Challenging On-Chip SRAM Security with Boot-State Statistics

Joseph McMahan[*], Weilong Cui[*], Liang Xia[†], Jeff Heckey[‡], Frederic T. Chong[§], and Timothy Sherwood[*]
[*]University of California, Santa Barbara
Email: {jmcmahan, cuiwl, sherwood}@cs.ucsb.edu
[†]Microsoft
[‡]Avago Technologies
[§]University of Chicago
Email: chong@cs.uchicago.edu

***Abstract*—On-chip memory is regarded by most secure system designers as a safe memory space, beyond the eyes of all but the most sophisticated attackers. Once a value is overwritten or the power has been removed, it is assumed that the data stored inside fully ceases to persist. However, as writes occur, the bit cells gradually wear; if data is written in an asymmetric way (with repeated writes of the same data), the stored information can later be partially reconstructed solely from statistical measurements of the cells' startup states. We present a technique for measuring the vulnerability of memory systems to such wear-in leakage, modeling the process as the recovery of bits from a noisy channel. We demonstrate our techniques on a 130nm SRAM device and demonstrate that if no countermeasures are used, a very simple prediction model is able to correctly reconstruct 27% of the bits of the written secret — enough to probabilistically reconstruct an RSA key.**

I. Introduction

While "off-chip" memories are subject to bus probing [14], subversion [23], or even physical removal from a live system [10], on-chip memories are generally regarded as far more difficult to attack. SRAMs do not appear to retain their data for very long after the power has been removed, and both on-chip probing and tampering require a level of sophistication and dedication well beyond the ordinary. However, the one place in the lifecycle of a bit-cell where the underlying analog nature of the devices shines through is at power-up: if cells are worn asymmetrically, they can leak a surprising amount of information to an attacker.

In "normal" usage, where 1 and 0 are written with equal probability, the wear should be distributed roughly evenly. However, in many mobile devices, security coprocessors, and emerging Internet of Things devices, a more restricted set of software means that vital information (such as keys) may always be written to the same location in the device's various buffers and memories. Repeatedly writing the same data into a memory (either by design or through an attacker's encouragement) will produce uneven wear in the SRAM cells, manifesting as a *shift in the distribution* of startup states for each cell — i.e., the likelihood of observing a "1" vs a "0" when the chip is powered on. Even if the attacker cannot directly access secrets while the system is in "secure" mode, they may boot the system in "secure" and "insecure" modes and observe the differences between the startup probabilities both before and after "secure" computations.

We describe a method to quantify how the observed shifts in probability distributions can give away non-trivial amounts of information about the values written there. Specifically, we demonstrate how the capacity of this leakage can be modeled as a noisy channel carrying information from the written bits to the observed startup probabilities. The information theoretic notion of mutual information (MI) allows us to efficiently compute exactly how many bits can be shared at most between arbitrary distributions – such as a random key written to memory and the startup probabilities of the SRAM cells where it was written. It further allows us to quantify the actual number of bits leaked under different assumptions about what is known *a priori* about the cell (e.g. is it in a region of other bits of similar inherent bias?). To ground our discussion in reality, we perform an analysis of experimental data collected from the on-chip SRAM of several TI MSP430 microprocessors to determine precisely how many bits of information are leaked on average from repeated writes. We show an example attack which recovers 28.4% of a key written on 3 chips with only 3.8% error.

II. Architectural Vulnerabilities to SRAM Wear

In small embedded systems, secret keys might be kept carefully encrypted in storage using a root hardware key, but once brought on chip they are written into a memory buffer for further operations. This sort of behavior is common in both stand-alone embedded systems and the small hardware units charged with managing the security of larger systems-on-chip (such as those very commonly proposed in architecture security work); both of these types of systems operate with limited memory space and use little or no paging.

Threat Model: An adversary must have access to the device before and after the wear-in takes place, and must be able to read data from the SRAM under attack in both the pre- and post-write states. In devices that explicitly have "secure" and "insecure" modes, the memory is characterized in insecure mode, left to run in secure mode to induce wear, and then characterized back in insecure mode again. Generally, the contents of memory on startup may not be viewed as privileged and may be shared. Though our experiments suggest all bits from a secret cannot be determined from a single device, if the same key is shared across multiple devices, a higher fraction can be successfully recovered. The threat is greatest to data that can be fully recovered from partial information, such as RSA keys, which require only 27% of bits to be leaked to reconstruct the entire key [12]. In Section IV, we show the possibility for 27% recovery rates from experimentally observed data using only three devices.

