Lemonade from Lemons: Harnessing Device Wearout to Create Limited-Use Security Architectures

Zhaoxia Deng
zhaoxia@cs.ucsb.edu
Department of Computer Science,
University of California, Santa Barbara

Ariel Feldman Stuart A. Kurtz Frederic T. Chong
{arielfeldman,stuart,chong}@cs.uchicago.edu
Department of Computer Science,
University of Chicago

ABSTRACT

Most architectures are designed to mitigate the usually undesirable phenomenon of device wearout. We take a contrarian view and harness this phenomenon to create hardware security mechanisms that resist attacks by statistically enforcing an upper bound on hardware uses, and consequently attacks. For example, let us assume that a user may log into a smartphone a maximum of 50 times a day for 5 years, resulting in approximately 91,250 legitimate uses. If we assume at least 8-character passwords and we require login (and retrieval of the storage decryption key) to traverse hardware that wears out in 91,250 uses, then an adversary has a negligible chance of successful brute-force attack before the hardware wears out, even assuming real-world password cracking by professionals. M-way replication of our hardware and periodic re-encryption of storage can increase the daily usage bound by a factor of M.

The key challenge is to achieve practical statistical bounds on both minimum and maximum uses for an architecture, given that individual devices can vary widely in wearout characteristics. We introduce techniques for architecturally controlling these bounds and perform a design space exploration for three use cases: a limited-use connection, a limited-use targeting system and one-time pads. These techniques include decision trees, parallel structures, Shamir's secret-sharing mechanism, Reed-Solomon codes, and module replication. We explore the cost in area, energy and latency of using these techniques to achieve system-level usage targets given device-level wearout distributions. With redundant encoding, for example, we can improve exponential sensitivity to device lifetime variation to linear sensitivity, reducing the total number of NEMS devices by 4 orders of magnitude to about 0.8 million for limited-use connections (compared with 4 billion if without redundant encoding).

CCS CONCEPTS

• **Security and privacy** → **Hardware security implementation**;

KEYWORDS

Degradation-based security measures, hardware security architectures, NEMS

ISCA '17, June 24-28, 2017, Toronto, ON, Canada

ACM ISBN 978-1-4503-4892-8/17/06…$15.00
https://doi.org/10.1145/3079856.3080226

ACM Reference format:
Zhaoxia Deng, Ariel Feldman, Stuart A. Kurtz, and Frederic T. Chong. 2017. Lemonade from Lemons: Harnessing Device Wearout to Create Limited-Use Security Architectures . In *Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017,* 14 pages.
https://doi.org/10.1145/3079856.3080226

1 INTRODUCTION

Wearout is not a new problem in computer architecture. Flash memories are the most well-known memory technologies with the wearout problem, and people have been working on efficient solutions to it for decades. In the big data era, the problem is even worse with continuously increasing density and capacity demand for memory. The lifetime of a flash cell dropped from 10,000 times to 2,000 times when the cell dimension scales down from 50nm to 20nm [41]. Moreover, ITRS [13] envisions many new technologies such as non-volatile memories (NVM), nanoelectromechanics (NEMS), and molecular devices, to solve the power-consumption problem in CMOS so as to sustain Moore's law. However, the wearout problem also exists in these technologies [44, 49, 67, 68]. The down-scaling from MEMS to NEMS, for instance, results in exponential degradation of device reliability [34, 37]. Representative NEMS contact switches can only work for one cycle to several thousand cycles without failures [15, 23, 37, 53]. As a result, existing research on these emerging technologies has been focused on how to extend the lifetime before they can really be applied.

Meanwhile, we take a step back and find that wearout can help build purposely limited-use security architectures. In applications that abusive accesses or adversarial accesses are not desired, wearout can provide strong security by automatically destroying the device and hence protects any secure information in it. For example, forward secrecy encryption [20, 26] in any public key systems [3, 40] (eg. the encryption of e-mail archives) requires a one-time key for the encryption of each message so that the compromise of a single private key does not compromise all the past messages. Traditionally, the one-time access of the keys is not enforced so the system still cannot defend against reusing or stealthy replications of the keys. Taking advantage of wearout, we can store the keys in a security architecture that wears out exactly after one access so that the one-time usage of keys is physically enforced and the security of messages will not be compromised.

However, taking advantage of wearout to create hardware security mechanisms is challenging. The problems we face in designing the security architectures include:

- How do we design for system-level minimum and maximum usage in the face of probabilistic wearout behaviors of each device and process variations among devices?

- Depending on security goals and usage targets, how should we adjust the parameters in our design to minimize area and energy cost?
- How do we balance the fabrication cost of more consistent devices (in terms of wearout) with the area cost of architectural techniques to achieve consistency (eg. redundancy and encoding)?

In this paper, we propose a methodology to create security architectures by harnessing device wearout. Our contributions are as follows:

- We use the two-parameter Weibull distribution to model the time to failure of each device. The two parameters in the model can be used to characterize different kinds of wearout devices. Manufacturing and process variations among individual devices are accommodated by introducing more variations into the distribution.
- Based on the probabilistic wearout model, we provide statistical guarantees on system-level usage bounds by designing application-dependent architectures and exploiting redundant encoding techniques. The system-level usage bounds ensure both reliability for legitimate users and security to defend against brute-force attackers.
- Extensive engineering space exploration has been performed to study the trade-offs among target access bounds, area cost, fabrication cost, etc.

In the following sections, we first introduce the wearout devices used in our security architectures and describe the probabilistic wearout model in Section 2. Then we talk about the threat model in Section 3 for three use cases of hardware security mechanisms: a limited-use connection, a limited-use targeting system and one-time pads, discussed in Section 4, 5 and 6, respectively. Note also, we talk about the limitations of the degradation-based security measures in Section 7. Then we discuss the literature of hardware security in Section 8. Finally, we conclude our work in Section 9.

2 DEVICE WEAROUT MODEL

2.1 NEMS contact switches

NEMS contact switches exhibit promising properties such as nano-scale physical dimensions, near zero OFF-state leakage, and large ON/OFF ratios. Representative NEMS contact switches are composed of a movable active element and an opposing electrode which interact by both electrical and mechanical forces to open and close the switch. When a pull-in voltage is applied to the active element, electrostatic forces deform the active element towards the opposing electrode so as to close the switch. After the voltage is removed, elastic forces pull the active element away from the opposing electrode so that the switch is open again.

Among other technologies, NEMS are advantageous in building limited-use security architectures because they have relatively tight wearout bounds and they are insensitive to harsh environments [37] including radiation, temperature, external electric fields, etc. Most fabricated NEMS switches can work properly for only one to several thousand cycles [15, 23, 37, 53]. Recently, NEMS lifetime can be extended up to millions or billions of cycles while at the cost of scaling up the physical dimensions [24, 28, 57]. The wearout characteristics of NEMS switches are highly dependent on the materials and

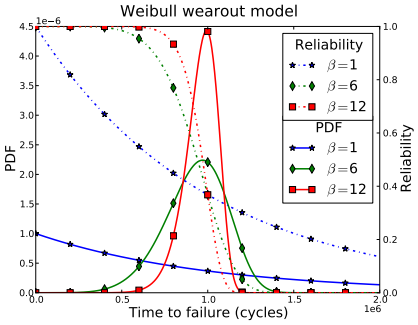


Figure 1: Weibull wearout model with different shape parameters. The red lines ($\beta = 12$) show the lifetime plots of MEMS devices [55] with geometrical variations.

structures employed. Generally speaking, any kind of electrical and mechanical aging, adhesion, fracture or burnout in the active element or electrode are potentially responsible for the failures of NEMS switches. For example, in [53], graphene-based NEMS switches were recorded to work over 500 cycles and then failed because the over-bent graphene was unable to recover. [23] reported that NEMS switches with silicon carbide nanowires can switch for tens of cycles before the nanowire was stuck to the electrode. In [35], the silicon carbide cantilevered NEMS switches failed after billions of cycles because of fracture at room temperature and melting at 500 °C.

Furthermore, due to their insensitivity to harsh environment, NEMS switches can help defend against attacks by varying the dynamic environment. For example, it is hard for attackers to extend the lifetime of NEMS switches by controlling the operating temperatures, especially these made of high temperature friendly materials such as SiC. Poly-SiC NEMS switches [28] were recorded to operate properly for at least $10^5 \sim 10^6$ cycles without failures at 500 °C , comparable to those at 25 °C. [35] has shown that SiC NEMS switches can operate more than 21 billion cycles at 25 °C while more than 2 billion cycles at 500 °C. However, failures at 25 °C were characterized by fracture while failures at 500 °C were probably caused by melting. For security architectures, we assume the lifetime at room temperature (25 °C) as the device wearout bound. More failures at extremely high temperatures will destroy the device faster, but will not compromise the secret information in them. At extremely low temperatures (eg. after freezing), it is hard to extend the device lifetime either because failures from fracture cannot be avoided. These features have enabled NEMS switches to be applied to security or harsh environment applications, eg. one-time-programmable FPGA interconnects in [29].

2.2 Probabilistic wearout model

In the reliability literature [56, 60], Weibull distributions have been commonly used to model the failure distributions of electronic devices [38], such as the breakdown of gate oxides in CMOS [50]. Similar models can also be applied to micro-/nano- scale devices [8]. It has been shown that the Weibull distribution can accurately fit fracture-strength data of emerging materials for NEMS/MEMS [11, 22], fracture-test data of cantilever beam MEMS [18], and tensile-strength data sets of carbon nanotubes [10].

