementary DOWN operations into which the process of decryption/signing is split. For 1,024-bit d, the DOWN complexity is 1,024 DOWN operations. However, for protecting the 1,024-bit RSA private exponent, we do not *need* to employ 1,024 DOWN operations. If we employ only two (where, in each DOWN operation, 512 bits of the private key are decrypted), no more than 512 bits of the private key can be revealed by a snapshot. In practice, if the block cipher used by the ScP employs 128-bit block sizes it may be more efficient to store the private key in 128-bit chunks. Each DOWN operation can call for kernel mode switching if the dedicated CPU instruction for encrypting/decrypting secrets is permitted only in a special secure kernel mode [7] (or a *concealed execution mode* [21]).

The DOWN policy readily lends itself to asymmetric schemes as long as the operations that employ the private key is restricted to modular exponentiation (RSA, DH key exchange, and El Gamal encryption schemes) or multiplication (El Gamal signature scheme and variants and ECC). However, the DOWN policy is *better suited for El Gamal and ECC schemes*, where there are *no restrictions on the choice of the private key* (unlike RSA where it is required to verify that the private keys p and q are indeed primes).

4 ID-Based Schemes

Key distribution schemes can be broadly classified into certificates-based and ID-based schemes. In the more conventional certificates-based schemes each entity is associated with 1) an ID, 2) public key, and 3) private key. Each entity is free to *choose* their own private key and compute the corresponding public key. An entity A can choose a private key R_A and derive a public key[4] U_A.

As the public key U_A provides no information about the identity A, a trusted third party has to securely specify a *binding* between the identity and the public key of every entity. This binding is usually specified through a *certificate*. For mutual authentication of entities A and B, they need to exchange their respective public-key certificates and perform some computations.

For ID-based schemes, the ID of an entity itself doubles as the public key, obviating the very need for certificates. In ID-based schemes, a key distribution center (KDC) chooses public parameters of the system and one or more master secrets. Using the secrets, the KDC can compute the private key(s) corresponding to *any* public key (ID). The private keys for a node with identity A are thus *assigned* by the KDC to the node A.

ID-based schemes are increasingly seen as preferable over certificates-based schemes for large-scale networks, and especially for many emerging application scenarios like ad hoc networks. In typical client-server interactions in existing networks, the client and server exchange public key certificates for mutual authentication, at the end of which, a shared secret is established. This secret can be used for authentication and encryption of a *large* number of packets exchanged between them subsequently. Thus, the overheads (exchange of certificates and their verification) incurred for establishing a shared secret can be leveraged for securing large amounts of data.

However, in ad hoc networks, a node will typically exchange small packets with *many* nodes. Thus, the overheads for exchanging certificates with each node may be prohibitive. The overheads may become especially high for very large-scale networks where *chains* of certificates [26] will need to be exchanged. With ID-based schemes, two entities A and B can independently compute a shared secret K_{AB} without exchanging certificates for this purpose.

Another desirable feature of ID-based schemes, especially for their use in conjunction with *tamper responsive* devices, comes from the fact that the keys are implicitly escrowed (by the KDC). In tamper-responsive devices, false-alarms leading to unintended zeroisation can never be ruled out. Without key escrow, an unfortunate end-user may be locked out of all data encrypted using a secret protected by the ScP. With escrowed ID-based schemes, such devices can be easily reinstated.

3. Most such optimizations involve exponentiating with the private key in $\mathbb{Z}_p$ and $\mathbb{Z}_q$, where $n = pq$ is the RSA modulus. Thus, *both* the exponent and the modulus (p and q) have to be protected—which may not be feasible.

4. For RSA, the secret primes p and q are chosen before the public value $n = p \times q$ can be evaluated. For El Gamal, the secret a is chosen before the public value $g^a \bmod p$ is evaluated.

4.1 IBS and IBE Schemes with DOWN

The first identity-based *signature* (IBS) scheme was proposed by Shamir [27].

4.1.1 Shamir's IBS Scheme

In this scheme, the KDC chooses 1) two large primes p and q, where $n = pq$, 2) $e \in \mathbb{Z}_n$ relatively prime to $\Phi(n) = (p-1)(q-1)$ (and e is preferably a large prime), and 3) a one-way function $f()$.

The KDC makes n, e, and $f()$ public. A node with ID ID_i is provided with a secret g_i, where $g_i^e \equiv ID_i \bmod n$—or g_i is an eth root of ID_i in $\mathbb{Z}_n$ (which can be easily computed by the KDC as the KDC knows the factors of n). To sign a message M, the signer 1) chooses a random $r \in \mathbb{Z}_n$, 2) computes $t \equiv r^e \bmod n$, and 3) computes $\alpha = r^{f(t,M)}$. The signature for a message M is (s, t), where

$$s \equiv \{g_i \alpha\} \bmod n. \tag{4}$$

The verification condition is $s^e \equiv ID_i t^{f(t,M)} \bmod n$.

Note that the operations with the secret g_i by the signer only involve *multiplication* ($\{g_i \alpha\}$), which poses no problems with DOWN implementations.

4.1.2 Pairing-Based Schemes

Shamir's identity-based scheme does not support encryption. Boneh and Franklin [28] responded to Shamir's challenge to develop the first ID-based scheme that could support both encryption and signatures. Such ID-based encryption (IBE) and signature (IBS) schemes [28], [29] rely on a bilinear mapping $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$, where $\mathcal{G}_1$ is an additive group, and $\mathcal{G}_2$ is a multiplicative group. Typically, $\mathcal{G}_1$ is a special ECC, and the mapping e represents a class of Weil pairings [28]. For pairing-based IBE/IBS schemes, the private key assigned to each node is a point in the ECC $\mathcal{G}_1$.

The pairing operation requires group additions involving a secret in $\mathcal{G}_1$, which in turn calls for a computation of *multiplicative inverses* using the secret. Computation of multiplicative inverses, say, $b = a^{-1} \bmod m$, where only one part of the secret a can be revealed at any time, does not appear to be trivial.