Example Vulnerability: Any system following a pattern of allocate-use-release for resources storing sensitive information may be vulnerable to the attack. As an example, consider the PIC24H microprocessor. The code memory of PIC24H can be partitioned into three segments with decreasing security privilege levels: Boot Segment (BS), Secure Segment (SS), and General Segment (GS). The SRAM is by default accessible by GS. BS and SS have the ability to allocate a piece of SRAM for exclusive access and release it to the GS afterwards during runtime by setting the SFR (Special Function Register).

A proprietary routine in SS that always wipes its secret key before releasing the SRAM is still vulnerable. An attacker can characterize the SRAM, run an application that calls the proprietary routine in SS n times, then re-characterize the SRAM, giving the cell bias shifts of the memory where the secret key was stored.

A. SRAM Wear Mechanisms

The two prevailing models of usage-induced wear in SRAM cells are Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) [28]. HCI occurs when electrons become trapped in the gate oxide, permanently changing the switching characteristics of the transistor [15]. NBTI is the result of elevated temperature or high electric field [22]; it results in positively-charged traps in the gate oxide, which cause the transistor's threshold voltage to increase. Proactive wear-out recovery approaches have been proposed that exploit microarchitectural redundancy to combat NBTI-induced cell failures [22]. Both of these physical processes potentially play a role in creating the wear effects observed in these experiments.

In the standard SRAM cell configuration, writes set voltages on the bit-lines to force the cell into a new state; the current supplied via the bit-lines is larger than that supplied to the cell to ensure success. Depending on which line has the 1, this wears one side of the cell more than the other, producing asymmetric wear that is observable over time. SRAM reads may also induce moderate cell wear, but our experiments focus on wear due to writes.

B. SRAM Cells and Startup States

A standard SRAM cell is bi-stable: it will always settle into a state in which the cell holds either a 0 or a 1. When an SRAM cell is powered on, it undergoes a brief period of metastability before one of the two inverters overpowers the other and one state wins out. The probability that a given cell will resolve into a 1 or a 0 remains constant in most circumstances; this probability is the result of the physical factors of process variation and threshold voltage mismatch within the specific cell. E.g., a given cell may settle into the 1 state on 70% of startups. This 70% is intrinsic to the variations in that particular cell, and will remain constant across many tests — unless the internal voltage differences are altered due to asymmetric wear, or noise characteristics change.

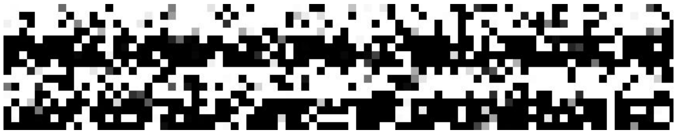

Fig. 1: Distribution of startup probabilities in a sample section of an SRAM. The black pixels always resolve to a 1, while the white pixels always resolve to a 0. The grey pixels have a probability between 0 and 100, with the depth of color reflecting the probability. A pattern of alternating regions of bias were present in all tested memories.

Previous work has proposed using power-up states as hardware fingerprints and random number generators [13]. This work also demonstrated through burn-in experiments that Negative Bias Temperature Instability (NBTI) can shift the startup state probabilities of an SRAM cell.

C. Observations of Cell Bias Trends

Through experimentation discussed in Section III, we found the majority of cells are "strongly biased" and always start in the 0 or 1 state. Approximately 13% of cells were "weakly biased" and could resolve into either the 0 or 1 state on startup, with some probability. The locations of these cells appear to be perfectly spatially random within memory, randomly varying across chips. The values of these weakly biased cells were *far* more likely to measurably shift due to wear.

Interestingly, each SRAM exhibited the same pattern of alternating stripes of strongly-0-biased and strongly-1-biased cells (Figure 1). This same pattern has been observed previously in DRAM, where it was explained as resulting from commonly-occurring alternating wiring patterns in the memory [10].