Hence, we take the general two-parameter Weibull distribution to model the failure distribution of NEMS switches in this paper. Assume that x represents the time to failure. Then the probability density function (PDF) of the time to failure is:

$$f(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1} e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \quad (1)$$

and the cumulative density function (CDF) is:

$$F(x) = 1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \quad (2)$$

Based on the CDF, the reliabity function is derived as follows:

$$R(x) = 1 - F(x) = e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \quad (3)$$

In all of the above equations, α is the scale parameter and β is the shape parameter. The two parameters can be estimated by fitting the lifetime data of a large population of similar devices. α approximates to the mean time to failure, and β mainly determines the variation of reliability degradation among these devices.

Figure 1 shows the failure PDF (Equation 1) and reliability function (Equation 3) with different βs. The variation of α will only change the axis scales. β is usually larger (sharper peaks in the PDF) with more homogeneous devices in which the wearout happens more consistently.

Since the manufacturing processes are less mature at nano-scales, we accommodate the process variations in αs and βs. Typically, process variations will result in lower βs. Trevor S. Slack et al. [55] has simulated Weibull lifetime models of MEMS devices considering geometrical variations, material variations in elastic modulus, resistance stress, etc. According to their simulation results, αs and βs are 2.6 million cycles and 12.94 with only geometrical variations, 2.2 million cycles and 7.2 with material elasticity variations, 1.8 million cycles and 8.58 with material resistance variations. We will experiment with various αs and βs for an extensive engineering space exploration in later sections.

3 THREAT MODEL

Since our target devices are often mobile devices, we assume the attacker has physical access to the device. In the cases of the limited-use connection and targeting system, we want to defend against brute-force attacks to decrypt the device by physically limiting the number of accesses to the storage decryption key. The cracking approaches we target are professional attacks that can exploit the nonuniform guessability in real-world passwords [62], as discussed in Section 4.1.

In the case of one-time pads, we want to defend against stealthy replications of device (the archaically named "evil maid attack"). The attackers may clone the one-time pads and make two copies, one copy to replace the receiver's original one and another copy for themselves to break the encryption in the future message transmission between the sender and receiver. Our secure architectures will resist cloning by making it difficult for attackers to ever read the entire contents of the device memory.

In this paper we assume the chip fabrication is trusted. And we leave as future work techniques to allow secure, one-time programming of our devices by end users. We assume the secret information is one-time programmed in the device memory at fabrication time and end-users will only need read operations through the NEMS network.

4 A LIMITED-USE CONNECTION

Apple iOS has developed many security features with integrated secure software and hardware support. However, to keep the devices easy to use, the most straightforward way to protect the devices is to use a passcode that is configurable by users. To protect the passcode from brute-force attacks, iOS provides several mechanisms [4]: 1) it automatically wipes out all data on the device if someone has consecutively failed 10 times in unlocking the device. 2) it has incremental time delays after each incorrect passcode attempt.

However, all these guarding mechanisms are recently reported to be easily bypassed. A company called MDSec [2] managed to bypass the internal counter in iPhone by cutting its power right before the counter is incremented, while still getting the passcode validation result. Therefore, the counter never gets updated and the attacker can break the passcode as fast as the hardware can support. [54] demonstrated similar attack to the counter through NAND mirroring. An iPhone 5c was forced to power down and recover from a backup NAND memory to restore the previous state once every a few passcode attempts, which enabled unlimited attempts to crack the passcode. Another hacking case exploited firmware updates [1]. iPhone can launch firmware updates automatically without the passcode, which means that the guarding mechanisms are disabled or the counter is disabled during the updates. Then brute-force attacks can easily succeed, especially with real-world biased passcodes.

Although some vulnerabilities mentioned above have been patched, the real problem with software solutions is that they can not prevent unknown vulnerabilities. Therefore, we propose to exploit NEMS switches to build a limited-use connection that can physically limit the number of accesses to the storage decryption key. The right passcode and the storage decryption key are needed to successfully decrypt the device storage and thus validate the passcode. We enforce a traversal to the limited-use connection before each read of the storage encryption key. After each passcode attempt, the failure probability of the connection increments. After a certain number of accesses, the connection will wear out automatically and the smartphone will be locked forever. Compared with security patches, our solution provides strong hardware enforced security, which cannot be compromised even under government's intervention.

In Section 4.1, we talk about the design principles and design options for the limited-use connection using NEMS switches. And we discuss system integration issues with NEMS switches in Section 4.2. Finally, we explore the engineering space given the design options in Section 4.3.

4.1 Using wearout to build a limited-use connection

The design of the limited-use connection needs to follow two principles:

- The connection should work reliably for all legitimate accesses during the smartphone's lifetime.
- The connection should wear out before attackers have high probabilities to guess the passcode.

As an example, we calculated the legitimate access bound (LAB) for a smartphone approximately as follows:

$$LAB = 5 * 365 * 50 = 91,250 \quad (4)$$

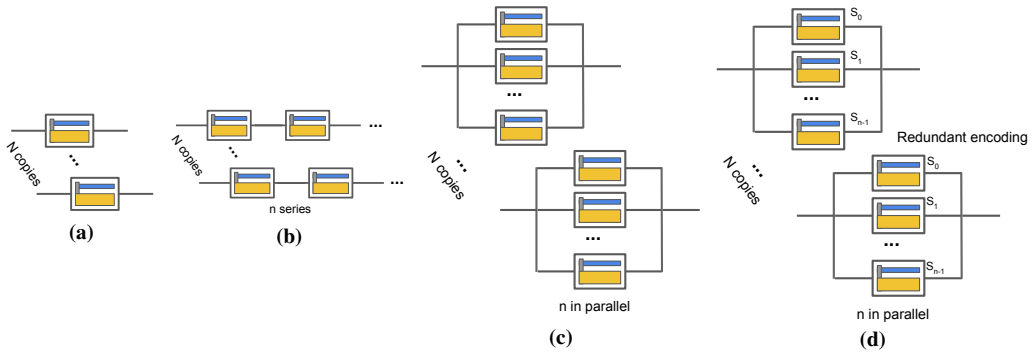


Figure 2: Design options for the limited-use connection using NEMS switches. Fig 2a uses N copies of single NEMS switches. Fig 2b uses N copies of n NEMS switches in series. Fig 2c uses N copies of n NEMS switches in parallel. Fig 2d uses same parallel structures but with redundant encoding.

If the real LAB is several times larger than that, we provide M-way replication of our entire architecture to scale the LAB by a factor of M, as described in Section 4.1.5.

The main challenge in designing the security architectures is how to control the system-level reliability degradation window $[t_1, t_2]$, as indicated in the two principles. The lower bound t_1 should be greater than the LAB while the upper bound t_2 should be less than the minimum guesses needed to crack any passwords. According to Blase Ur et al.'s work of measuring real-world password guessability [62], professional attackers usually try passwords in the order of empirical popularity. For 8-character passwords including characters from all different classes (lowercase letters, uppercase letters, numbers, and special characters), only a few very popular passwords can be guessed within 91,250 attempts. The guessing probability increases to 1% and 2% with 100,000 and 200,000 attempts, respectively.

In the following sections, we first attempt to design security architectures that wear out as quickly as possible after the LAB. We focus on seeking for architectural techniques that can help control the degradation window. Then we extend the upper bound to 100,000 and 200,000 accesses if the software helps reject the most popular 1% and 2% passwords, discussed in Section 4.3.

According to the design principles, we consider four possible design options, as illustrated in Figure 2. These design options are explained in detail as follows.

4.1.1 N copies of single NEMS switches. With a single NEMS switch, it is difficult to meet the system-level demand of minimum and maximum accesses. On one hand, the empirical lifetime of a single NEMS switch is usually not as long as the LAB. On the other hand, even if there exists such a NEMS switch, the degradation window expands millions of cycles, as shown in Figure 1. Thus, we consider using N copies of NEMS switches, which can divide the system-level access bounds by a factor of N. Then the design principles for each copy are adjusted as follows:

- Each copy should work reliably for $\frac{\text{LAB}}{N}$ accesses.
- Each copy should wear out before $\frac{\text{LAB}}{N}+1$ accesses.

Although the second requirement still requires small degradation windows, the first requirement scales the mean value down to $\frac{\text{LAB}}{N}$. This down-scaling helps create a small degradation window, as shown in Figure 3a.

However, when α is very small, the lifetime of the device is expected to be very small. The manufacturing and process variability is probably hard to control for such brittle and fragile devices. We will discuss more architectural options next to avoid such challenges in fabrication while still satisfying both design requirements.

4.1.2 N copies of n NEMS switches in series. Instead of looking for NEMS switches that can fail extremely fast, we consider chaining NEMS switches in series to accelerate the wearout, as in Figure 2b. In the chaining architecture, if any single NEMS switch in the chain fails, the whole chain fails. Unfortunately, we found that increasing the number of NEMS switches in the chain has no significant impact on the failure rate.

Assume we have n NEMS switches in series. Then the reliability of the chain is:

$$R(x) = \left(e^{-\left(\frac{x}{\alpha}\right)^{\beta}}\right)^{n} = e^{-n\left(\frac{x}{\alpha}\right)^{\beta}} \tag{5}$$

Compared with the reliability function for a single device in Equation 3, n devices with α in series are equivalent to a single device with $\frac{\alpha}{n^{(1/\beta)}}$. Let the denominator equals y: $y = n^{(1/\beta)}$. Then we get: $n = y^{\beta}$. If we want to increase y to scale down α as in Figure 3a, then n should increase to y^{12} in each copy with $\beta = 12$. To avoid the explosion of NEMS switches, we discard this option in the following discussion.

