Attacking an SRAM-based PUF through Wearout

Alec Roelke
University of Virginia
Charlottesville, VA
ar4jc@virginia.edu

Mircea R. Stan
University of Virginia
Charlottesville, VA
mircea@virginia.edu

***Abstract*—Physical unclonable functions (PUFs) provide a fast and cheap solution to secret key generation. Natural variations in silicon create unique "fingerprints" that are useful for identification. In a 6T SRAM array, these variations cause individual cells to skew their power-on tendency toward storing a 0 or a 1. Wearout effects interfere with those variations, changing the power-on behavior of an SRAM cell in the opposite direction of a stored bit and affecting its reliability as a PUF. In this work, we take advantage of this effect by exposing an SRAM array to high voltage and temperature to activate and accelerate wearout and show that it can cause significant changes to the SRAM's fingerprint. Then we propose an attack on an SRAM PUF that makes use of these conditions with several stored data patterns to modify its fingerprint and then compare the effectiveness of each pattern at producing false negatives for identification challenges. In doing so, we show that false negatives can be increased to 100% in less than 24 hours, effectively erasing the fingerprint and rendering the PUF unusable.**

Keywords—Aging, Wearout, SRAM, PUF

I. Introduction

As electronics continue to increasingly pervade our lives, their security becomes a great concern. A physical unclonable function (PUF) is a useful tool for providing security to electronic devices. By taking advantage of natural variations in electronic circuits that occur during fabrication, a PUF can provide a unique identifier that can't be copied or tampered with during fabrication [1][2]. Additionally, it is able to evaluate input quickly and with little physical overhead [1]. Potential use cases for a PUF include protection of software keys, encryption of data in internal and removable nonvolatile memory, and identification of electronic devices [3].

PUFs typically come in two flavors: memory-based and delay-based. Memory-based PUFs are typically bistable circuits such as SRAMs or sense amplifiers that store a value after being enabled. The value stored can be used to compute the response to a challenge. Delay-based PUFs use process variations in a circuit to produce a delay that can be measured and used in a similar manner to a memory-based PUF's stored value. Delay-based PUFs are typically inverting circuits connected as ring oscillators [4]. In this work, we only consider memory-based PUFs and propose a method by which an SRAM PUF can be attacked without having to physically modify the device or gain knowledge of its response to a challenge. The result of this attack could mean loss of access to a service, inability to verify a device's identity, or even total loss of encrypted data.

A. The SRAM PUF

A PUF based off of a 6T SRAM cell (Figure 1) makes use of the value of that cell when it is first powered on. While it is off, both sides are 0. When power is turned on, the PMOS transistors of the two inverters will both try to pull their side up to V_{DD}. As the voltages of the two sides rise, the NMOS transistors begin to conduct and pull them back down again. If there were no variations, the current through each side would be equal, both sides would rise at the same rate, and a metastable state would be reached. In reality, physical variations in the transistors cause the two sides' currents to be unequal. Because of this, one of the PMOS transistors is able to raise the voltage of its side faster. The voltage of the other side drops and returns to 0 while the first side raises to V_{DD}. Noise creates a random element to the value of the cell at power-on that is heavily influenced by variation.

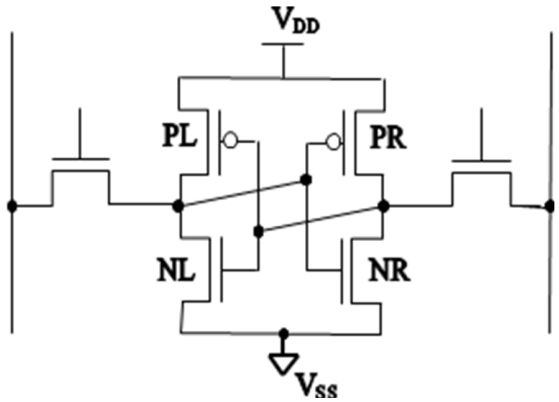

Figure 1. Diagram of a 6T SRAM cell. When the cell powers on, the two sides compete to pull to 1. The sizes of the PMOS transistors affects the rate at which the voltage of the corresponding side can rise, creating a skew in power-on state tendency [10].