D. Quantifying Cell Bias Leakage

To provide a theoretic bound on the *amount* of information that can actually leak, we employ the information theoretical concept of Mutual Information — a measure of co-dependence of two random variables. "Information," measured in bits, can be thought of as a quantitative reduction in variance. Given two random processes X and Y, the mutual information between them gives how much one knows (in bits) about the value of Y given the value of X, or X given the value of Y[1]. The mutual information of independent variables is 0.

The mutual information of two random, discrete variables X and Y can be calculated as

$$\mathrm{MI}(X;Y) = \sum_x \sum_y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right),$$

where we have summed over all possible values $x \in X$ and $y \in Y$. $p(x,y)$ is the joint probability distribution of X and

[1] Statistical correlation is a related concept, but mutual information is a more general quantity and makes no assumptions about linear or monotonic relationships.

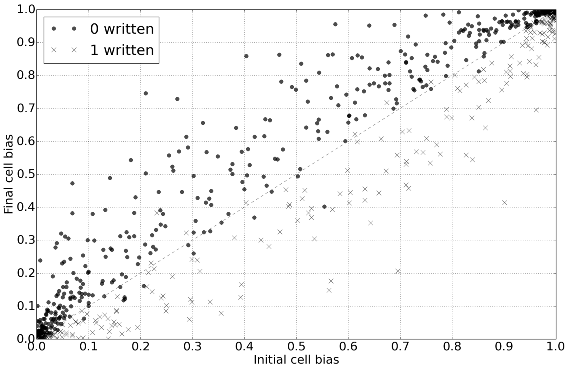

Fig. 2: Probability-bias change in cells. Each point represents a single SRAM cell, with the x-value giving the bias (chance of starting in the 1 state) before writes, and the y-value the bias after the writes.

Y, giving the probability of variables X and Y both taking the specific values x and y. $p(x)$ and $p(y)$ are the marginal distributions of each variable.

This mathematical tool provides us with a method of quantifying information leakage through this new channel. The joint information of data written in SRAM and the observed cell biases will yield an MI figure bounding the number of bits that can be successfully reconstructed on average from the bias measurements alone. This measurement mathematically limits how many bits you can *possibly* learn of one variable given the value of the other.

We treat the process as communication through a noisy bit-channel that corrupts the data, with the written data acting as input and the observed bias as channel output. The goal of the analysis is to compute precisely how much information can be shared between the written data and experimentally observed values through this channel.

III. Experimental Results and Analysis

The experimental data was all collected using common TI microprocessors (130nm TI MSP430G2553) on TI MSP430 launchpad dev boards. The chip was selected for the experiments because of its ease of programming and debugging, along with the accessible on-chip SRAM. Half of the microcontroller's SRAM (256B) was used for testing, while the other half was occupied with the program control itself.

First, the 2048-bit memory space of each chip was characterized 1000 times by reading the cell values just after startup; next, a specific data pattern was repeatedly written many thousands of times. After this, the memory was re-characterized to observe changes on cell startup probability.

Measuring the decay time of this model chip's SRAM, almost all bits are perfectly intact within a fraction of a second after power-off, but all had lost their value after 3 seconds. To ensure that delay-decay remanence effects did not influence the data, each power-on measurement was conducted at least 5 seconds after the previous power-off.

A. Wear Characterization

As mentioned earlier, it was observed that weakly-biased cells with stronger biases (farther from 50%/50%) were less susceptible to changes in their bias. Figure 2 plots each weakly-biased cell according to its bias before and after the writes took place. This figure graphically demonstrates the statistical pattern that allows for data to be reconstructed: note the very clear trend of writing 0's (black dots in the figure) correlating with cell bias increasing towards 1, and of writing 1's (the lighter x's) corresponding to a decreasing cell bias towards 0. This trend among the weakly biased cells to change bias in a predictable direction based just on the data written is the channel through which one can reconstruct some of the of the private data from the SRAM.

B. Physical Wear Conditions

To explore variations in response to differing physical environments, wear experiments were conducted in a dry-ice cooled environment (-50°C), a heated environment (75°C), and at high (3.8V) and low (1.75V) voltages. Compared to several hundred thousands writes at a baseline of standard temperature and operating voltage (25°C, 3.55 V), voltage experiments had no major effects on wear, while cooling provided a modest decrease of bias shift.