Although ECC schemes call for group addition (which requires computation of multiplicative inverses—see Appendix A.2), the operation is performed on points on the ECC, which *do not reveal any information about the private key.* For ECC scheme points on the elliptic curve may be publicly known generators or intermediate values of computations. For pairing-based IBE schemes on the other hand, the *private key is itself a point on the* elliptic curve. Thus, the more versatile pairing-based IBE/IBS schemes do not (yet) have simple DOWN implementations.

4.2 Low-Complexity ID-Based Encryption Schemes

Although IBE/IBS schemes are very much desirable (especially for use in conjunction with tamper-responsive devices), we saw that encryption schemes (IBE) may not lend themselves to simple DOWN implementations. Furthermore, they demand high complexity inside the ScP (perhaps computationally more expensive than conventional public key schemes for the same level of security). What we *ideally* desire are ID-based schemes that demand *very low* computational complexity inside the ScP. Thus, *limiting ScPs to perform only symmetric cryptographic computations can be a good strategy.*

4.2.1 Scalability

Scalable networks should cater for practically unlimited number of entities and permit such entities to join the network at any time. One of the compelling advantages of asymmetric schemes is that they scale very well (ID-based schemes require less overheads than certificates-based schemes in this respect). Note that any number of entities can be assigned private keys (for ID-based schemes) or can receive certificates from the certificate authority (for certificates-based schemes) and can receive the private key/public key certificate at any time.

Authentication strategies for scalable networks based purely on symmetric cryptographic primitives either require a trusted server for mediation (Kerberos-like schemes based on the symmetric Needham-Schroeder protocol [6]) *or* are susceptible to collusions. The former is unacceptable for many application scenarios. The latter is facilitated by KPS.

In the rest of this section, we provide an overview of different KPSs. In Section 5, we introduce some novel KPSs that can provide such high levels of collusion resistance with very low overheads, wherein their susceptibility to collusions is of very little practical consequence.

4.2.2 Key Predistribution Schemes

A KPS consists of a KDC and N entities with unique IDs (say, $A, B, C, \ldots$). The KDC chooses a set of P secrets $\mathbb{S}$. Each entity receives k secrets. The set of k secrets $\mathbb{S}_A$ assigned to A is a function of $\mathbb{S}$ and the ID A. A and B can independently discover a pairwise K_{AB} using (in general) $m \leq k$ of their k secrets.

The total number of entities N (or the maximum network size) that can be assigned secrets is only limited by the number of bits used for representing the IDs (IDs should be unique). However, KPSs are susceptible to *collusions*. An attacker who has physically compromised many entities and gained access to their secrets may be able to compromise pairwise secrets between entities that have *not* been physically compromised. An n*-secure* KPS can resist an attacker who has pooled secrets from up to n entities. For most n*-secure* KPSs $k \propto n$, and $P \propto n^2$.

4.2.3 The "Basic" KPS

For the trivial "basic" KPS, for a network size of N, the KDC chooses $P = \binom{N}{2}$ secrets, and each entity is assigned $k = N - 1$ secrets. Although such a scheme is *not* susceptible to collusions, its scalability is severely restricted as each entity needs to store $\mathbb{O}(N)$ values. With a limitation of $\mathbb{O}(n)$ storage, the "basic" KPS can support only a network size of n. Scalable KPSs, on the other hand, (with the same limitation of $k = \mathbb{O}(n)$ storage) can support unrestricted network sizes but can only tolerate collusions of (up to) n entities.

4.2.4 SKGS

That security-complexity trade-offs are possible to support unlimited network size was first realized by Blom [31] who proposed the first KPS in the literature. In the SKGS (symmetric key generation system) based on MDS (maximum distance separation) codes proposed by Blom [31], a n*-secure* scheme requires $k = n + 1$ secrets to be assigned to each node. For SKGS $m = k$, namely, all k secrets need to used for evaluating *any* pairwise secret (see Appendix A.1).

4.2.5 LM-KPS

Leighton and Micali [30] were the first to propose a KPS (LM-KPS) with probabilistic (n,p)-secure assurances. For an (n,p)-secure KPS, an attacker who has access to all secrets of n entities can expose a fraction p of all pairwise secrets. Unlike deterministic KPSs where the failure of the KPS occurs catastrophically, (n,p)-secure PKPSs fail gracefully with increasing n. In other words, $p(n)$ increases gracefully with increasing n. As long as $p(n)$ is low (say, 2^{-64} for some sufficiently large n), it is computationally infeasible for an attacker who has exposed all secrets from n entities to even determine *which* pairwise secrets can be compromised using the pool of exposed secrets.

In LM-KPS defined by two parameters (k,L), the KDC chooses a set of $P=k$ secrets $\{K_1\cdots K_k\}$, and each entity is provided with a set of k secrets. The k secrets are repeatedly hashed versions (between 1 and L times) of the secrets chosen by the KDC. Thus, A is provided with secrets

$$\mathbb{S}_A=\{K_1^{\bar{a}_1},K_2^{\bar{a}_2},\ldots,K_k^{\bar{a}_k}\},\quad 1\leq\bar{a}_i\leq L, \tag{5}$$

where $K_i^j=h^j(K_i)$ represents the result of successive hashing (j times) of K_i using a secure hash function $h()$. The parameter L is thus the "maximum hash depth." For LM-KPS, the upper bound for k for some desired (n,p)-security is $\mathbb{O}(n^3)$ [30]. For LM-KPS $m=k$ (like SKGS), as all k secrets of a node are used for evaluation of any pairwise secret.

4.2.6 Random Allocation of Subsets (RAS)

Many KPSs based on random allocation of a subset (RAS) of k keys to every entity from a pool of P keys chosen by the KDC have been proposed. For ID-based RAS schemes, the indices of the secrets assigned to A is tied to the ID A through a one-way function [33], [34]. Extensions of RAS schemes with LM-KPS [30] have also been proposed [35]. For ID-based RAS schemes, a public function $F(A)$ determines the k of the P indices of secrets assigned to A. For HARPS [35], $F(A)$ generates k hash depths (between 1 and L) in addition to the k indices. For all such schemes for (n,p)-security, we require $k\propto n\log(1/p)$ and $P/k\approx n$. As any two nodes will share only $k^2/P=\mathbb{O}(\log(1/p))$ secrets (on the average), only $m=\mathbb{O}(\log(1/p))$ of the k secrets need to be *used* for evaluation of any pairwise secret. However, A and B have to compute $F(A)\cap F(B)$ to *determine* the m shared indices. The complexity for evaluating $F(A)\cap F(B)$ is at least $\mathbb{O}(k)$.