The power-on tendency of a 6T SRAM cell can be indicated by a metric called "skew," [5] which is illustrated in Figure 2. The area under the curve on each side of the vertical axis indicates the relative probability that the cell will start up at that value. A heavily 1-skewed cell will have a greater area on the 1 side and, likewise, for 0-skewed on the 0 side, while a balanced cell will have equal areas on both sides. When taken over an entire chip, a "fingerprint" can be formed using the skews of all of the cells (Figure 3). Because variations across a chip cause a different skew in each cell that is not necessarily correlated with the skew in the corresponding cell in other chips, this fingerprint is unique to each SRAM [5]. A quantitative version of skew can be represented by the probability, *p*, of a cell's power-on value to be a 1. As shown in Figure 3, most cells across a chip are heavily 0- or 1-skewed, indicating that skew is very sensitive to variations.

B. PUF-based Identification

Despite the strong 0- or 1-skewed bias of most SRAM cells, some cells are still balanced, and noise still plays a role

978-1-4673-9039-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ISVLSI.2016.68

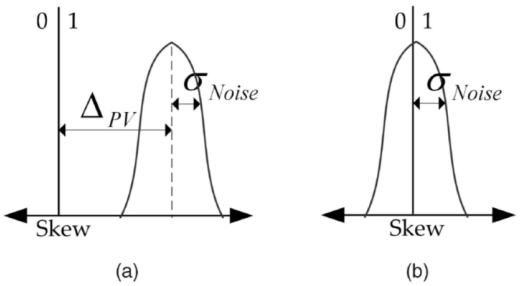

Figure 2. Illustration of 6T SRAM cell skew. The width of the curve illustrates the random effect of noise, and the offset from the horizontal axis represents the effect of variation. When the curve is shifted right, as in (a), the probability of storing a 1 on power-on is high; likewise for the left side and 0. When the curve is centered on the vertical axis, as in (b), the probabilities for both 0 and 1 are equal [5].

in the power-on state. Thus a common difficulty with PUF-based identification is the random element in the response to a challenge. For example, in an SRAM PUF, this is brought upon by the probabilistic nature of each cell's startup tendency. This type of identifier is known as a "fuzzy" identifier [6]. To use such a device for identification, a "fuzzy extractor" must be defined that can account for the randomness in the identifier's response. A fuzzy extractor extracts a uniformly distributed random number from a non-uniformly-distributed one and defines several functions that eliminate noise from the fuzzy identifier's response [6]. Fuzzy extractors work by defining a space that contains all possible values the identifier could attain (for example, for an 8-bit SRAM this space would contain numbers from 0 to 255). Within this space a uniformly-distributed set of codes is defined that is not the same as the set of fuzzy identifiers' fingerprints and is much larger. A pair of functions called a sketch is then defined. One function of the pair maps identifiers to codes and the other recovers from an identifier's

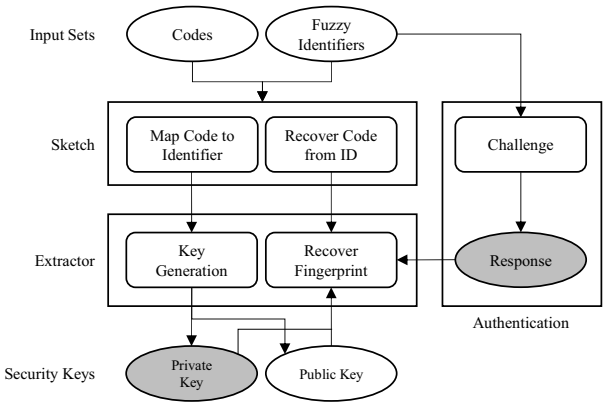

Figure 4. Diagram of a fuzzy extractor. Items in rounded rectangles indicate functions and items in ovals indicate the data those functions operate on. A gray background indicates data that must be kept secret. Note that the fuzzy identifiers themselves can be public, but their responses to challenges must be hidden.