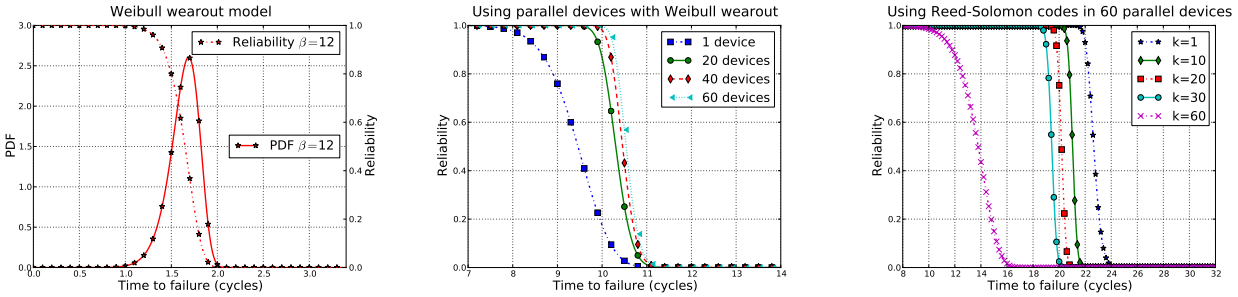
4.1.3 N copies of NEMS switches in parallel. Here we propose another technique to control the degradation window, which is exploiting parallel structures to improve the system reliability before all devices wear out, as shown in Figure 2c. Assuming N copies of parallel structures, the requirements for each copy are as follows:

- At least one NEMS switch in each copy should work reliably for $\frac{\text{LAB}}{N}$ accesses.
- All NEMS switches in each copy should wear out before $\frac{\text{LAB}}{N}+1$ accesses.

Assume each copy has n NEMS switches in parallel. The reliability of this parallel structure is :

$$R(x) = 1 - \left(1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}}\right)^{n} \tag{6}$$

The redundancy in the parallel structure provides error tolerance so that the high reliability threshold is pushed toward the degradation edge, as shown in Figure 3b. With 98% reliability, the parallel



(a) By scaling α down, we get a degradation window within 1. Given $\alpha = 1.7$ and $\beta = 12$, the reliability of each device is close to 1 when $t = 1$, but close to 0 when $t = 2$.

(b) Using parallel NEMS switches to push the high reliability threshold toward the degradation edge. $\alpha = 9.3$ and $\beta = 12$ for all NEMS switches.

(c) Using redundant encoding in parallel structures to accelerate degradation. $\alpha = 20$ and $\beta = 12$ for all NEMS switches.

Figure 3: Different techniques to control the hardware degradation window

structure with 40 devices can work for the 10th access. With only 2.2% probability the parallel structure will continue working for the 11th access. In this design option, the total number of devices may increase, from 91,250 to 365,000 ($40 * (91,250/10) = 365,000$) approximately, but the mean time to failure of NEMS switches is relaxed from one to about ten cycles.

4.1.4 Parallel NEMS switches with redundant encoding. In this section, we introduce Shamir's secret sharing mechanism and redundant encoding techniques to further speed up the degradation of the limited-use connection. Instead of using highly redundant and reliable 1-out-of-n parallel structures, we require at least k NEMS switches working in a parallel structure in order to decrypt the device storage, which results in a k-out-of-n parallel structure. Increasing k to some extent tightens the wearout bounds of each parallel structure so that the connection wears out faster. The challenging part, however, is that the k-out-of-n parallel structure should provide reliable connection when k or more NEMS switches work properly but wear out quickly when only $k-1$ or less NEMS switches do the same.

To enforce that, we exploit Shamir's secret sharing mechanism and redundant encoding techniques. The general idea is the following: We encode the storage decryption key into n components and spread them in n read-destructive storage connected by the NEMS switches in a parallel structure. The encoding mechanism enforces that at least k components are needed to successfully get the key while no knowledge about the key will be revealed with less than k components.

Shamir's secret-sharing scheme. Shamir's secret-sharing scheme [51] constructs fast degradation codes. One of its classical scenarios is the secret sharing among many people. On one hand, the scheme allows efficient and reliable secret sharing if at least k out of n people authorize accesses to the secret. On the other hand, the scheme prevents leaking any information about the secret if there is only authorization from $k-1$ people or less. The scheme is also called a (k,n) threshold scheme and its reliability degrades immediately at $k-1$. Unlike classical secret-sharing scenarios that tolerate partial errors for more efficient sharing (authorizing access with the majority of people's permission [51] or matching interests with the majority of attributes [31]), our security architectures need to tolerate erasures from device failures.

The encoding and decoding in Shamir's secret sharing scheme is based on polynomial construction and interpolation. The insight is that given k independent points on a 2D plane, there is one and only one polynomial of degree $k-1$ that passes through all the k points. The encoding algorithm involves constructing such a polynomial of degree $k-1$ with the secret hidden in the coefficients:

$$q(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1} \quad (7)$$

With this polynomial, we encode the secret into n points, for instance, $q(1)$, ..., $q(n)$. The n points are then distributed to n people (or n devices) in the security application. Given any k points, all coefficients can be easily computed by interpolation so as to recover the secret. With $k-1$ points or less, however, no information about the secret can be inferred.

Error correction codes with fast degradation. Reed-Solomon codes [31, 39] are the error correction version of Shamir's secret-sharing scheme and they are commonly used in the error correction of large amounts of data in devices such as flash disks, CDs and DVDs. Theoretically, other linear codes could also construct similar (k,n) threshold secret sharing schemes, but it is hard to reason about the security because of the hardness of approximating the minimum distance of any linear code [21].

Figure 3c shows the reliability of a parallel structure with Reed-solomon codes. With redundant encoding, the reliability of this security architecture becomes:

$$R(x) = \sum_{i=k}^{n} \binom{n}{i} \left(e^{-\left(\frac{x}{\alpha}\right)^{\beta}}\right)^{i} \left(1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}}\right)^{n-i} \quad (8)$$

We use 60 NEMS switches in the parallel structure and can relax the wearout bound for each NEMS switch to around 20 cycles. With $k = 1$, the degradation window size is about 2, while with $k = 30$, the degradation window size reduces to about 1, as shown in Figure 3c. With $k = 30$, the parallel structure provides 92% reliability for the 20th access while only 2% probability for the 21st access. However, when k is close to the total number of parallel devices in the structure, the reliability degrades very early that the degradation window is stretched out again.

Assisted with Reed-Solomon codes, we are able to build the limited-use connection with approximately the same number of devices ($60 * (91,250/20) = 365,000$) compared with the parallel structure without encoding, but we can further relax the wearout

bound to 20 cycles. The overhead, however, includes the encoding/decoding complexity and extra storage for the component keys.

4.1.5 M-Way Replication of Modules. A legitimate usage factor of 50 times per day is potentially low for some users, so we propose to support increasing this usage by a factor of M by replicating our entire structure M times. The replicated modules must be used serially and each must employ a new password. In this way, an attacker can only attack each module separately to its upper access bound, but the user can achieve usage that is the sum of the lower access bounds of all M modules.

The cost of this scheme, however, is that a new password must be chosen when migrating from one module to another and the storage encryption key must be re-encrypted with the new password. For example, if we wish to increase usage from 50 times to 500 times per day, we use a 10-way replication factor, which implies that the user must choose a new password and re-encrypt storage every 6 months during our target 5-year lifetime of the smartphone.

4.2 System integration

To build the limited-use connection in practical mobile devices, we need to integrate the NEMS network with conventional CMOS. Here we show that the CMOS-NEMS integration is feasible in the manufacturing process and the CMOS-NEMS interface does not compromise the security of the device storage.

Manufacturability. NEMS devices are CMOS-compatible: they do not require exotic materials or fabrication flow. In the literature, there have been integrated CMOS-NEMS circuits for leakage-control and power-gating [16, 30]. A possible integration solution, for example, is to have NEMS and CMOS circuits in differet layers of the chip with a sandwiched metal layer in between [14].

Security. We bury the secret key many layers below the surface of the chip and only connect that secret key to the surface through NEMS devices. Although there are CMOS-NEMS connections both from the surface to the NEMS network and from the NEMS network to the deeply buried secret key, the latter connections are difficult to access and thus provide a level of physical security. Circumventing the surface connections does not help the adversary access the secret.

4.3 Engineering space exploration

In this section, we talk about the engineering space exploration for the limited-use connection. Without loss of generality, the discussion will be focused on architectural options using N copies of parallel structures with or without encoding. As discussed earlier, our goal is to guarantee the system-level access bounds to protect the passcode from real-world brute-force attacks. Here we list several parameters in the engineering space of the limited-use connection: device wearout characteristics, redundant encoding, and system-level access bounds. Next, we will discuss these parameters and their trade-offs in fabrication cost, area cost, encoding complexity, etc. The experiments are based on numerical simulations with different engineering options.

4.3.1 Device wearout characteristics. We first choose different αs and βs to characterize various kinds of NEMS switches. Since α approximates to the average lifetime of devices in the Weibull model, we set α according to the lifetime of representative NEMS switches as listed in [37], ranging from 1 cycle to 1000 cycles. In the following discussion, we show most of our results with α from 10 cycles to 20 cycles. The redundant encoding technique enables linear scaling with the increase of αs so that we can accommodate loose wearout bounds with a linear increase of the number of NEMS switches in the architecture. For the parameter β, we try various values from 4 to 16 according to the Weibull modeling of various kinds of devices in the literature. For example, as mentioned in Section 2.2, MEMS devices in [55] have βs of 12.94, 7.2, 8.58 with geometrical variations, material elasticity and resistance variations. According to [59], typical β values for MEMS reliability fall in 0.5 to 5. We try to push β values down (eg. 4) with redundant encoding to tolerate more process variations.