5 DOWN Friendly Key Predistribution Schemes

For all KPSs, every secret can be used by the entities *and* delivered to the entities independently. Thus, strict implementation of the DOWN policy does not pose any major hurdles. The DOWN assurance guarantees that *no more than one* of the k secrets can be compromised from an entity (ScP) as long as the first-step (and only step) countermeasure cannot be bypassed.

With the DOWN assurance, an attacker has to expose one secret from k ScPs in order to compromise an "equivalent" of one ScP. It should be noted that the equivalence of "one secret from k ScP," and "all k secrets from one ScP" is a strict one *only* for deterministic KPSs. It is actually a *pessimistic estimate* for (n,p)-secure KPSs. In such schemes, although an attacker is guaranteed k *independent* secrets by exposing "all k secrets from one ScP," exposing "one secret from k ScPs," may result in *less than* k independent secrets (as the same secret could be "compromised" from different nodes). Thus, for such schemes, the attacker has to expose one secret from *at least* k ScPs to compromise an equivalent of one ScP.

An n-secure KPS with k secrets is rendered nk-secure with the DOWN assurance. An (n,p)-secure KPS with k secrets per entity is rendered *at least* (nk,p)-secure.

The k secrets assigned to any ScP do not have to be stored inside the ScP. They will be encrypted with the master secret and stored outside the ScP. For example, encrypted secrets could be stored in pluggable flash storage devices (SD cards supporting several Gbytes of storage are already very common) or even storage locations that can be readily accessible over (possibly insecure) networks. Even if millions of secrets are assigned to each ScP, the storage required is only a few megabytes. Thus, the storage complexity for the secrets is *not* a serious issue in almost every conceivable application scenario.

Also, note that, although k secrets need to be stored, in general, only $m\leq k$ need to be used for evaluation of a pairwise secret. The value m is also the DOWN complexity as only one secret can be exposed in RAM in an elementary DOWN operation. The value m also influences bandwidth overheads in scenarios where encrypted secrets have to be fetched over a network. KPS schemes that are DOWN *friendly* should ideally have the following properties 1) $n\propto k$ in order to take good advantage of the DOWN assurance, 2) low value of m, and 3) low-computational overheads.

For the two schemes (MBK and HMBK), we shall see that 1) the storage required is $k\propto n$, 2) the DOWN complexity m can be made *as small as we desire*, and 3) the computational complexity is also $\mathbb{O}(m)$. The security of such schemes is therefore limited *only* by available (external) storage. Thus, although *even without* the DOWN assurance, they can make good use of storage to reduce their susceptibility to collusions (as $n\propto$ the storage k) with the DOWN assurance their collusion resistance increases as $nk\propto k^2$.

5.1 MBK: Multiple "Basic" KDS

In the basic KDS, for a network of N entities, the KDC chooses $\binom{N}{2}$ secrets, and each entity is provided with $N-1$ secrets. The primary problem with the basic KDS is that it does not scale very well. A trivial scalable extension of basic KDS is to employ several such independent "small-scale" deployments of basic KDSs in parallel.

In the MBK, m such independent deployments, each catering only for a network size of M are used. However, together, the m systems cater for practically unrestricted network sizes. We will assume that each entity is assigned a b-bit ID (say, $b=128$ or 160) to facilitate choice of IDs as a secure one-way function of textual descriptors.

The KDC chooses m sets $\mathbb{S}^1\cdots\mathbb{S}^m$, *each* set consisting of $\binom{M}{2}+M=\binom{M+1}{2}$ secrets, where

$$\mathbb{S}^i=\{K_i(j_1,j_2)\},\quad 1\leq i\leq m,\quad 1\leq j_1,j_2\leq M, \tag{6}$$

and $K_i(j_1,j_2)=K_i(j_2,j_1)$. An entity with b-bit ID A is assigned m "short-IDs," in each of the m systems. Thus, each short-ID is s-bit long, where $s\geq\log_2(M)$ (for $M=1024=2^{10}$, each short-ID will be 10 bits long). More specifically, a simple one-way function $a_i=f(A,i)$, $1\leq i\leq m$ is used to assign such short-IDs (m random integers between 1 and M) to A. The function $f()$ could be a pseudorandom bit stream generator, where each instance of execution of $f()$ calls for generation $\log_2 M$ bits.

Now, A is provided with M secrets from each of the m systems—corresponding to the short-ID in each system. Specifically, A with short IDs $a_1 \cdots a_m$ is provided with $k = m \times M$ secrets

$$\mathbb{S}_A = \{K_i(a_i, j)\}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq M. \tag{7}$$

B, like wise, is provided with $k = Mm$ secrets $\mathbb{S}_B = \{K_i(b_i, j)\}$.

A and B can independently discover m shared secrets $S_i = K_i(a_i, b_i) = K_i(b_i, a_i)$ by evaluating $f(A, i)$ and $f(B, i) m$ times (for $1 \leq i \leq m$). Both A and B can determine their respective short IDs $\{(a_i, b_i)\}$ in all m schemes. In order to establish a session secret K_S with B, A chooses K_S at random and encrypts K_S successively with each of the m secrets $S_i, 1 \leq i \leq m$. Thus, MBK calls for 1) m evaluations of $f()$ to determine the indices of m secrets to be fetched from storage (where mM encrypted secrets are stored), 2) m DOWN operations, and 3) m block cipher operations.

5.2 MBK Performance

MBK provides probabilistic assurances. First note that there is a finite probability that two entities with different b-bit "long" IDs may be assigned the same set of m short IDs ($\log_2 M$-bits each). The probability of this event however is very low—the same as the probability of collision in a $m \log_2 M$-bit hash function. For example, for $m = 64$, $M = 1,024$, this probability is a miniscule $2^{-64\times 10/2} = 2^{-320}$.