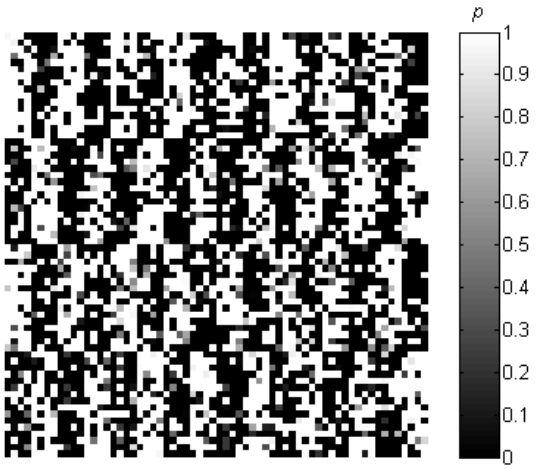

Figure 3. Example section of an SRAM's fingerprint. Each square represents the *p*-value of a single bit; bright squares indicate higher chance of storing 1 at power-on and likewise for dark squares and 0. Some cells have *p*-values close to 0.5, but most are close to 0 or 1.

response its corresponding code. Finally, the extractor itself is a pair of functions that make use of the sketch to generate private and public keys from an identifier's fingerprint and a code (or "enrolls" the identifier) and to regenerate an identifier's fingerprint from its response, private key, and public key. Figure 4 shows a diagram of the operation of a fuzzy extractor. This process is defined in greater detail in [6] and an algorithm for an extractor is defined in [7]. An important benefit of a fuzzy extractor is that it can conceal the identifier's fingerprint. To that end, improvements have been made to increase the difficulty of externally obtaining the fingerprint such as those in [8] and [9].

C. *Effects of Wearout on SRAM PUFs*

Process variations have several effects on an SRAM's operation, including reduction of reliability and yield by reducing noise margins and raising V_{min} [10]. As discussed previously, they also affect the skew of each cell and provide the fingerprint necessary to use the SRAM as a fuzzy identifier. In [10] it is shown that wearing out, or stressing, an SRAM cell via bias temperature instability (BTI) in a stable state changes the strength of one of the PMOS transistors, effectively adjusting the skew of that cell (changing ΔPV in Figure 2). The direction of the change can be manipulated by writing different values to the cell. In [11], it is suggested that this can be used to attack an SRAM PUF by forcing individual bits toward certain values. In this paper we first confirm that wearout can be used to cause significant changes to an SRAM's fingerprint and then propose an attack by which that fingerprint can be modified enough to prevent successful identification of the SRAM without any knowledge of its fingerprint and without physical modification of the hardware.

The rest of this paper is organized as follows: In Section II, related work is discussed. Section III defines the experiments to be performed and Sections IV, V, and VI discuss the results and their impact. The work concludes with Section VII.

Table I. AS6C6264 Specifications [18]

Quantity	Value
Size	8192 × 8 bits
Nominal V_{DD}	3V
Maximum V_{DD}	5V
Operating Temperature	0–70°C

II. Related Work

In [10], Wang et al. show that applying BTI stress to a 6T SRAM cell causes one of its sides to become stronger at the expense of the other. This can be used to balance them or to repair cells that have become so imbalanced that they have lost their ability to read, write, or store data. The authors show that by balancing a cell this way during the burn-in process, V_{min} can be reduced for read, write, and hold functions by up to 128, 75, and 91 mV, respectively, and the overall yield of devices can be increased. In this work, we use this process to modify the fingerprint of an SRAM PUF.