It was found that writing in a heated environment accelerated the wear-out process, requiring fewer writes for the statistical shifts to saturate. In as few as 1,000 writes in a heated environment, the SRAM had already begun to show measurable wear. The 3 chips used in the test attack underwent $\sim$ 100,000 repeated writes in a heated environment, providing 6144 cells used in the analysis.

Since NBTI physical effects are non-permanent, daily measurements were conducted on a chip to see if wear effects quickly dissipated within days (which would limit the attack window). In one week of daily measurements, the average weakly-biased cell moved only 1.5%.

C. Mutual Information Analysis

Our goal is to compute how much information can flow from written data to observed cell bias. We note three separate measurements an attacker can make: 1) the cell bias before writing data; 2) the cell bias after data is written; and 3) the bias region to which the cell belongs. We use our experimental data to bound channel capacity for four scenarios:

Scenario 1: The only information is the observation of startup probabilities after the chips have been used.

Scenario 2: Knowledge of the bias striping is also used as 1 additional bit of information on the channel output.

Scenario 3: The attacker has access to the chips before use; the channel output is a pair of values: the bias before the writes took place, and the bias after.

Scenario 4: All information; i.e., the channel output is of three values: the bias before the writes took place, the bias after, and the one bit encoding stripe region of the cell.

The results of the mutual information analyses are summarized in table I. The data shows a maximum channel capacity of 132.4 millibits. This upper limit represents 13.24% of the

TABLE I: Results of mutual information analysis.

Scenario	Information Used			Shared Info.
	post-write	stripe	pre-write	(millibits)
1	✓			63.5
2	✓	✓		77.1
3	✓		✓	129.4
4	✓	✓	✓	132.4

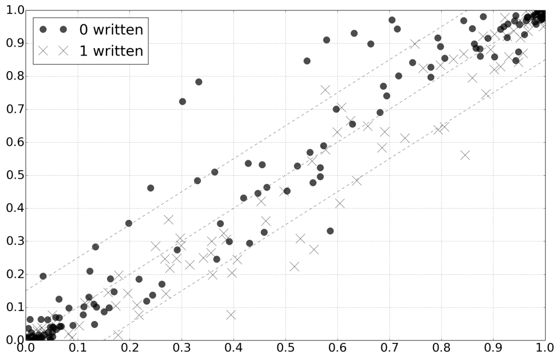

Fig. 3: Probability-bias change in cells for a chip worn with non-uniform data.

written data being recoverable on average, if the channel is able to run at theoretical capacity. The Mutual Information computation provides an upper bound on co-dependence; as the number of measurements of the variables increases, the bound is able to tighten on the amount of information that is actually shared between variables.

D. Read-Only and Mixed-Write Wear

Experiments were also conducted to explore how non-uniform writes to a single cell affected wear, as well as if reads alone could produce noticeable wear. In both cases, biases shifted by comparable amounts to the standard heated, write-only wear experiments, but these shifts appear to be mostly random. With less data in these experiments, a firm conclusion cannot be drawn, but mutual information computations using logarithmically-scaled bins (to more heavily weight larger bias shifts) yield non-zero information when compared to shared information in random keys as a baseline.

A chip worn with read-only wear (no writes after initial write) yielded 2.9 millibits per cell more than the baseline (1,000 random keys), corresponding to 1.4% bit leakage over random data. A chip worn with non-uniform data (1 out of every 5 writes inverting the data) yielded 8.9 millibits more than the baseline, corresponding to 4.3% bit leakage over random (figure 3 shows these bias shifts). These results suggest that wear effects are still measurable under different system usages, meaning that the potential for security breach extends beyond repeatedly-written, uniform data.

IV. Proof of Concept: 27.3% Bit-Recovery

Even modest channel capacities, as in Section III, can provide serious vulnerabilities. We here outline a naive prediction method and apply it to our experimental data of a 2048-bit secret written in SRAM on three tested chips.

Only weakly biased cells will carry information; of the 2048 cells, 784 (38.3%) were weakly biased in at least one of the three chips. Each cell on each chip has a measured change in bias; for each cell, we add the three biases together. As most cells were weakly-biased in only one of the chips, this has little effect; for others, this creates a naive consensus model for the three data points which more heavily weighs larger changes in observed bias. To reduce mispredicts, we take a threshold cutoff value: cells with combined bias change lower than the cutoff value are considered not to be in agreement. If the bias-change sum exceeds the threshold and is negative, we predict a 1 was written in the cell; if above the threshold and positive, we predict 0.