Then we study how many such devices the limited-use connection requires to meet the fast degradation requirement. The results are shown in Figure 4a without redundant encoding. With small αs, the NEMS switches have tight wearout bounds so that small parallel structures can meet the fast degradation requirement. With large αs, the number of NEMS switches increases exponentially to compensate with the loose wearout bounds (The y-axis in Figure 4a is in log scale). Similarly, with large βs, the NEMS switches are relatively consistent in the degradation so that small parallel structures are feasible. With small βs, the number of devices increases dramatically to control the variations.

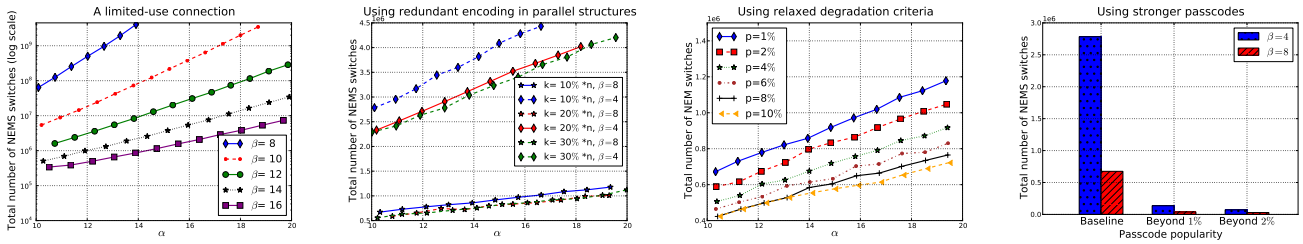
Given the number of devices needed in the architecture, we estimate the area cost analytically assuming an H-tree layout of the NEMS switches and wires. The contact area of each NEMS switch is assumed to be $100nm^2$ and the distance between switches is assumed to be $1nm$ [37]. The area cost of representative engineering options are summarized in Table 1. Although loose wearout bounds and process variations can be compensated by investing more NEMS switches in the architecture to minimize the fabrication cost, the correspondent area cost also increases significantly. For example, when α is 18.69 and β is 10, the area cost is $0.52mm^2$, which could be hard to afford especially when we need to deploy the security hardware on mobile devices. In the next section, we demonstrate that we can reduce the area cost using redundant encoding techniques.

4.3.2 Redundant encoding. As discussed earlier, with redundant encoding, we enforce at least k working NEMS switches in each parallel structure in order to successfully decrypt the key that is required to validate the passcode. This encoding technique helps tighten the wearout bounds for each parallel structure even with NEMS switches that have relatively loose wearout bounds.

Figure 4b shows the effect of encoding the parallel structures with different levels of redundancy. The total number of NEMS switches needed decreases dramatically and scales linearly rather

Table 1: Area cost of the limited-use connection

(α, β)	without encoding (mm^2)	with encoding $k = (10\% * n)$ (mm^2)
(10.51, 16)	1.27e-4	3.2e-5
(10.21, 10)	2.03e-3	1.3e-4
(19.68, 16)	2.03e-3	1.3e-4
(18.69, 10)	5.2e-1	1.3e-4



(a) Without redundant encoding **(b) With redundant encoding** **(c) Relaxed degradation criteria** **(d) Using stronger passcodes**

Figure 4: The total number of NEMS switches needed with different engineering options for the limited-use connection.

than exponentially with the increase of device wearout bounds. For example, without encoding, when α is 14 and β is 8, the number of NEMS switches needed in the architecture is about 4 billion. However, if with redundant encoding and $k = (10\% * n)$, the number of NEMS switches needed is only about 0.8 million with the same α and β. So, the redundant encoding helps reduce 4 orders of magnitude in the total number of NEMS switches. Moreover, it also improves the system tolerance to higher device variations with $\beta = 4$. With $k = (30\% * n)$, however, the decrease in the number of devices needed is negligible so that we can stop enforcing more requisite components in the encoding and decoding.

We assume the storage for component keys should be proportional to the size of the parallel structure and we accommodate that in the area cost evaluation in Table 1. We do not need extra logical circuits for the encoding/decoding because they can be done in CPU.

The switching energy is proportional to the size of each parallel structure. Assume the energy cost for each operation in NEMS switches is 10^{-20} Joule [37]. When α is 14, β is 8 and $k = (10\% * n)$, the total number of NEMS switches is 0.8 million and each parallel structure has 141 NEMS switches. Then the energy cost for each access is 1.41e-18 Joule. Since we use parallel structures, the switching time for each access equals individual NEMS switch's switching time, which is around 10 ns [37].

4.3.3 System-level access bounds. If small variations of the system-level access bounds are desired, we can tune the fast degradation criteria to achieve the variation. The fast degradation criteria in previous experiments are defined as follows: Each parallel structure has at least 99% probability for t accesses while at most 1% probability for $t+1$ accesses. The reliability of the lower bound can be extended to 99.99999% (for exponential decrease in the failure probability) with 3x linear increase in NEMS switches using redundant encoding, in which the minimum 91,250 accesses are guaranteed.

More interestingly, if the application has high tolerance to the upper bound of accesses, then, in each copy, the degradation criteria for $t+1$ accesses, p, can be relaxed from 1% to 10%, for instance. We analyze the number of devices needed and the empirical access bounds when using different degradation criteria, as shown in Figure 4c. When we increase p from 1% to 10%, the empirical access upper bound increases from 91,326 to 92,028 accesses, while the total number of NEMS switches needed is reduced by 40%.

Furthermore, if the passcode is sufficiently secure for many more attempts, we can extend the upper bound to the minimum guesses needed to crack the passcode. According to [62], only the most popular 1% passwords can be guessed with at least 100,000 attempts. Similarly, 2% most popular passwords can be guessed with at least 200,000 attempts. We need at least 675,250 NEMS switches to architect the limited-use connection with an upper bound of 91,326 accesses when $\beta = 8$ and $k = (10\% * n)$, according to Figure 4b. However, with upper bound targets of 100,000 and 200,000 accesses, we only need 38,325 and 29,200 NEMS switches, respectively, as shown in Figure 4d.

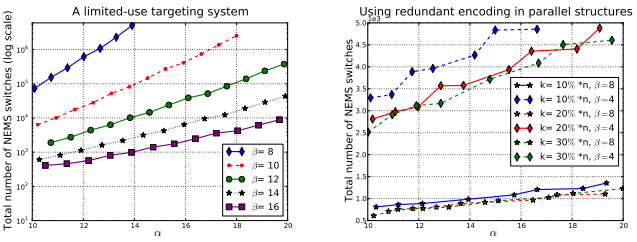
5 A LIMITED-USE TARGETING SYSTEM

Similar to the security enhancement in smartphones, targeting systems can also benefit from the physically enforced limited-usage of targeting commands. Political alliances change over time and devices should be used only for the immediate mission.

Targeting systems are usually composed of three functional subsystems: the command and control system, the communication network, and the launching station. The command and control system makes targeting decisions according to the real-time data from radars and these orders will be encrypted and transmitted securely to the launching station through the encrypted communication link. Although targeting systems have been designed with a high priority for security, there are still many vulnerabilities. Cyber attacks are reported that the hacker can execute commands on a targeting system remotely by either gaining access to the control system or the real-time communication network [7].

We propose to enhance the security of targeting systems with limited-use security architectures. Since the launching station is remotely operated through an encrypted communication link, we restrict the maximum attempts to decrypt the targeting commands at the launching station, which enforces an upper bound to the execution of the targeting commands. By enforcing the upper bound, we can enhance the security of the targeting system from two aspects: 1) it prevents excessive usage of the targeting system beyond the original task, 2) it prevents brute-force attacks to crack the encryption system.

As demonstrated in the limited-use connection use case, device wearout can help physically enforce the access bounds. Similarly, given an expected usage of the targeting commands in one task, such as 100 times, we can exploit the device wearout to build an architecture that automatically wears out after the 100th access to the command decryption key. And we build the architecture inside the launching system so that every access to the command decryption key needs to traverse through the architecture. Compared with the limited-use connection use case, the access bound here is relatively small that a small number of parallel structures are feasible. And the degradation criteria of the parallel structures should be strict



(a) Without redundant encoding (b) With redundant encoding

Figure 5: The total number of NEMS switches needed with different engineering options for the limited-use targeting system.

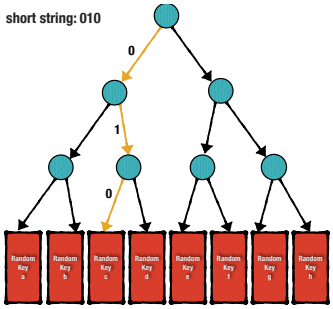


Figure 6: The decision-tree structure for randomness amplification. The receiver follows the short string to get the random key. At each branch, '0' means left and '1' means right. For instance, the short string "010" directs to random key c.

because we do not want a single unintentional targeting command to be executed.

As a result, our design principles of using device wearout to build a limited-use targeting system are: 1) the targeting system should work reliably for the expected number of usage, 100 times, for instance. 2) the targeting system should not work for the 101st time. Since the design goals here are similar as in the last use case, we skip the detailed discussion of design options. Figure 5 shows the total number of NEMS switches needed with different engineering options. Since the targeted access bound is relatively small, the number of NEMS switches needed in the architecture is also reduced by several orders of magnitude compared with the limited-use connection use case. Without redundant encoding, the limited-use targeting system needs at least 8,855 NEMS switches with $\alpha = 20$ and $\beta = 16$. The worst case in Figure 5a is 842,941 NEMS switches with $\alpha = 14$ and $\beta = 8$. With redundant encoding, the total number of NEMS switches can be reduced to 810 when k=(10%*n), $\alpha = 10$ and $\beta = 8$, as shown in Figure 5b. The curves are less smooth because of the small usage target. Only 5 to 10 parallel structures are needed in total and small variations in device wearout bounds can change the total number of parallel structures.