Helfmeier et al. propose a method by which the fingerprint of one SRAM PUF can be cloned onto another using a focused ion beam (FIB) [12]. The authors characterize the fingerprint of the source SRAM by measuring photon emissions during power-on. Once the SRAM's fingerprint is reconstructed, a FIB is used to expose the transistors of another SRAM and modify their behaviors to match it. This modification can be accomplished by either destroying one of the PMOS transistors or by thinning some of the cell's transistors, changing its skew. While the authors show that an SRAM PUF's security can be breached by reading its fingerprint this way, their method of modifying another SRAM's fingerprint is invasive and requires expensive machinery.

Bhargava et al. explore three techniques for improving reliability of SRAM and sense-amplifier PUFS [13]. The first technique is to use directed wearout to reduce the number of errors that can occur due to random processes in the response of a PUF to a challenge. Using this technique, the authors are able to reduce errors by 40%. The second technique is to evaluate the PUF multiple times and average its responses both to generate the fingerprint and to respond to a challenge. Using this technique, errors are reduced to 0.5% using a fingerprint defined by 1000 readings and a response defined by 40 readings. The final technique is to control the rate at which the PUF is activated (power-on for SRAM, enable for sense amplifiers). By increasing ramp time to 14 seconds, the authors are able to reduce errors to 0.67%.

In [7], Maes et al. build upon the concepts described in [6] to design and implement an algorithm for a fuzzy extractor on an FPGA using SRAM as the fuzzy identifier. Alongside the identifier is helper data that serves the function of a public key consisting of a vector containing the probability of each bit to be in error from the fingerprint, the output of the sketch function on the fingerprint, and an identifier for the hash function that was applied to create the secret key. Using the probability vector and sketch output, the original fingerprint can be regenerated and hashed to recreate the secret key. The authors show that a 128-bit secret key can be generated from 1536 bits of SRAM using their algorithm.

Table II. Experiments 1 and 2 Wearout Conditions

Experiment	Quantity	Value
1 (Accelerated Wearout)	Temperature	120°C
	V_{DD}	5V
	Time	3 days
	Pattern	All Ones
2 (Accelerated, Active Wearout)	Temperature	120°C
	V_{DD}	7V
	Time	3 days
	Pattern	All Zeroes

Then in [14], Maes and van der Leest explore several methods of exploiting the data-dependent properties of SRAM wearout to examine their effects on PUF reliability. They discover that the best way to improve reliability is to stress a device at the inverse of its power-on value while the best way to reduce reliability is to do so at the non-inverted power-on value. Additionally, applying stress with part of the inverse power-on state is close to using the full power-on state if the stress pattern can be regenerated periodically and errors can be corrected. The authors measure reliability by measuring bit errors in an SRAM's response, but do not show how this affects its ability to be used in a fuzzy extractor.

III. Wearout Experiments

All experiments are performed using 8-kB AS6C6264 commercial SRAM IC. Relevant specifications are shown in Table I [18]. Measurements are performed at 3.3V V_{DD} and room temperature and consist of 100 power-on state readings.

A. Affecting the Fingerprint with Wearout

The purpose of this experiment is to show that significant, controllable changes can be made to an SRAM's fingerprint and that permanent wearout effects cause those changes to remain unchanged for long periods of time. Two experiments are performed: Accelerated wearout in which high temperature is the main accelerant of wearout while the SRAM is filled with ones, and then accelerated, active wearout in which high voltage is used to activate wearout and temperature is an accelerant while the SRAM is filled with zeroes. The procedures of the two experiments are as follows, with the stress conditions enumerated in Table II:

1. Read the initial state of the SRAM.
2. Write the SRAM and stress for 72 hours.
3. Read the final state of the SRAM.
4. Read the SRAM's state every 15 minutes as it recovers while powered off at room temperature until wearout has been relieved.