What is of greater concern is the ability of an attacker who has managed to extract all mM secrets from n entities. Although with the DOWN assurance, an attacker cannot extract more than one secret from each entity, we shall (for now) ignore the assurance provided by the DOWN policy. Let $p(n)$ represent the probability that an attacker who has pooled together all secrets from n entities (that does not include A and B) can determine K_{AB}. It can be shown (see Appendix A.3) that $p(n)$ and the parameters M and m are related as

$$p(n) \approx (1 - \mathrm{e}^{\frac{-2n}{M}})^{\mathrm{m}}. \tag{8}$$

5.2.1 Choice of Parameters m and M

The choice of m and M that *minimizes* $k = mM$ (and thus storage for the secrets) for a desired $p(n)$ is (see Appendix A.3)

$$\left.\begin{array}{l} m^* = \log(1/p)/\log(2) \\ M^* = 2n/\log(2) \end{array}\right\} \Longrightarrow k^* = M^* m^* = \frac{2n \log(1/p)}{\log^2 2}. \tag{9}$$

If storage is not an issue, it is indeed preferable to minimize m, as the value m is simultaneously 1) the number of elementary DOWN operations, 2) the complexity of the public function, and 3) the bandwidth overhead in scenarios where the secrets are fetched over a network.

For some desired $p(n)$, if we desire to reduce m by a factor a (to $m = m^*/a$), we need to increase M and $k = Mm$, where $k > k^*$. It can be shown (see Appendix A.3) that

$$\frac{k}{k^*} = \frac{mM}{m^* M^*} = \frac{\log(1 - 1/2)}{a \log(1 - 1/2^a)}. \tag{10}$$

For example, for $a = 2$, $k/k^* = 1.204$. For $a = 3$, $k/k^* = 1.730$.

TABLE 1
Comparison of SKGS, RAS, and MBK

KPS	Storage (k)	m	CPF	Coll. Resist.	
				WoD	WD
SKGS	$n+1$	$n+1$	None	n	$n(n+1)$
RAS	$ne \log(1/p)$	$\mathrm{e} \log(1/p)$	$\mathbb{O}(k)$	(n, p)	(nk, p)
MBK	$\frac{2n \log(1/p)}{\log^2 2}$	$\frac{\log(1/p)}{\log 2}$	$\mathbb{O}(m)$	(n, p)	(nk, p)

Numerical example. MBK with parameters $M = 2,048$, $m = 64$ is ($n = 710, p = 2^{-64}$)-secure (or $p(710) = 2^{-64}$). The storage required for each entity is 1 Mbyte (for $mM = 131,072$ 64-bit secrets). With the DOWN assurance, an attacker has to compromise one secret each from over nk ScPs (about 93 *million* ScPs). The same $p(n)$ can also be achieved by choosing $m = 24$ and $M = 8,192$ for which $k = mM$ is higher by a factor 1.5. However, with the DOWN assurance the latter can resist compromise of one secret from over 140 million ScPs (as nk is also higher by a factor 1.5).

5.2.2 Down and MBK Synergy

It is interesting to note the synergy between DOWN and MBK facilitated by inexpensive storage resources. Although reducing m renders $k = mM > k^*$ (increase in storage required) for the same $p(n)$, it is really not so terrible a disadvantage in conjunction with the DOWN policy. When the number of secrets assigned to each ScP is increased, the "utility" of each secret is low—both for the possessor of the secret *and* the attacker. With the DOWN assurance, the extent of "information" that can be retrieved by tampering with any ScP is *diluted* when a large number of secrets are assigned to each ScP. The DOWN assurance thus has this very desirable property of converting what is conventionally seen as (storage) *inefficiency* of a KPS into *better security*! The fact that storage is the least expensive of resources bodes very well for KPSs that can take advantage of this resource.

Increasing the storage complexity (beyond k^*) can further reduce the already low $\mathbb{O}(m)$ computational complexity. Recall that the DOWN assurance relies on the assumption that the master secret cannot be compromised. The lower the computational complexity inside the trust boundary, the lower the restrictions on possible shielding techniques, and the higher the reliability of the single-step countermeasure for protecting the master secret. Thus, increased availability of storage indirectly improves the *basis* of the DOWN assurance *and* leads to better collusion resistance.

5.2.3 MBK versus RAS Schemes

It is pertinent to point out that random subset allocation schemes require less storage (by a factor 1.53—see Appendix A.4) compared to MBK to achieve the same $p(n)$. However, for such schemes, the computational complexity for determining the shared indices is $\mathbb{O}(k)$. For MBK, it is just $\mathbb{O}(m)$. Furthermore, unlike MBK, where it is possible to reduce m (by increasing k) for RAS schemes m cannot be reduced below $\log(1/p)$ to achieve (n, p)-security (see Appendix A.4).

Table 1 provides a ready comparison of different KPSs in terms of the relationships between

1. total storage (k);
2. number of DOWN operations (m);
3. complexity of public functions (CPF);