Overall, this correctly predicts 27.3% of the bits, and mispredicts 3.78%. Though the RSA reconstruction algorithm relies on knowing 27% of the bits with certainty [12], many additional works have explored successful approaches to reconstructing RSA keys when bits are known with uncertainty [11], [18], [20].

V. Related Work

Different techniques attempt to address direct attacks through fine grain memory permissions [6], [29], capabilities [30], or other mechanisms limiting an attacker's ability to read or modify program state [5]. Our work instead deals with techniques for quantification of, and preventative measures for, indirect attacks — attacks where the physical or logical side effects of a system can be exploited.

Along these lines is the "cold boot" attacks [10] on DRAM: by lowering the temperature of the chips, Halderman et al were able to prevent DRAM cell decay for several hours, giving time to fully read and reconstruct a system image and crack several commercial file encrypting utilities. However, freezing and removing a portion of memory is not an option for on-chip SRAM. In addition to cold-boot vulnerabilities, Kim et. al. show that DRAM are susceptible to bit-flips in a way that bypasses traditional protection mechanisms through the repeated access of rows [16].

There are many other less direct ways, which take advantage of observations of the dynamics of the system as they operate on secret data. Variations in timing [1], [3], [24], [27] and power utilization [17] or RF emanation [7] can be used to reconstruct keys when many samples are taken.

The indirect attacks closest to this work are memory remanence attacks, such as the cold-boot attack. Memory remanence is the general phenomenon of data lingering in a volatile memory system past a break in power supply. The notion of examining physical wear-in of chips to reconstruct data has existed for decades, with documented methods of attack ranging from I_{DDQ} testing to physically dismantling and probing a chip [9]. As most of these methods rely on physically examining or probing chips to measure wear and reconstruct data, they require specialized equipment and a deep knowledge of the specific device's underlying physics.

The manifestation of SRAM wear in observable changes in cell startup state was first observed by Gutmann [8], where it was observed that storing the same data in SRAM over long periods has the effect of altering the state of the SRAM when it powers up. In one extreme case involving SRAM wear, it was found that key values of a bank security module were intact in SRAM on power-up [2]. Anderson and Kuhn posited that the data had been "burned in" to the SRAM, though this example was called one of "the most extreme cases" [9]. We demonstrate here that a moderate level of "burn-in" is completely observable for a non-negligible fraction of tested SRAM cells, using only software measurement; though not at the level of an entire key leaking, even a small number of bits leaked per chip can lead to fully reconstructed secrets.

The effects of asymmetric memory wear in memory have been observed previously; one novel steganographic technique documents hiding information in flash memory [26]. It relies on the uneven wear induced by writing a 1 vs a 0 in flash cells to alter the time to program each cell. Using this, a secret message can be encoded in the cells' programming times, readable by measuring the average duration of writes for each cell on the device. To prevent these sorts of attacks, methods have been proposed to prevent uneven wear in memories. Chow et al [4] have proposed a system of zero-ing out data after use, and suggest the notion of data life cycles to periodically deallocate longer-lived data. One technique of encoding secrets has demonstrated that an adversary with bit-reading capabilities of 95% fidelity will be unable to recover even a single bit of the original secret [25]. Similar schemes of encoding secrets even in SRAM may counter attacks that attempt to statistically reconstruct SRAM contents from startup states, but this challenges the standard treatment of SRAM as safe memory.

Some works have even proposed using memory remanence effects to *improve* security protocols. One paper exploits characteristics of memory decay phenomena to construct a "physical clock" for the implementation of clockless security protocols in embedded systems [19], [21].

No previous work has performed an analysis of the amount of information that can be reconstructed through statistical measurements of SRAM startup states.