Figure 5 shows the changes to the numbers of strong zeroes and strong ones (defined as cells whose *p* values are 0.1 or below and 0.9 or above, respectively) over time during the two experiments. During Experiment 1, the number of strong ones decreased by 8.8% and strong zeroes increased by 6.87%. During Experiment 2, strong ones increased by 36.42% and strong zeroes decreased by 25.96%. Since the

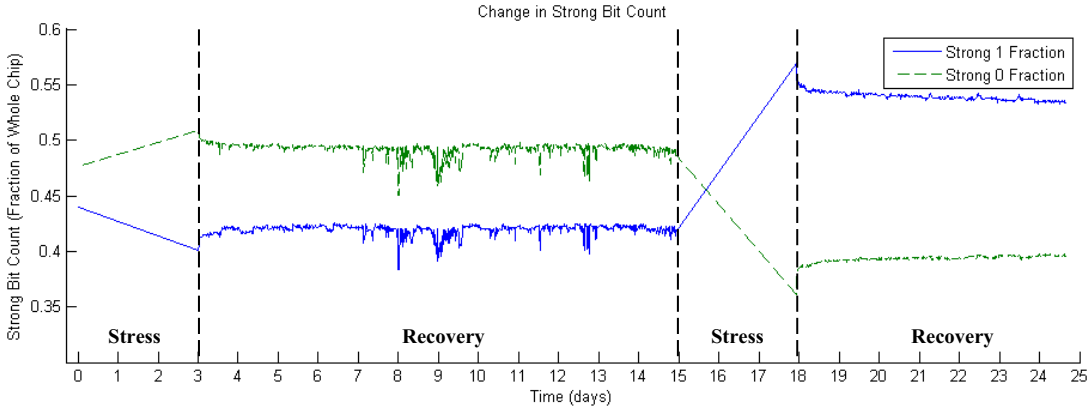

Figure 5. Fraction of strong ones (blue, solid) and strong zeroes (green, dashed) over the course of experiments 1 and 2. The extent to which these fractions change implies the ability to control the startup tendency of the chip. In both cases, most recovery occurs during the first day and then slows down nearly to a stop, indicating that the effects of the fingerprint erasure will be permanent.

numbers of strong zeroes and ones for the fresh chip are close, it is reasonable to assume that the changes would be similar but reversed if the bit patterns had been reversed (that is, if Experiment 1 had used all zeroes and Experiment 2 had used all ones). This corresponds to approximately 8% differences in the two values from the fresh chip. This is much less than the measurements from Experiment 2, which show a 29.31% increase in strong ones over the fresh value and a 24.17% decrease in strong zeroes over the fresh value. Therefore, it can be concluded that elevating the voltage caused a significant enhancement to the wearout effect on the SRAM. Raising V_{DD} by 40% can nearly triple the amount of damage that can be caused to a fingerprint.

We have shown that even strong bits can be affected by wearout. But what about the strongest of the strong? In [15], Xiao et al. propose a method to identify the most stable bits. They show that bits whose *p* values are exactly 0 or 1 and are surrounded by other similar bits tend to be the most stable and are reliable even under extreme temperature and voltage conditions even when other bits are not. The authors provide a heuristic for determining adjacent bits when the SRAM's physical layout is unknown: Concatenate each word in the SRAM and assume that adjacent bits in this string are adjacent physically. A bit is "stable," and usable in a fingerprint, if it has at least twenty strong neighbors. We determine the effectiveness of this method to mitigate our attack by calculating how many of stable bits remain strong after being worn out. The SRAM contains 280 stable bits, and at the end of the experiment 74, or 26.43%, of them were no longer stable. Even the most stable bits in the SRAM are susceptible to wearout.

The extent to which the strong bit counts change during each experiment, and in particular during the second experiment, indicates that it is possible to significantly change the startup tendency of the chip using accelerated stress. Additionally, most natural recovery occurs within the first day and after that slows down nearly to a stop, indicating high retention of the wearout effect and suggesting that the proposed attack will create a lasting effect.

B. Erasing the Chip's Fingerprint

Having shown that an SRAM's fingerprint can be significantly affected by wearout, we now show that this can be used to increase the number of errors in a PUF's response above the amount that can be corrected for by the recovery process, effectively erasing it. We make use of the fuzzy extractor defined in [7] to enroll fingerprints and recover them from responses.