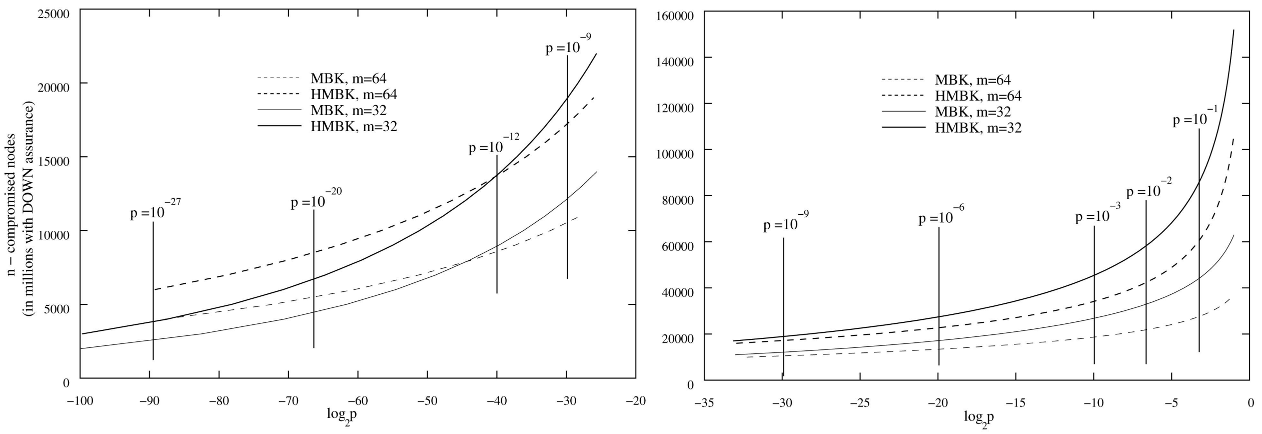

Fig. 1. Comparison of MBK and HMBK for the same storage complexity—$k = mM = 2^{20}$ (one million keys) for two different values of m (the DOWN complexity), $m = 64$ $(M = 2^{14})$ and $m = 32$ $(M = 2^{15})$. For HMBK, $L = 64$. The y-axis is n—the number of compromised nodes (*without* the DOWN assurance. For performance, in conjunction with the DOWN assurance, the y-axis should be read in *millions* (as $k = mM = 2^{20}$ for all schemes). The x-axis is $\log_2(p)$. For convenience, the plot is divided into two, one for $p < 10^{-9}$ and one for $p > 10^{-9}$).

4. collusion resistance without DOWN assurance (WoD); and
5. collusion resistance with (WD) DOWN assurance.

5.3 Hashed MBK

A simple extension of the MBK scheme, the HMBK can improve the performance of MBK for the same M and m. Specifically, HMBK realizes improvements over MBK by realizing an increase in n for the same $p(n)$. HMBK is actually a combination of MBK and the KPS with probabilistic assurances proposed by Leighton and Micali, LM-KPS [30], defined by two parameters (k, L) (see Section 4.2.5).

HMBK is defined by three parameters (M, m, L). As in MBK, the KDC chooses m sets of secrets[5] $\mathbb{S}^1 \cdots \mathbb{S}^m$, where each set consists of $\binom{M+1}{2}$ secrets. All such secrets could also be generated from a single master secret M_i chosen by the KDC.

However, the public function $f()$ now produces two values—a $\log_2 M$-bit value and a $\log_2 L$-bit value, where L is the maximum hash depth of the keys. For example, for $M = 1{,}024$ and $L = 64$, $f(A, i) = [a_i \| \bar{a}_i]$ returns a 16 bit integer—the first 10 bits determines the short-ID $1 \leq a_i \leq M$, and the last six bits determine the hash depth, $1 \leq \bar{a}_i \leq L$. The secrets assigned to node A are now

$$\begin{aligned} \mathbb{S}_A &= \{K_i^{\bar{a}_i}(a_i, j)\}, \quad \text{where} \\ K_i^{\bar{a}_i}(a_i, j) &= h^{\bar{a}_i}(K_i(a_i, j)), \end{aligned} \tag{11}$$

for $1 \leq i \leq m$, $1 \leq j \leq M$. In other words, a secret $(K_i(a_i, j)$ is repeatedly hashed $\bar{a}_i$ times before it is assigned to A as $h^{\bar{a}_i}(K_i(a_i, j))$.

As in MBK, any two entities can determine m shared secrets. In MBK, for A and B (with $f(A, i) = a_i$ and $f(B, i) = b_i$), the secret corresponding to index i is $K_i(a_i, b_i) = K_i(b_i, a_i)$. For HMBK, however, A has the secret $K_i^{\bar{a}_i}(a_i, b_i)$, and B has the secret $K_i^{\bar{b}_i}(a_i, b_i)$ (where $f(B, i) = b_i \| \bar{b}_i$). Thus, if $\bar{a}_i > \bar{b}_i$, B has to repeatedly hash its secret $K_i^{\bar{b}_i}(a_i, b_i)$, $(\bar{a}_i - \bar{b}_i)$ times to determine a common secret $S_i = K_i^{\bar{a}_i}(a_i, b_i)$. Likewise, if $\bar{b}_i > \bar{a}_i$, A will have to hash its secret $K_i^{\bar{a}_i}(a_i, b_i)$, $(\bar{b}_i - \bar{a}_i)$ times. For each of the m indexes, the shared secret between A and B is at a hash depth $\max(\bar{a}_i, \bar{b}_i)$—or

$$S_i = K_i^{\max(\bar{a}_i, \bar{b}_i)}(a_i, b_i). \tag{12}$$

The probability $p(n)$ for HMBK (see Appendix A.5) is then

$$p(n) = (1 - \epsilon'(n))^m, \quad \epsilon'(n) = \sum_{j=1}^{L} \frac{2l - 1}{L^2} \left(1 - \frac{l\gamma}{L}\right)^n. \tag{13}$$

As a first-order approximation, it can also be shown that $\epsilon'(n) \approx \epsilon(3n/2)$, implying that HMBK can tolerate 3/2 times as many compromised nodes as MBK for the same value of m and $k = mM$.

5.3.1 HMBK Performance

Fig. 1 depicts plots of n versus $\log_2(p)$ for MBK and HMBK for $(m = 2^6, M = 2^{14})$, $(m = 2^5, M = 2^{15})$. Note that for both cases, $k = mM = 2^{20}$. $L = 64$ for HMBK. The choice of $k \approx$ one million is primarily for convenience as the collusion resistance *with* DOWN assurance $(nk = nmM)$ can be readily calculated by scaling the value of n (y-axis) by a *million*. For example, HMBK with $m = 64$, $M = 16{,}384$, $L = 64$ is $(n = 8{,}484, p = 10^{-20})$-secure *without* the DOWN assurance and is $nk = 8{,}484 \times 10^6$-secure or over 8 *billion secure* with the DOWN assurance. HMBK with $m = 64$, $M = 32{,}768$, and $L = 64$ is $(n = 13{,}740, p = 10^{-12})$-secure without the DOWN assurance and can resist exposure of one secret each from about 14 *billion* ScPs. Note that when $k = mM$ is fixed, larger M implies that the KPS is optimized for larger n (as the choice $M \approx 2n/\log 2$ minimizes p). Thus, it is not surprising that lower m $(m = 32)$ performs better for larger n (for the same k).

Apart from being more efficient than MBK, HMBK also boasts a more graceful degradation of security[6] with increasing n. This property is unfortunately not evident in the first-order approximation $(\epsilon'(n) \approx \epsilon(3n/2))$. The values in the tables have been calculated using (13).