6 USING DEVICE WEAROUT TO BUILD ONE-TIME PADS

One-time pads [19] are important cryptographic algorithms used in many important scenarios as they can provide perfect secrecy [32, 52]. However, to guarantee the perfect secrecy, important rules for one-time pads include that there must be only two copies of the keys (one for the sender and one for the receiver), one copy should be securely transmitted from the sender to the receiver before message transmission, and the sender and receiver must destroy each key immediately after each message encryption/decryption [5].

Traditionally, secret keys were written in real paper pads. Paper, however, severely limits the bandwidth of key distribution. One approach would be to use read destructive memories, which can share the keys and destroy them after use. They can not, however, resist adversarial cloning. Attackers may clone the one-time pads and make two copies, one copy to replace the receiver's original one and another copy for themselves to break the encryption in future message transmission between the sender and receiver (the archaically named "evil maid attack"). Another approach would be to use Physically-Unclonable Devices (PUFs) [58, 64] to fabricate an unclonable one-time pad. PUFs depend upon process variations in each chip, however, making it difficult to fabricate two identical chips so that a sender and receiver could share the pad. We need to both defend against stealthy replications by making it difficult for attackers to ever access the secret keys, yet offer reliable secret sharing between the sender and receiver.

We propose to use wearout devices to provide hardware enforced security for one-time pads. In Section 6.1, we use decision trees to distribute large random keys as a form of randomness amplification. In Section 6.2, we build hardware decision trees with NEMS switches to physically enforce the one-time usage of the keys. In Section 6.3, we exploit redundant encoding techniques to guarantee that the receiver can reliably retrieve the key but adversaries can not. The impacts of engineering options such as decision-tree heights, device wearout characteristics are discussed in Section 6.4.

6.1 Secure transmission of large random keys

In one-time pads, each message employs a new key and the key must be at least as long as the message. As a result, large keys are required for long messages and these keys must be transmitted securely before any message transmission. To relax the requirement for secure transmission of the whole block of random keys, we design decision trees that store many potential keys in their leaves and each key is indexed by a short string about the path information, as illustrated in Figure 6. We assume only the sender and receiver share the right path so that adversaries can only do random path trials to obtain the secret keys.

As a result, the secure transmission of large random keys is divided into the secure transmission of two parts: a short path string and a decision tree that contains many potential random keys. On one hand, the secure transmission of the short path string is less expensive compared with the whole block of random keys. There are many choices for the transmission media and people can even memorize it. If using a temporary channel for the short string, there is less opportunity for adversaries to break it since the transmission time could be very short. On the other hand, decision trees are implemented in wearout devices on a chip, and the secure transmission of them is guaranteed by our design with NEMS devices, which will be explained in Section 6.2.

The chip that contains many decision trees (many random keys) is our new set of "one-time pads" that should be delivered to the

receiver beforehand for many instances of potential message transmission. Even if the chip is obtained by adversaries, the adversaries still have little chance obtaining the right random keys without the right path information. Moreover, the decision tree will be destroyed very quickly after one trial because of the underlying wearout devices. We will explain the hardware implementation of decision trees in detail in Section 6.2 and Section 6.3.

6.2 Hardware design of one-time decision trees with NEMS switches

6.2.1 Design principles. To guarantee the security of one-time pads in decision trees, we need to follow the rules that random strings must be transmitted securely (without stealthy cloning) and destroyed immediately after use. In general, the hardware design of decision trees using NEMS switches is guided by the following principles:

- At least one path should work so that the receiver is able to use the key at least once.
- Not many paths should work, which will prevent adversaries from getting the key within limited trials.
- Each tree should be area efficient so that we can maximize the density of one-time pads on a fixed-size chip.

We build the decision-tree circuit with NEMS switches as intermediate nodes of the tree. On one hand, to allow the receiver to use the tree at least once, the path to the right random string should be successful at least for the first time, which imposes a lower bound on the path reliability. On the other hand, to effectively withstand adversaries' attacks, the paths should fail as quickly as possible so that not many of the paths work successfully, which imposes an upper bound on the path reliability. Similar to the first two use cases, we can physically enforce the access bounds by carefully engineering the decision-tree structures with NEMS switches.

6.2.2 Hardware design of decision trees. Figure 7 shows the schematic of the decision-tree circuit. The decision-tree circuit uses the short path string as control bits to open up the path to the right random string and sends out the right random string serially. The intermediate nodes in the decision tree are implemented with NEMS switches that wear out very quickly, while the random keys in the leaves are implemented in read destructive shift registers. Simply relying on an array of read-destructive or one-time programmable devices (eg. anti-fuse technologies), however, would be vulnerable to "evil maid attacks". The read-destruction could be compromised if reading with a lower voltage. Attackers could also easily clone the devices and bypass the destruction. Our architecture with NEMS switches on the paths will resist cloning by making it difficult for attackers to get access to the memory. And we distribute the random keys into many small memory devices so that it will be hard for attackers to inspect all of them.

As shown in Figure 7, one NEMS switch is used at each branch of the decision tree. Only if all the intermediate nodes along the right path survive after the first access can the receiver successfully obtain the target random string. If each NEMS switch could only be accessed once ideally, then accessing each intermediate node should allow you to choose one path but destroy the other one at the same time since the intermediate switch would fail next time. As a result, only one access could be made to the ideal decision tree and this decision tree meets all of our design goals. However, practical NEMS switches are hard to provide deterministic wearout bounds due to fabrication and process variations. As a result, we exploit the probabilistic modeling of NEMS switches, discussed in Section 2, to reason about the security of hardware decision trees.

6.2.3 Probabilistic reasoning. According to the reliability model in Equation (3), the probability of each NEMS switch surviving the first access is $R(1) = e^{-\left(\frac{1}{\alpha}\right)^{\beta}}$. Assume the height of the decision tree is H. Then the probability of successfully getting through the right path is $(R(1))^H = e^{-\left(\frac{1}{\alpha}\right)^{\beta} H}$. If any of the NEMS switches on the right path failed for the first access, then no one would ever get to the right random key.

A successful decision-tree design should enable a successful first access but prevent any subsequent accesses. The probability for a successful second trial can be throttled by designing a high tree or using NEMS switches with tight wearout bounds so as to guarantee the security and one-time usage of the decision tree. However, under this condition, the probability of a successful first access is also restricted. We should guarantee that the receiver can succeed at least once without sacrificing the security at the same time.

One solution to improve the one-time pads' reliability is to provide multiple copies of the same decision tree for each transmission. The receiver can get the key as far as the right path in one copy is successful. However, the challenge of this solution is to avoid leaking information to adversaries with multiple copies of the same random keys. We solve this problem by exploiting Shamir's secret sharing mechanism and redundant encoding, as discussed in Section 6.3.

6.3 Redundant encoding for reliable and secure key transmission

To prevent information leakage to adversaries with redundant copies, we encode the random strings with error-correction codes and spread them into different copies. Some copies may be erased because of the device failures. The receiver can recover the right random string with a small number of failed copies. The desired feature of the error correction codes, however, is fast degradation once we get more failures beyond our error tolerance target to withstand adversaries' attacks.

Based on Shamir's secret-sharing scheme, as described in Section 4.1.4, each random string is encoded into n component keys stored at the same position of n copies of the decision tree: S_1, S_2, ..., S_n. Each random string S can be computed with k or more S_i components. As a result, to guarantee both the reliability and security of random keys, we need to guarantee that the receiver has close to one probability to get through k or more paths successfully, while adversaries have close to zero probability to do the same. The main difference between the receiver and adversaries is that we assume adversaries have no knowledge about the right path information.

6.3.1 Probabilistic modeling of successful one-time pads. Given the redundant encoding technique, we can calculate the receiver's success probability and adversaries' success probability analytically. The receiver's success probability on one copy is:

$$S_{recv}^{(1)} = e^{-\left(\frac{1}{\alpha}\right)^{\beta} H} \tag{9}$$

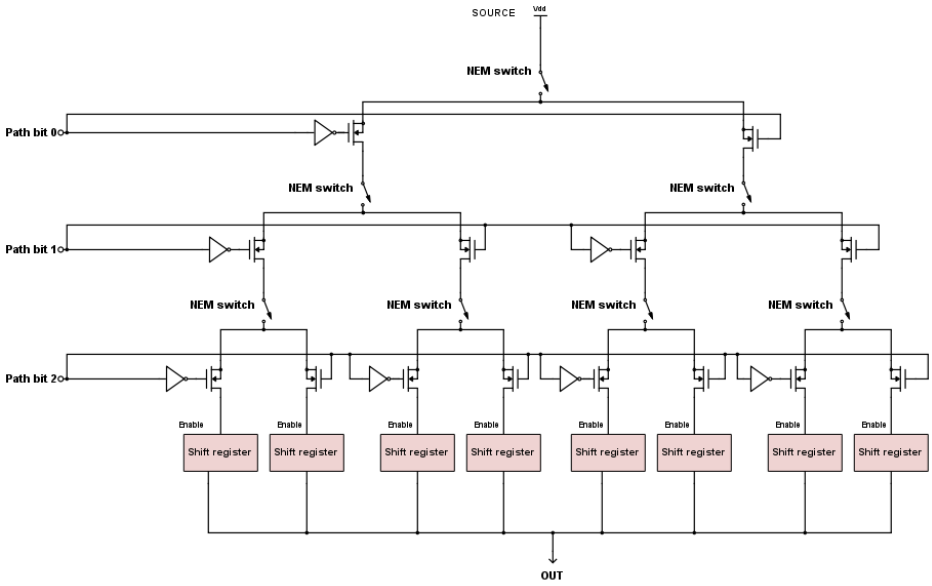


Figure 7: Schematic graph of the implementation of a 3-layer decision tree using NEMS switches