We use four AS6C6264 chips that will be stressed with different data patterns summarized in the legends of Figure 6 and Figure 7 and whose fingerprints are defined using 100 power-on states while fresh. Since there have been some proposals to use part of the normal system memory as a PUF [3], chip 1 will be stressed with random bits to simulate the effect of using the PUF for arbitrary data storage. To simulate the effect of a standalone PUF, chip 2 will be stressed while storing its fingerprint, chip 3 will be stressed with all ones, and chip 4 will be stressed with all zeroes.

Next, for each chip, the maximum order for a Reed-Muller code that will be used to generate helper data is found. This is done with the following process for each chip:

1. Generate 50 order 1 Reed-Muller codes.
2. For each code, calculate the bitwise XOR of that code and the chip's fingerprint. These values will serve as helper data for fingerprint recovery.
3. For each response in the fresh measurement, attempt to recover the chip's fingerprint from each of the 50 codes generated in step 1.
4. If all recovery attempts succeed, increase order by one and go back to step 1. Otherwise, the maximum order is the one from the previous iteration.

The order of Reed-Muller code that will be used for erasure experiments is the lowest value found over all four chips, maximizing the size of the secret key that can be generated. The bit error probability vector can be calculated from the *p* vector of the fingerprint by finding the distance for each bit from 0 or 1, whichever is closer.

Finally, each chip is stressed in an accelerated, active wearout condition at 120°C and with V_{DD} set to 7 V for one week with a measurement performed every 24 hours. With each response taken during the measurement, an attempt is made to recover the fingerprint of the SRAM using the algorithm from [7] and each of the 50 codes used in the

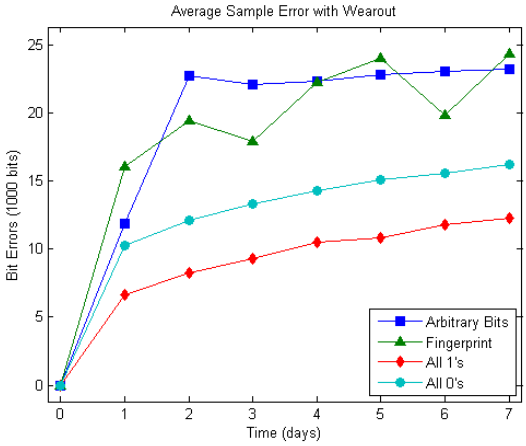

Figure 6. Average number of errors from the fresh fingerprint as each chip wears out.

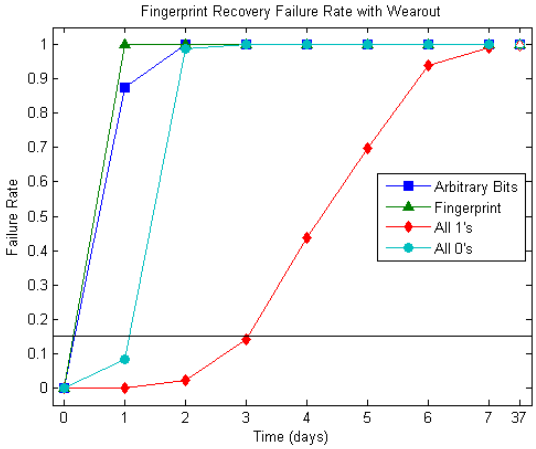

Figure 7. Rate of failure to recover original fingerprint as each chip wears out. The horizontal line at 0.15 indicates the rate at which a chip is considered unreliable. Empty markers indicate recovery failure rate caused by permanent wearout.

previous step. An erasure is considered successful if the extractor can no longer reliably recover the fingerprint from the 100 responses using the helper data for each of the 50 codes, which in [7] is defined as a 15% failure rate. A final measurement is taken one month later after the chip naturally recovers to measure the permanence of the erasure.