The additional overheads for HMBK (compared to MBK) is that in each of the m DOWN operations, a few repeated hash operations have to be performed. Note that for each of the m shared secrets one of the two entities has to hash

5. Or m KDCs choose one set of secrets each.

6. This property has also been observed for HARPS [35], which is a combination of random subset allocation and LM-KPS, just as HMBK is a combination of MBK and LM-KPS.

forward $L/3$ times on the average (the expected value of the difference between two randomly chosen hash depths between 1 and L is $L/3$). Thus, (on the average) $L/6$ additional hash computations are required in each of the m DOWN operations. It is important to note that the *number of DOWN operations* is still the same. In practice, the number of DOWN operations has a greater significance as each operation may call for kernel mode switching.

For many application scenarios, even 128 Mbytes per ScP (for example, ScPs plugged into PDAs that can use flash storage also plugged into PDAs) may not be unreasonable. For such scenarios, n can be increased from about 8,000 for HMBK (for $p < 10^{-20}$) to $n = 128,000$. The assurance in conjunction with the DOWN policy, namely, nk, increases from about 8 billion to 2 *trillion*. In other words, increasing storage by a factor 16 (from 8 Mbytes to 128 Mbytes—or k from 1 million to 16 million) improves the collusion resistance with DOWN assurance by a factor $16^2 = 256$ (from 8 billion to 2 trillion). Note that with large storage MBK/HMBK can be considered "reasonably secure" *even without* the DOWN assurance.

6 Conclusions

This paper introduced the DOWN policy that can substantially lower the complexity of circuitry required for countermeasures and improve the assurances offered by trustworthy computers like cryptographic coprocessors to protect their secrets. This is achieved by eliminating dependence on expensive and vulnerable multistep countermeasures. Thus, the DOWN policy can simultaneously lower the cost *and* improve reliability of cryptographic coprocessors. To achieve this, DOWN takes advantage of the ability to perform computations with fractional parts of secrets.

It was shown that many asymmetric cryptographic primitives lend themselves readily to DOWN implementations. Specifically, asymmetric schemes for which computations using private keys are restricted to multiplications and exponentiation lend themselves readily. Thus far, it appears that the DOWN policy may *not* be as easily extended for protecting private keys of pairing-based IBE schemes, or more generally, schemes for which operations with private keys include computation of modular *multiplicative inverse*. Furthermore, schemes for which there are no *constraints* on the choice of private keys (for example, ECC schemes, El Gamal, and variants) are preferable for use in conjunction with DOWN.

Even while the need to keep the complexity inside the trusted boundary has been very well recognized in the literature, it is implicitly assumed that ScPs should have the ability to support asymmetric cryptography. All commercially available secure coprocessors support various types of asymmetric encryption and signatures schemes. In emerging application scenarios, calling for very large scale deployments of inexpensive devices bound to low complexity ScPs, the use of asymmetric cryptographic primitives may not be feasible. It was argued that low-complexity ID-based schemes are very much desirable for many emerging applications scenarios.

Two novel ID-based KPS were proposed, which were explicitly designed to take a good advantage of the DOWN policy and demand very low-computational complexity inside the ScP. Furthermore, by making use of an increasingly inexpensive and abundant storage, the security of the proposed schemes can be increased to such extents that their "susceptibility to collusions" is of no practical consequence. The storage resources do not have to be protected and could be an external pluggable storage or even storage locations accessed over insecure networks. For supporting the meager amount of computations required, ScPs with a very low capability general purpose processor (which however supports some additional dedicated instructions for operations using the master secret and/or the hardware block cipher) and a dedicated AES block cipher, is more than adequate.

Our ongoing work indicates that DOWN assurances can also be employed to improve the security of the broadcast authentication scheme by Canetti et al. [36], where every node appends a large number of HMACs and any verifier can verify a subset of the appended HMACs. However, with additional constraints on the bandwidth for the appended HMACs, such schemes can only (thus far) realize linear improvements with storage (unlike schemes for pairwise authentication that can realize quadratic gains with storage). Some of our other ongoing work include 1) techniques for extending the DOWN policy to IBE schemes based on pairing, 2) investigation of the applicability of the DOWN policy for asymmetric schemes over Galois field $GF(2^n)$, and 3) investigation of the feasibility of an ID-based key predistribution infrastructure (KPI) and employing DOWN friendly symmetric key distribution schemes as an alternative to the certificates-based public key infrastructure (PKI) for securing large-scale ubiquitous computing networks.

Appendix A

A.1 Blom's SKGS

For a network size of $N \leq q$, SKGS employs a *public* $(n+1) \times N$ MDS generator matrix $\mathbf{G} = [\mathbf{g_0 g_1} \cdots \mathbf{g_N}]$ ($\mathbf{g_i}$s are column vectors of length $n+1$). The KDC chooses a $(n+1) \times (n+1)$ symmetric matrix $\mathbf{D}$ with $\binom{n+1}{2}$ independent values chosen randomly from $\mathbb{Z}_q$, where $q \geq N$. Node that A is provided with $k = n+1$ values (secrets) of the $\mathbf{d^A} = \mathbf{D}\mathbf{g_A}$. Two nodes A and B (with secrets $\mathbf{d^A}$ and $\mathbf{d^B}$, respectively) can calculate $K_{AB} = (\mathbf{d^A})^{\mathbf{T}}\mathbf{g_B} = (\mathbf{d^B})^{\mathbf{T}}\mathbf{g_A}$, which no other node can. As long as an attacker has access to secrets from n or less nodes, the attacker learns nothing about $\mathbf{D}$.