The probability of getting at least k out of n copies successfully is:

$$S_{recv}^{(k+)} = \sum_{i=k}^{i=n} \binom{n}{i} \left(S_{recv}^{(1)}\right)^{i} \left(1 - S_{recv}^{(1)}\right)^{(n-i)} \tag{10}$$

For adversaries, we first consider that they can get through x paths successfully out of n copies. Then, we need to calculate the success probability of k or more out of x being the correct path, with the probability of each successful path being the correct path as follows (since there are $2^{(H-1)}$ paths in total):

$$P = \frac{1}{2^{(H-1)}} \tag{11}$$

The adversaries' success probability for getting through one path in one copy is:

$$S_{adv}^{(1)} = e^{-\left(\frac{1}{\alpha}\right)^{\beta} H} \tag{12}$$

The probability of getting through x paths in n copies is:

$$Prob(x) = \binom{n}{x} \left(S_{adv}^{(1)}\right)^{x} \left(1 - S_{adv}^{(1)}\right)^{(n-x)} \tag{13}$$

The probability that k or more out of x successful paths are the right paths is:

$$Prob_x(k+) = \sum_{i=k}^{i=x} \binom{x}{i} P^i (1-P)^{(x-i)} \tag{14}$$

The probability of getting at least k out of n copies successfully for adversaries is:

$$S_{adv}^{(k+)} = \sum_{x=k}^{x=n} (Prob(x) Prob_x(k+)) \tag{15}$$

, in which $k \leq x \leq n$.

The engineering goal for one-time pads in decision trees is to make sure that $S_{recv}^{(k+)}$ is close to one and $S_{adv}^{(k+)}$ is close to zero. Parameters in the above equations such as H, α, β, n, k will be discussed in Section 6.4 for trade-offs among fabrication cost, area cost, encoding complexity, etc.

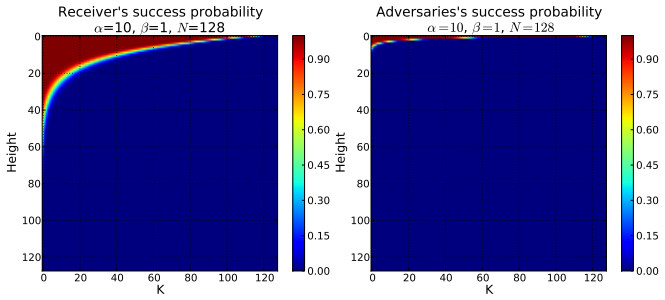
6.4 Engineering space exploration

In this section, we explore the engineering space that can 1) guarantee the success of one-time pads (in both security and reliability) and 2) reduce the fabrication cost and area cost.

We first use a specific type of NEMS switch with an expected lifetime of 10 cycles: $\alpha = 10$ and $\beta = 1$. We can deal with high process variations (small βs) because only the reliability of the first access can affect receiver's and adversaries' success probability. For each key transmission, we use 128 copies of the same decision tree ($n = 128$).

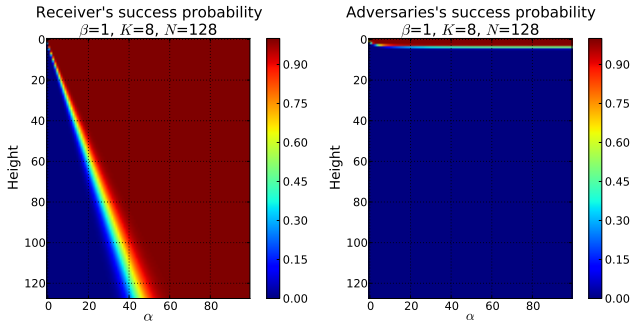
6.4.1 Redundancy levels. Redundancy levels and encoding complexity will be reflected in the values of k. The receiver's and adversaries' success probability with different ks and tree heights Hs is presented in Figure 8. The intersection of the red area in Figure 8a and the blue area in Figure 8b is the success space for one-time pads. With high redundancy, the secret is easy to recover. In contrast, with low redundancy, the access to the secret becomes more difficult since only receiving enough components can help recover the secret. As a result, low redundancy leads to high security. As shown in Figure 8a and Figure 8b, both receiver's and adversaries' success space shrink quickly with the increase of k (decrease of redundancy) but adversaries fail faster. Moreover, when both n and k increase, the complexity of encoding and decoding random strings increases because the latency for constructing and solving the polynomial systems increases.

6.4.2 Tree heights. Higher trees can also enhance the security of one-time pads because 1) the path to the random keys gets longer so that the probability of getting through all the nodes along the path for both the receiver and adversaries becomes smaller, and 2) the number of paths increases exponentially so that it is even harder for adversaries to get on the right path. In Figure 8a, the receiver's success area shrinks when the tree height increases. However, in Figure 8b, the tree height can effectively block adversaries. When



(a) receiver (b) adversaries

Figure 8: Success probability with different tree heights and ks (redundancy levels). The intersection of the red area in the left figure and the blue area in the right figure is one-time pads' success space for both reliability and security.



(a) receiver (b) adversaries

Figure 9: Success probability with different wearout bounds (MTTFs) and tree heights.

the tree height is 8 or more, the adversaries' success probability reduces to zero even if the redundancy level is very high (k is close to 0).

In summary, redundancy provides reliability for the receiver and the tuning of redundancy levels can trade encoding/decoding performance for security. And higher trees can be exploited to further improve the security of one-time pads, while with higher area and performance cost.

In Figure 8, we used a specific type of NEMS switch in which the mean time to failure is about 10 cycles. This can achieve a large success space for one-time pads. However, such NEMS switches may be expensive to fabricate to enforce the wearout fast and under control. Figure 9 shows the impacts of different device wearout bounds defined by the mean time to failure or α values. With higher αs, both the receiver and adversaries have a higher probability of getting the right key. To ensure security, we need higher trees or less redundancy to compensate for the loose wearout bounds of devices. We can see the trade-off between tree heights and wearout bounds when $H \leq 7$: higher trees compensate for looser wearout bounds. When $H \geq 8$, the tree height can effectively withstand any adversaries' attacks. Lower device variations (larger βs) lead to smaller wearout windows. When the target access bound is only one cycle, larger βs postpone the wearout so they do not help ensure security, which also indicates high tolerance to process variations while offering high reliability for the sender and receiver.

6.5 Evaluation

Given the design of hardware one-time pads in decision trees, we want to evaluate how many times a single $1mm^2$ chip can be used for message transmissions between the sender and receiver. We also want to evaluate the latency and energy cost for retrieving a key each time.

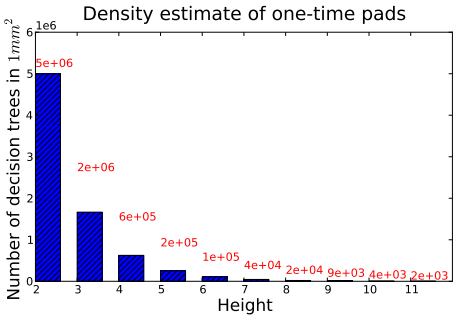


Figure 10: Density estimate

6.5.1 Density estimate. We assume an H-tree layout for the decision tree circuit. The area cost of a complete binary tree in H-layout is at the order of the number of leaves in the tree if nodes are separated with unit distance [12]. Given that the height of each decision tree is H, there are $2^{(H-1)}$ leaves. We assume $100nm^2$ for the dimension of each NEMS switch [8, 36, 37]. Then the area cost for the decision tree is about $100 * 2^{(H-1)} nm^2$. The area cost for shift registers is linear to the size of random strings. The size of each random string is assumed to be proportional to the tree height, that is, around $1000H$ bits. Then the area cost for all the shift registers is $2^{(H-1)} * 1000H * 50nm^2$, assuming a $50nm^2$ cell in the registers.

The number of decision trees we can accommodate in the chip is calculated, as shown in Figure 10. If $H = 4$ and $N = 128$, then we can transmit around $4,687$ one-time pads in this chip.

6.5.2 Latency and energy cost. The latency to retrieve a random key is proportional to the path length and number of decision-tree copies: $lat = \alpha HN$, with α being the delay of a single NEMS switch, at the order of 10 ns [37]. If $N = 128$ and $H = 4$ as in our previous example, the latency to get to the random strings is around 0.00512ms in the worst case. The latency for reading the random string out from the shift register is proportional to the length of the random string. We assume that the propagation delay per bit is around 20ns, such as in MM74HC165 parallel-in/serial-out shift registers. Then the delay of reading the whole random string out is about 0.08ms ($20ns * 1000H$). So the total delay for each random key retrieval is around 0.08512ms.

Similarly, the energy cost on the path to the random string is 5.12e-18 ($NH * 10^{-20}$) Joule in the worst case. The energy cost in reading the random strings out is negligible since only one read from a shift register is needed after successfully getting through a path.

7 LIMITATIONS

This is an initial work to exploit a contrarian view of wearout to build limited-use security architectures. However, more experimentation and discussion on NEMS failure models and security mechanisms are needed in the future.