IV. Results

From the first part of the erasure experiment, Reed-Muller codes with order four and rank 2517 will be used for fingerprint reconstruction. Due to the chips' min-entropy value of 5.08% [5], this is equivalent to a 127-bit secret key.

Figure 6 shows the average number of errors that occur when reading each chip as it wears out. Chip 2, which stored its fingerprint, shows the fastest initial increase in errors and highest final error rate. In order, chip 1, chip 4, and chip 3 have slower initial increases and lower final error rates. The high values from chip 2 are due to the inverse nature of wearout in SRAM [10]; a bit that stores a value while wearing out will weaken the corresponding side of the cell and reduce the probability it will start up at that value, thus increasing error rate. Chip 4 fails faster than chip 3 because both chips had higher tendency for zeroes in their fingerprints. When stressing an SRAM that stores a uniform value, the bits that tend to start up at that value will increase in error rate while those that tend to start up at the opposite value will decrease in error rate. Stressing while storing all zeroes in a chip with more bits that tend to start up as 0 than as 1 causes a higher error rate than doing so while storing all ones.

Figure 7 shows the failure rate of recovering each chip's fingerprint from the samples taken during each measurement using each of the 50 codes. In all cases, each chip passes above the required failure rate of 15%, or is erased, after three days, with chips 1 and 2 erasing after the first day. Corresponding with the average error rate shown in Figure 6, storing a chip's fingerprint while stressing it causes it to erase the fastest. Storing arbitrary data causes the second fastest erasure; while chip 1 did not quite reach 100% error rate after one day, it still rose far above the required 15% error rate at 84.58%. This suggests that allowing the SRAM to remain powered on while storing its startup value, which is most likely to contain its fingerprint, or allowing access to it for data storage during operation can cause nearly equal damage in erasing the fingerprint. Less effective at erasing the fingerprint is storing a constant value across the entire SRAM; both took over 24 hours to become erased. Chip 3 is particularly slow; after 72 hours of stress it barely rises above 15% failure rate and never reaches 100%, even after a week. Even so, it is clear that wiping out an SRAM's state after powering on so it can't be read will still lead to enough wearout to erase its fingerprint.

Figure 7 also shows the failure rate of each chip after one month of relief from stress following the experiment. In all cases, failure to recover a fingerprint remains above 15%, suggesting that the fingerprint erasure will not recover naturally. Once a PUF has been exposed to enough wearout to render it unusable, it will not normally be usable again.

V. Erasure as an Attack

In [3], several potential use cases and applications of SRAM PUFs are listed. These include key protection for applications or critical assets, encryption of internal and external memories, and device identification. Several of these cases are susceptible to a denial of service attack by erasing an associated PUF's fingerprint with wearout. The retrieval of an access key for an application or data can be prevented, incorrectly revoking a user's privileges when that user has a legitimate license. Similarly, a device with an SRAM PUF that has been compromised this way will no longer be able to identify itself among its peers. In the case of data encryption, any data previously encrypted by the PUF will no longer be accessible since the encryption key cannot be retrieved, potentially leading to total loss of data. If an embedded PUF is used for encryption of internal storage, erasing the PUF's fingerprint can cause the device to cease to function.

VI. Defense Against Erasure

Because PUF identification fundamentally requires that a measurement has some resemblance to the original fingerprint

and because, as shown in Section III, wearout causes significant enough change to remove that resemblance, efforts to improve reliability of worn-out devices are limited. In [16], the authors propose a method to improve reliability by using wearout to even out the number of zeroes and ones in an SRAM PUF's fingerprint and then strengthen weak bits. As shown in Section III.A, strong bits are susceptible to being flipped by wearout, reducing the effectiveness of this strategy as a defense.

Even so, it is possible to defend against such an attack. In [17], a process of actively recovering from wearout by exposing the affected circuit to negative voltage is proposed. The authors show that by proactively applying negative voltage to a circuit, permanent wearout can be reduced or even eliminated. In Figure 7, we show that natural recovery does not recover enough identification reliability to re-enable use. By applying negative voltage after or during stress, it would be possible to reduce permanent effects enough that the PUF can be used again with the original enrollment information.