A.2 Group Addition in Elliptic Curves

For an ECC defined by $\mathcal{G} = \{(x,y)\} : y^2 \equiv x^3 + a_0 x + a_1 \bmod p$ addition of two points $P = (x_P, y_P), Q = (x_Q, y_Q) \in \mathcal{G}$ yields $P + Q = R = (x_R, y_R)$, where $x_p, y_P, x_Q, y_Q, x_R, y_R \in \mathbb{Z}_p$, and

$$\begin{aligned} x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - X_R) - y_P \\ \lambda &= \begin{cases} (y_Q - y_P)(x_P - x_Q)^{-1} & P \neq Q \\ (3x_P^2 + a_0)(2y_P)^{-1} & P = Q. \end{cases} \end{aligned} \tag{14}$$

A.3 MBK

Note that the secret $K_i(a_i, b_i)$ is not unique to nodes A and B. There is a probability that some node C (in the attacker's pool of compromised nodes) may also have the secret $K_i(a_i, b_i)$, *if*

$f(C,i) = c_i \in \{a_i, b_i\}$. As the output of the one-way function $f()$ is uniformly distributed between 1 and M, the probability that a node C has an access to $K_i(a_i, b_i)$ is

$$\Pr\{c_i \in \{a_i, b_i\}\} = \gamma = \frac{2M-1}{M^2} \approx 2/M, \tag{15}$$

where the approximation holds for large M. The probability that an attacker who has access to all secrets from n nodes (that does not include A and B) *cannot* discover $S_i = K_i(a_i, b_i)$ is

$$\epsilon(n) = (1-\gamma)^n. \tag{16}$$

Thus, the probability $p(n)$ that the attacker *can* compromise *all* m such that the S_is shared by A and B (and, thus, determine K_{AB}) is

$$\begin{aligned} p(n) &= (1-\epsilon(n))^m = (1-(1-\gamma)^n)^m \\ &= \approx (1-\mathrm{e}^{-\mathrm{n}\gamma})^{\mathrm{m}} \approx (1-\mathrm{e}^{\frac{-2\mathrm{n}}{\mathrm{M}}})^{\mathrm{m}}, \end{aligned} \tag{17}$$

where the first approximation follows from the well-known identity $(1-1/x)^x \approx \mathrm{e}^{-1}$ for large x.

A.3.1 Choice of MBK Parameters

We can reqwrite (17) as

$$k = \frac{2n \log(1/p)}{-x \log(1-\mathrm{e}^{-\mathrm{x}})}, \quad \text{where } x = \frac{2n}{M}. \tag{18}$$

Minimizing k calls for maximizing $-x\log(1-\mathrm{e}^{-x})$, which occurs when $\mathrm{e}^{-x} = 1/2$, or $x = \log(2)$. Now, substituting $\mathrm{e}^{\frac{-2n}{M}} = 1/2$ in (17), we have $p(n) = \frac{1}{2^m}$, $m = \log_2(1/p) = \log(1/p)/\log(2)$, $M = 2n/\log(2)$, and

$$k = mM = \frac{2n\log(1/p)}{(\log(2))^2}. \tag{19}$$

If we desire to reduce m by a factor a (to $m' = \frac{m}{a}$), we need $\mathrm{e}^{\frac{-2n}{M}} = 1 - 1/2^a$ to achieve the desired $p(n)$. Thus, we need to choose increase $M' = 2n/\log(1-1/2^a)$ and, thus,

$$k' = m'M' = k\frac{\log(1-1/2)}{a\log(1-1/2^a)}. \tag{20}$$

A.4 MBK versus Subset Allocation Schemes

For subset allocation schemes [35]

$$p(n) = (1-\xi(1-\xi)^n)^k, \quad \xi = k/P. \tag{21}$$

The choice of $\xi^* = 1/(n+1) \approx 1/n$ (for large n) maximizes $\xi(1-\xi)^n$ and thus minimizes the storage k. Making use of the identities $(1+1/x)^x \approx \mathrm{e}$ for large x and $\log(1-y) \approx -y$ for $y \ll 1$, we can easily see that $k \approx n\mathrm{e}\log(1/p)$. Thus, to achieve the same $p(n)$, RAS schemes require k less by a factor $\frac{2}{\mathrm{e}\log^2 2} \approx 1.53$ compared to MBK (see (19)).

The value m can be reduced by k (and reducing ξ by a larger factor) such that $m = \xi k$ is reduced. It is easy to see that m cannot be reduced below $\log(1/p)$ even if $k \to \infty$. For very large k (and small $m = \xi k$), we have $\xi \to 0$. More specifically, for large k, we have $\xi \ll 1/n$. As $\xi(1-\xi)^n \approx \xi(1-n\xi) \approx \xi$ for $\xi \ll 1/n$, we can rewrite (21) as

$$\begin{aligned} p(n) &= (1-\xi(1-\xi)^n)^k = (1-\xi)^{m/\xi} \\ &\approx \left((1-\xi)^{1/\xi}\right)^m \approx \mathrm{e}^{-\mathrm{m}}. \end{aligned} \tag{22}$$

In other words, even when $k \to \infty$, $m = \xi k$ *cannot* be reduced below $\log(1/p)$ to achieve (n,p)-security.

A.5 HMBK

In this case, for each of the m indices, the shared secrets S_i, $1 \le i \le m$ between A and B is at a hash depth $\max(\bar{a}_i, \bar{b}_i)$. Unlike MBK where it was sufficient for some node C with $f(C,i) = c_i \in \{a_i, b_i\}$ to determine the secret corresponding to index i, for HMBK, an additional condition has to be satisfied. The condition is $\bar{c}_i \le \max(\bar{a}_i, \bar{b}_i)$.

For any i, if we define $p_l = \Pr\{\max(\bar{a}_i, \bar{b}_i) = l\} = \frac{2l-1}{L^2}$, $1 \le l \le L$, and $q_l = \Pr\{c_i \le l\} = l/L$, the probability ϵ' that any $S_i = K_i^{\max(\bar{a}_i, \bar{b}_i)}(a_i, b_i)$ is safe from an attacker (who has exposed all mM secrets from n nodes) is

$$\epsilon' = \sum_{j=1}^{L} p_l(1-q_l\gamma)^n = \sum_{j=1}^{L} \frac{2l-1}{L^2}\left(1-\frac{l\gamma}{L}\right)^n, \tag{23}$$

compared to $\epsilon(n) = (1-\gamma)^n$ for MBK (see (16)). As a first-order approximation, it is easy to see that the expected value of l is $\bar{l} = \frac{2L}{3}$. Thus

$$\epsilon'(n) \approx \left(1-\frac{2\gamma}{3}\right)^n \approx (1-\gamma)^{3n/2} = \epsilon(3n/2). \tag{24}$$