The primary limitation of the current work is that, even with techniques to decrease system-level sensitivity to device variation, device parameters must still fall within a specific range to make system use targets practical. Note also that we reduce sensitivity to the scale parameter (that represents devices' mean time to failure) but not the shape parameter (which represents devices' variations in lifetime degradation). Furthermore, although the Weibull model is highly parameterized, we need experimental data to validate the range of parameters that are realistic of this or other alternative models. Finally, we might compromise the device's availability for legitimate users in order to guarantee strong confidentiality and integrity. An attacker could purposely degrade the NEMS network through many password guesses to consume the legitimate usage bound, which will hurt the availability for legitimate users. The key issue, however, is to guarantee confidentiality and integrity of the data after the attacker got access to the device. In our design, intentional consumption of the legitimate usage bound could only degrade the device faster, but not leak any information of the confidential data.

8 RELATED WORK

Hardware security has attracted a lot of interests recently. As mobile and embedded devices become ubiquitous, attackers could easily get access to the physical devices. In order to achieve confidential information in the devices, they could exploit any hardware measures to crack the devices, such as brute-force, reverse engineering, side channels, etc [47]. Researchers have proposed a variety of countermeasures to defend against such kinds of physical attacks.These countermeasures can be generally classified into two taxonomies: software assisted approaches and pure hardware approaches:

Software assisted approaches. Many existing hardware security solutions involve the cooperation of software and hardware [4, 42]. For example, modern smartphones have tamper-resistant hardware modules (eg. SoC's secure enclave) to protect the processing and the storage of the device's encryption key and user's confidential information, yet some important security policies that restrict the usage of the keys are implemented in software. The software interface, however, could expose security vulnerabilities and compromise the hardware security. Software may have bugs that lead to violations of the policy. The device could be reprogrammed, either by adversaries or under duress, through these software APIs to implement a different policy. Hence, pure hardware solutions are desired to physically secure the confidential data.

Hardware approaches. Hardware approaches exploit physical device characteristics to protect the data. The specific hardware measures have been ad-hoc and we summarize them into the following three taxonomies:

(1) Physical disorder based security: Physically unclonable functions (PUFs) and random number generators (RNGs) are most popular hardware security primitives in this category [48]. They are based on the inherent randomness in each device due to process variations to generate a unique key for each device for device authentication, IP protection, random number generation, etc. A variety of hardware technologies have been explored to provide the physical disorder, such as SRAM [25], DRAM [61], memristors [33, 63], nanotechnologies [46], etc.

(2) Physical degradation based security: This taxonomy refers to hardware measures that enforce physical usage bounds through purposely degradation of the hardware. [17] created the first self-enforceable hardware for software and content usage metering. They employed the aging effects in transistors due to negative bias temperature instability to measure the time a particular licensed software is used. Similarly, [45] used the SRAM decay phenomenon to measure time for batteryless embedded devices in order to throttle response rates to adversarial accesses. Recently, [66] proposed a memristor-based neuromorphic computing system that can resist adversarial learning of the model and data by degrading the learning accuracy nonlinearly after more inputs are applied to the learning algorithm. Our proposal also stems from this taxonomy but we tailor the devices' degradation characteristics to meet the system level usage requirements through a series of techniques. The methodology can be generally applied to many security architectures where a physically enforced usage window is desired.

(3) Physical destruction based security: Some other hardware approaches can restrict data accesses through self-destructing circuits. However, most current self-destructing devices can only destruct the data, but not the devices [43, 65]. By reprogramming or cloning the devices, all the internal data could be cloned into multiple copies so that the data cannot be secured. Other self-destructing devices need external operations to trigger the destruction [6, 9, 27]. For example, DARPA displayed a new chip built on strained glass substrates that can shatter within 10 seconds when remotely triggered [6]. Nevertheless, our system wears out automatically without a need for remote control.

Meanwhile, more research efforts are needed in the future to study the general design principles, the evaluation and verification methodologies of security architectures that can offer hardware-enforced security.

9 CONCLUSION

In this paper, we propose methodologies of using wearout devices to build security architectures. We explored a probabilistic wearout model with Weibull distribution to characterize the behaviors of NEMS wearout. Based on these characteristics, we design architectures that can physically limit attacks while accommodating legitimate usage. Three use cases are examined: a limited-use connection, a limited-use targeting system and one-time pads. We first present a family of architectural techniques to meet minimum and maximum system-level usage bounds and characterize the design space in terms of device variability (which affects fabrication cost) and device count (which affects area and power). Then we use redundant encoding techniques to improve the security architectures from exponential scaling to linear scaling with the increase of device wearout bounds in the limited-use connection and limited-use targeting system use cases. In the use case of one-time pads, the redundant encoding (Shamir's secret sharing scheme) can effectively throttle the possibility of leaking secret information to adversaries and thus guarantee both reliability and security of one-time pads.

Overall, we envision new opportunities for physically limiting vulnerability to attacks through careful engineering of intentional device wearout.

REFERENCES

[1] Apple firmware updates. https://blog.trailofbits.com/2016/02/17/apple-can-comply-with-the-fbi-court-order/.

[2] Apple iOS Hardware Assisted Screenlock Bruteforce. http://blog.mdsec.co.uk/2015/03/bruteforcing-ios-screenlock.html.

[3] IEEE Standard Specifications For Public Key Cryptography. http://grouper.ieee.org/groups/1363/.

[4] iOS Security Guide. https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

[5] One-time pads (OTP). http://users.telenet.be/d.rijmenants/en/onetimepad.htm.

[6] Self-destructing chips. http://spectrum.ieee.org/tech-talk/computing/hardware/us-militarys-chip-self-destructs-on-command.

[7] Targeting System Attacks. http://europe.newsweek.com/german-missiles-hacked-by-foreign-source-329980?rx=us.

[8] Ali Arab and Qianmei Feng. 2014. Reliability research on micro-and nano-electromechanical systems: a review. *The International Journal of Advanced Manufacturing Technology* 74, 9-12 (2014), 1679–1690.

[9] N Banerjee, Y Xie, Md M Rahman, H Kim, and CH Mastrangelo. 2014. From chips to dust: The MEMS shatter secure chip. In *Micro Electro Mechanical Systems (MEMS), 2014 IEEE 27th International Conference on*. IEEE, 1123–1126.

[10] AH Barber, I Kaplan-Ashiri, SR Cohen, R Tenne, and HD Wagner. 2005. Stochastic strength of nanotubes: an appraisal of available data. *Composites Science and Technology* 65, 15 (2005), 2380–2384.

[11] Maria Berdova, Oili ME Ylivaara, Ville Rontu, Pekka T Törmä, Riikka L Puurunen, and Sami Franssila. 2015. Fracture properties of atomic layer deposited aluminum oxide free-standing membranes. *Journal of Vacuum Science & Technology A* 33, 1 (2015), 01A106.

[12] Richard P. Brent and HT Kung. 1980. On the area of binary tree layouts. *Inform. Process. Lett.* 11, 1 (1980), 46–48.

[13] Juan-Antonio Carballo, Wei-Ting Jonas Chan, Paolo A Gargini, Andrew Kahng, and Siddhartha Nath. 2014. ITRS 2.0: Toward a re-framing of the Semiconductor Technology Roadmap. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on*. IEEE, 139–146.

[14] Rajat Subhra Chakraborty, Seetharam Narasimhan, and Swarup Bhunia. 2007. Hybridization of CMOS with CNT-based nano-electromechanical switch for low leakage and robust circuit design. *IEEE Transactions on Circuits and Systems I: Regular Papers* 54, 11 (2007), 2480–2488.

[15] Soogine Chong, Byoungil Lee, Subhasish Mitra, Roger T Howe, and H-S Philip Wong. 2012. Integration of nanoelectromechanical relays with silicon nMOS. *IEEE Transactions on Electron Devices* 59, 1 (2012), 255–258.

[16] David A Czaplewski, Gary A Patrizi, Garth M Kraus, Joel R Wendt, Christopher D Nordquist, Steven L Wolfley, Michael S Baker, and Maarten P De Boer. 2009. A nanomechanical switch for integration with CMOS logic. *Journal of Micromechanics and Microengineering* 19, 8 (2009), 085003.

[17] Foad Dabiri and Miodrag Potkonjak. 2009. Hardware aging-based software metering. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09*. IEEE, 460–465.

[18] Raden Dewanto, Tao Chen, Rebecca Cheung, Zhongxu Hu, Barry Gallacher, and John Hedley. 2012. Reliability prediction of 3C-SiC cantilever beams using dynamic Raman spectroscopy. In *Nano/Micro Engineered and Molecular Systems (NEMS), 2012 7th IEEE International Conference on*. IEEE, 270–273.

[19] Whitfield Diffie and Martin E Hellman. 1979. Privacy and authentication: An introduction to cryptography. *Proc. IEEE* 67, 3 (1979), 397–427.

[20] Whitfield Diffie, Paul C Van Oorschot, and Michael J Wiener. 1992. Authentication and authenticated key exchanges. *Designs, Codes and cryptography* 2, 2 (1992), 107–125.

[21] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. 2003. Hardness of approximating the minimum distance of a linear code. *Information Theory, IEEE Transactions on* 49, 1 (2003), 22–37.

[22] HD Espinosa, B Peng, N Moldovan, TA Friedmann, X Xiao, DC Mancini, O Auciello, J Carlisle, CA Zorman, and M Merhegany. 2006. Elasticity, strength, and toughness of single crystal silicon carbide, ultrananocrystalline diamond, and hydrogen-free tetrahedral amorphous carbon. *Applied physics letters* 89, 7 (2006), 073111.

[23] XL Feng, MH Matheny, Christian A Zorman, Mehran Mehregany, and ML Roukes. 2010. Low voltage nanoelectromechanical switches based on silicon carbide nanowires. *Nano letters* 10, 8 (2010), 2891–2896.