From Figure 6 and [17], it is apparent that most wearout occurs in a relatively short time with a decreased rate afterward. By applying an extended burn-in process to the transistors of an SRAM, the device could be "hardened" to future wearout, reducing the potential of its effects to change the device's fingerprint post enrollment.

VII. Conclusion

We investigated the use of directed wearout to attack an SRAM PUF. A PUF has an advantage over other forms of secret-key generation in that it is naturally unique, compact, and can quickly calculate responses to challenges. An SRAM PUF has an additional advantage since, in principle, it can make use of existing hardware on a device and be useful after identification for data storage. However, while powered on it is susceptible to wearout that can be directed by storing particular values. This process pushes its startup tendency away from the value being stored; if that value is the same as its startup tendency, then the error rate will be increased.

We showed that an SRAM PUF can be attacked by applying activated and accelerated stress with elevated voltage and temperature. Storing its fingerprint or arbitrary data can reduce its ability to recover its fingerprint 100% in the former case and over 84% in the latter case in only a day and storing a uniform value can reduce it over 15% in three days. This suggests that an SRAM PUF should not be usable as data storage once identification is complete, and that it should be powered down immediately afterward, as the most effective way to erase its fingerprint is to stress it while storing its most probable state at power-on, which is its fingerprint. Even if the SRAM's power-on state is wiped by writing all ones or zeroes, wearout can still cause enough damage to erase its fingerprint. Furthermore, we showed that this damage is permanent; once a PUF is worn out and unusable, it will not naturally recover enough to be used again. The most obvious result of this attack is that once a PUF's fingerprint has been erased, it will no longer be possible to use attached services.

VIII. Acknowledgments

This work has been supported by NSF and SRC under the CLASH project and by CFAR, a STARNET center. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DDGE-1315231.

References

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon Physical Random Functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[2] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical One-Way Functions," in *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[3] "PUF – Physical Unclonable Functions." NXP Semiconductors N.V., Feb 2013.

[4] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest, "Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS," in *2012 Proceedings of the ESSCIRC*, 2012, pp. 486–489.

[5] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," in *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[6] X. Boyen, "Reusable Cryptographic Fuzzy Extractors," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, 2004, pp. 82–91.

[7] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs," in *Cryptographic Hardware and Embedded Systems — CHES*, 2009, pp. 332–347.

[8] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient Helper Data Key Extractor on FPGAs," in *Cryptographic Hardware and Embedded Systems — CHES* 2008, pp. 10–13.

[9] M. Hiller, M. Weiner, L. Rodrigues Lima, M. Birkner, and G. Sigl, "Breaking Through Fixed PUF Block Limitations with Differential Sequence Coding and Convolutional Codes," in *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices*, 2013, pp. 43–54.

[10] J. Wang, S. Nalam, Z. Qi, R. W. Mann, M. Stan, and B. H. Calhoun, "Improving SRAM Vmin and yield by using variation-aware BTI stress," in *2010 IEEE Custom Integrated Circuits Conference (CICC)*, 2010, pp. 1–4.

[11] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.

[12] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning Physically Unclonable Functions," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 1–6.

[13] M. Bhargava, C. Cakir, and K. Mai, "Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 25–30.

[14] R. Maes and V. van der Leest, "Countering the effects of silicon aging on SRAM PUFs," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 148–153.

[15] K. Xiao, M. T. Rahman, D. Forte, Y. Huang, M. Su, and M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of SRAM-PUF," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 101–106.

[16] A. Garg and T. T. Kim, "Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1941–1944.

[17] X. Guo and M. R. Stan, "Work hard, sleep well - Avoid irreversible IC wearout with proactive rejuvenation," in *21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 649–654.

[18] Alliance Memory, Inc., "8K X 8 Bit Low Power CMOS SRAM," 2007.