References

[1] M. Kwiatkowska and V. Sassone, "Science for Global Ubiquitous Computing," *Grand Challenges in Computing (Research),* T. Hoare and R. Milner, eds., 2004.
[2] S.W. Smith, *Trusted Computing Platforms: Design and Applications.* Springer, 2005.
[3] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, "Cryptographic Processors—A Survey," Computer Laboratory Technical Report UCAM-CL-TR-641, Univ. of Cambridge, Aug. 2005.
[4] R. Anderson and M. Kahn, "Tamper Resistance—A Cautionary Note," *Proc. Second Usenix Workshop Electronic Commerce,* pp. 1-11, 1996.
[5] S.W. Smith and S. Weingart, "Building a High-Performance Programmable Secure Coprocessor," *Computer Networks,* vol. 31, pp. 831-860, 1999.
[6] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM,* vol. 21, no. 12, Dec. 1978.
[7] D. Lie, C.A. Thekkath, and M. Horowitz, "Implementing an Untrusted Operating System on Trusted Hardware," *Proc. 19th ACM Symp. Operating Systems Principles,* pp. 178-192, Oct. 2003.
[8] P.C. van Oorschot, A. Somayaji, and G. Wurster, "Hardware-Assisted Circumvention of Self-Hashing Software Tamper Resistance," *IEEE Trans. Dependable and Secure Computing,* Apr. 2005.
[9] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Tamper Proof Security: Theoretical Foundations for Security Against Hardware Tampering," *Proc. Theory of Cryptography Conf.,* Feb. 2004.
[10] S. Weingart, "Physical Security for the mABYSS System," *IEEE Security and Privacy,* pp. 38-51, 1987.
[11] S. White, S. Weingart, W. Arnold, and E. Palmer, "Introduction to the Citadel Architecture: Security in Physically Exposed Environments," Technical Report RC16672, IBM Thomas J. Watson Research Center, Mar. 1991.
[12] J.D. Tygar and 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.

[13] B. Chen and R. Morris, "Certifying Program Execution with Secure Processors," *Proc. Ninth Workshop Hot Topics in Operating Systems,* May 2003.

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

[15] M.G. Karpovsky, K. Kulikowski, and A. Taubin, "Robust Protection against Fault-Injection Attacks of Smart Cards Implementing the Advanced Encryption Standard," *Proc. Int'l Conf. Dependable Systems and Networks,* July 2004.

[16] P. Kocher, "Differential Power Analysis," *Advances in Cryptology—Proc. Ann. Int'l Cryptology Conf.,* pp. 388-397, 1999.

[17] C. Aumeller, P. Bier, W. Fischer, P. Hofreiter, and J.P. Seifert, "Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures," Cryptology ePrint Archive, http://eprint.iacr.org/2002/073.pdf, 2002.

[18] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving Smart Card Security Using Self-timed Circuits," *Proc. Eighth Int'l Symp. Advanced Research in Asynchronous Circuits and Systems,* 2002.

[19] O. Kommerling and M. Kuhn, "Design Principles for Tamper-Resistant Smart-Card Processors," *Proc. Usenix Workshop Smartcard Technology,* pp. 9-20, 1999.

[20] Y. Ishai, A. Sahai, and D. Wagner, "Private Circuits: Securing Hardware Against Probing Attacks," *Advances in Cryptology—Proc. Ann. Int'l Cryptology Conf.,* Aug. 2003.

[21] J.P. McGregor and R.B. Lee, "Protecting Cryptographic Keys and Computations via Virtual Secure Coprocessing," *ACM SIGARCH Computer Architecture News Archive,* vol. 33, no. 1, Mar. 2005.

[22] P. Gutman, "Secure Deletion of Data from Magnetic and Solid-State Memory," *Proc. Sixth Usenix Security Symp.,* July 1996.

[23] R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Proc. Int'l Workshop Security Protocols,* Apr. 1997.

[24] D.R. Stinson, *Cryptography, Theory and Practice,* second ed. Chapman and Hall CRC, 2002.

[25] C. Couvreur and J.-J. Quisquater, "Fast Decipherment Algorithm for RSA Public-Key Cryptosystem," *Electronics Letters,* vol. 18, no. 21, pp. 905-907, 1982.

[26] D. Clarke, J.-E. Elien, M. Fredette, A. Marcos, and R.L. Rivest, "Certificate Chain Discovery in SPKI/SDSI," *J. Computer Security,* vol. 9, no. 4, pp. 285-322, 2001.

[27] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Advances in Cryptology—Proc. Ann. Int'l Cryptology Conf.,* vol. 196, pp. 47-53, 1984.

[28] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Advances in Cryptology—Proc. Ann. Int'l Cryptology Conf.,* pp. 213-229, 2001.

[29] R. Dutta, R. Barua, and P. Sarkar, "Pairing-Based Cryptography: A Survey," Report 2004/064, Cryptology ePrint Archive, 2004.

[30] T. Leighton and S. Micali, "Secret-Key Agreement without Public-Key Cryptography," *Advances in Cryptology—Proc. Ann. Int'l Cryptology Conf.,* pp. 456-479, 1994.

[31] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques, Advances in Cryptology,* pp. 335-338, 1984.

[32] T. Matsumoto and H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem," *Proc. Ann. Int'l Cryptology Conf., Advances in Cryptology,* pp. 185-193, 1987.

[33] R. Di Pietro, L.V. Mancini, and A. Mei, "Random Key Assignment for Secure Wireless Sensor Networks," *Proc. ACM Workshop Security of Ad Hoc and Sensor Networks,* Oct. 2003.

[34] M. Ramkumar, N. Memon, and R. Simha, "Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks," *Proc. Global Telecomm. Conf.,* 2003.

[35] M. Ramkumar and N. Memon, "An Efficient Random Key Predistribution Scheme for MANET Security," *IEEE J. Selected Areas of Comm.,* Mar. 2005.

[36] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. INFOCOMM '99,* 1999.

Mahalingam Ramkumar received the PhD degree in electrical engineering from the New Jersey Institute of Technology in 2000. He has been an assistant professor in the Department of Computer Science and Engineering, Mississippi State University, since August 2003. His research interests include Cryptography, key distribution, ad hoc network security, and data hiding. He is a member of the IEEE.