[24] Daniel Grogg, Christopher L Ayala, Ute Drechsler, Abu Sebastian, Wabe W Koelmans, Simon J Bleiker, Montserrat Fernandez-Bolanos, Christoph Hagleitner, Michel Despont, and Urs T Duerig. 2014. Amorphous carbon active contact layer for reliable nanoelectromechanical switches. In *2014 IEEE 27th International Conference on Micro Electro Mechanical Systems (MEMS)*. IEEE, 143–146.

[25] Jorge Guajardo, Sandeep S Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA intrinsic PUFs and their use for IP protection. In *International workshop on Cryptographic Hardware and Embedded Systems*. Springer, 63–80.

[26] Christoph G Günther. 1989. An identity-based key-exchange protocol. In *Advances in Cryptology-Eurocrypt*. Springer, 29–37.

[27] Jin-Woo Han, Myeong-Lok Seol, Yang-Kyu Choi, and M Meyyappan. 2016. Self-Destructible Fin Flip-Flop Actuated Channel Transistor. *IEEE Electron Device Letters* 37, 2 (2016), 130–133.

[28] Tina He, Rui Yang, Srihari Rajgopal, Mary Anne Tupta, Swarup Bhunia, Mehran Mehregany, and Philip X-L Feng. 2013. Robust silicon carbide (SiC) nano-electromechanical switches with long cycles in ambient and high temperature conditions. In *Micro Electro Mechanical Systems (MEMS), 2013 IEEE 26th International Conference on*. IEEE, 516–519.

[29] Tina He, Fengchao Zhang, Swarup Bhunia, and Philip X-L Feng. 2015. Silicon Carbide (SiC) Nanoelectromechanical Antifuse for Ultralow-Power One-Time-Programmable (OTP) FPGA Interconnects. *IEEE Journal of the Electron Devices Society* 3, 4 (2015), 323–335.

[30] Michael B Henry and Leyla Nazhandali. 2012. From transistors to NEMS: Highly efficient power-gating of CMOS circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 8, 1 (2012), 2.

[31] Ari Juels and Madhu Sudan. 2006. A fuzzy vault scheme. *Designs, Codes and Cryptography* 38, 2 (2006), 237–257.

[32] David Kahn. 1974. *The codebreakers*. Weidenfeld and Nicolson.

[33] Patrick Koeberl, Ünal Kocabaş, and Ahmad-Reza Sadeghi. 2013. Memristor PUFs: a new generation of memory-based physically unclonable functions. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 428–431.

[34] Jeong Oen Lee, Yong-Ha Song, Min-Wu Kim, Min-Ho Kang, Jae-Sub Oh, Hyun-Ho Yang, and Jun-Bo Yoon. 2013. A sub-1-volt nanoelectromechanical switching device. *Nature nanotechnology* 8, 1 (2013), 36–40.

[35] Te-Hao Lee, Swarup Bhunia, and Mehran Mehregany. 2010. Electromechanical computing at 500 C with silicon carbide. *Science* 329, 5997 (2010), 1316–1318.

[36] Owen Loh, Xiaoding Wei, Changhong Ke, John Sullivan, and Horacio D Espinosa. 2011. Robust Carbon-Nanotube-Based Nano-electromechanical Devices: Understanding and Eliminating Prevalent Failure Modes Using Alternative Electrode Materials. *small* 7, 1 (2011), 79–86.

[37] Owen Y Loh and Horacio D Espinosa. 2012. Nanoelectromechanical contact switches. *Nature nanotechnology* 7, 5 (2012), 283–295.

[38] John I McCool. 2012. *Using the Weibull distribution: reliability, modeling and inference*. Vol. 950. John Wiley & Sons.

[39] Robert J. McEliece and Dilip V. Sarwate. 1981. On sharing secrets and Reed-Solomon codes. *Commun. ACM* 24, 9 (1981), 583–584.

[40] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 1996. *Handbook of applied cryptography*. CRC press.

[41] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. 2015. A Large-Scale Study of Flash Memory Failures in the Field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. ACM, 177–190.

[42] Patrick Mutchler, Adam Doupé, John Mitchell, Chris Kruegel, and Giovanni Vigna. 2015. A large-scale study of mobile web app security. *Mobile Security Techologies* (2015).

[43] KP Ng, MC Lee, KC Kwong, and Mansun Chan. 2009. Diode based gate oxide anti-fuse one time programmable memory array in standard CMOS process. In *Electron Devices and Solid-State Circuits, 2009. EDSSC 2009. IEEE International Conference of*. IEEE, 457–460.

[44] Moinuddin K Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. 2009. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 14–23.

[45] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P Burleson, and Kevin Fu. 2012. TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks. In *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 36–36.

[46] Jeyavijayan Rajendran, Ramesh Karri, James B Wendt, Miodrag Potkonjak, Nathan McDonald, Garrett S Rose, and Bryant Wysocki. 2015. Nano meets security: Exploring nanoelectronic devices for security applications. *Proc. IEEE* 103, 5 (2015), 829–849.

[47] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* 102, 8 (2014), 1283–1295.

[48] Masoud Rostami, James B Wendt, Miodrag Potkonjak, and Farinaz Koushanfar. 2014. Quo vadis, PUF?: trends and challenges of emerging physical-disorder based security. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 1–6.

[49] Hebatallah Saadeldeen, Diana Franklin, Guoping Long, Charlotte Hill, Aisha Browne, Dmitri Strukov, Timothy Sherwood, and Frederic T Chong. 2013. Memristors for neural branch prediction: a case study in strict latency and write endurance challenges. In *Proceedings of the ACM International Conference on*

Computing Frontiers. ACM, 26.

[50] Udo Schwalke, Martin Pölzl, Thomas Sekinger, and Martin Kerber. 2001. Ultra-thick gate oxides: charge generation and its impact on reliability. *Microelectronics reliability* 41, 7 (2001), 1007–1010.

[51] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[52] Adi Shamir. 1983. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems (TOCS)* 1, 1 (1983), 38–44.

[53] Zhiwen Shi, Hongliang Lu, Lianchang Zhang, Rong Yang, Yi Wang, Donghua Liu, Haiming Guo, Dongxia Shi, Hongjun Gao, Enge Wang, and others. 2012. Studies of graphene-based nanoelectromechanical switches. *Nano Research* 5, 2 (2012), 82–87.

[54] Sergei Skorobogatov. 2016. The bumpy road towards iPhone 5c NAND mirroring. *arXiv preprint arXiv:1609.04327* (2016).

[55] Trevor S Slack, Farshid Sadeghi, and Dimitrios Peroulis. 2009. A phenomenological discrete brittle damage-mechanics model for fatigue of MEMS devices with application to LIGA Ni. *Journal of Microelectromechanical Systems* 18, 1 (2009), 119–128.

[56] Miloš Stanisavljević, Alexandre Schmid, and Yusuf Leblebici. 2010. *Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures*. Springer Science & Business Media.

[57] Frank Streller, Graham E Wabiszewski, and Robert W Carpick. 2015. Next-Generation Nanoelectromechanical Switch Contact Materials: A Low-Power Mechanical Alternative to Fully Electronic Field-Effect Transistors. *IEEE Nanotechnology Magazine* 9, 1 (2015), 18–24.

[58] G Edward Suh and Srinivas Devadas. 2007. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*. ACM, 9–14.

[59] Danelle M Tanner, Norman F Smith, LLOYD W IRWIN, William P Eaton, KAREN SUE HELGESEN, J JOSEPH CLEMENT, WILLIAM M MILLER, SAMUEL L MILLER, MICHAEL T DUGGER, JEREMY A WALRAVEN, and others. 2000. *MEMS reliability: infrastructure, test structures, experiments, and failure modes*. Technical Report. Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).

[60] Mohammad Tariq Jan, Nor Hisham Bin Hamid, Mohd Haris Md Khir, Khalid Ashraf, and Mohammad Shoaib. 2014. Reliability and Fatigue Analysis in Cantilever-Based MEMS Devices Operating in Harsh Environments. *Journal of Quality and Reliability Engineering* 2014 (2014).

[61] Fatemeh Tehranipoor, Nima Karimian, Wei Yan, and John A Chandy. 2016. DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2016).

[62] Blase Ur, Sean M Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L Mazurek, William Melicher, and Richard Shay. 2015. Measuring real-world accuracies and biases in modeling password guessability. In *24th USENIX Security Symposium (USENIX Security 15)*. 463–481.

[63] Yandan Wang, Wei Wen, Hai Li, and Miao Hu. 2015. A novel true random number generator design leveraging emerging memristor technology. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*. ACM, 271–276.

[64] Yinglei Wang, Wing-kei Yu, Sarah Q Xu, Edwin Kan, and G Edward Suh. 2013. Hiding information in flash memory. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 271–285.

[65] Jinbo Xiong, Zhiqiang Yao, Jianfeng Ma, Ximeng Liu, and Qi Li. 2013. A secure document self-destruction scheme: an ABE approach. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE, 59–64.

[66] Chaofei Yang, Beiye Liu, Hai Li, Yiran Chen, Wujie Wen, Mark Barnell, Qing Wu, and Jeyavijayan Rajendran. 2016. Security of neuromorphic computing: thwarting learning attacks using memristor's obsolescence effect. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 97.

[67] Doe Hyun Yoon, Naveen Muralimanohar, Jichuan Chang, Parthasarathy Ranganathan, Norman P Jouppi, and Mattan Erez. 2011. FREE-p: Protecting non-volatile memory against both hard and soft errors. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*. IEEE, 466–477.

[68] Lunkai Zhang, Brian Neely, Diana Franklin, Dmitri Strukov, Yuan Xie, and Frederic T Chong. 2016. Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 519–531.