VI. Conclusion

This work cuts at the assumption made by myriad architecture papers dealing with security: that on-chip SRAM should be strictly regarded as beyond the reach of attackers due to the physical hurdles involved. By modeling the process as the recovery of bits from a noisy channel, we show how we can use Mutual Information as a general method for quantifying the capacity for SRAM startup states to leak information. Our computed information channel capacity indicates that with a small number of chips, one can feasibly reconstruct enough data to crack an RSA key. Our demonstrated reconstruction, using a simplistic method of guessing bits, correctly recovers 27% of the key's bits — potentially enough to recover an entire RSA key — using data drawn from only 3 chips.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. 1239567, 1162187, and 1563935.

References

[1] O. Acıiçmez et al. Cache based remote timing attack on the AES. In *Topics in Cryptology–CT-RSA 2007*, pages 271–286. 2006.

[2] R. Anderson and M. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *Proceedings of International Workshop on Security Protocols*, 1997.

[3] D. J. Bernstein. Cache-timing attacks on AES. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf, Apr. 2005. Revised version of earlier 2004-11 version.

[4] J. Chow et al. Shredding your garbage: reducing data lifetime through secure deallocation. In *USENIX Security*, 2005.

[5] J. R. Crandall and F. T. Chong. Minos: Control Data Attack Prevention Orthogonal to Memory Model. In *MICRO*, pages 221–232. IEEE Computer Society, 2004.

[6] J. Devietti et al. Hardbound: Architectural support for spatial safety of the c programming language. *SIGPLAN Not.*, 43(3):103–114, 2008.

[7] K. Gandolfi et al. Electromagnetic Analysis: Concrete Results. In *CHES*, volume 2162, pages 251–261, 2001.

[8] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *USENIX Security*, 1996.

[9] P. Gutmann. Data remanence in semiconductor devices. In *USENIX Security*, 2001.

[10] J. A. Halderman. et al. Lest We Remember: Cold Boot Attacks on Encryption Keys. In *USENIX Security*, 2009.

[11] W. Henecka et al. Correcting errors in rsa private keys. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 351–369. 2010.

[12] N. Heninger and H. Shacham. Reconstructing rsa private keys from random key bits. In *CRYPTO*, pages 1–17, 2009.

[13] D. E. Holcomb et al. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, (9):1198–1210, 2009.

[14] A. Huang. Keeping secrets in hardware: The microsoft xboxtm case study. In *CHES*, pages 213–227. 2003.

[15] J. Keane and C. H. Kim. Transistor aging. *IEEE Spectrum*, 2011.

[16] Y. Kim et al. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In *ISCA*, pages 361–372. IEEE, 2014.

[17] P. Kocher et al. Differential power analysis. In *Advances in Cryptology*, pages 388–397, 1999.

[18] N. Kunihiro and J. Honda. Rsa meets dpa: Recovering rsa secret keys from noisy analog data. Cryptology ePrint Archive, Report 2014/513, 2014. http://eprint.iacr.org/.

[19] Y. Oren et al. On the effectiveness of the remanence decay side-channel to clone memory-based pufs. In *CHES*, volume 8086, pages 107–125. 2013.

[20] K. G. Paterson et al. A coding-theoretic approach to recovering noisy rsa keys. In *ASIACRYPT*, page 386, 2012.

[21] A. Rahmati et al. TARDIS: Time and Remanence Decay in SRAM to Implement Secure Protocols on Embedded Devices without Clocks. In *USENIX Security*, 2012.

[22] J. Shin et al. A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime. In *ISCA*, 2008.

[23] G. E. Suh et al. Efficient memory integrity verification and encryption for secure processors. In *MICRO*, 2003.

[24] Topham and Gonzalez. Randomized cache placement for eliminating conflicts. *IEEETC: IEEE Transactions on Computers*, 48, 1999.

[25] J. Valamehr et al. Inspection resistant memory: architectural support for security from physical examination. In *ISCA*, pages 130–141, 2012.

[26] Y. Wang et al. Hiding Information in Flash Memory. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, 2013.

[27] Z. Wang and R. B. Lee. Covert and side channels due to processor architecture. In *ACSAC*, pages 473–482. IEEE, 2006.

[28] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. 2011.

[29] E. Witchel et al. Mondrian memory protection. *SIGOPS Oper. Syst. Rev.*, 36(5):304–316, 2002.

[30] J. Woodruff et al. The cheri capability model: Revisiting risc in an age of risk. In *ISCA*, 2014.