Towards an Optimal Countermeasure for Cache Side-Channel Attacks

Nivedita Shrivastava and Smruti R. Sarangi

***Abstract*—In the last 15 years, we have witnessed a never ending arm's race between the attacker and the defender with respect to cache-based side-channel attacks. We have seen a slew of attacks, countermeasures (CMs), counterattacks, counter-countermeasures and so on. We analyze the evolution of this area, propose three necessary conditions for designing a successful CM, and then analyze timing and address-based CMs for popular algorithms such as AES and PRESENT. We show that an optimal yet trivial solution for timing-based CMs is possible. Furthermore, address-based CMs are inferior to timing-based CMs, and they can be broken in $O(n^{\log(\log(n))})$ time.**

***Index Terms*—cache side-channel attacks, countermeasures, leakage, formal guarantees**

I. Introduction

FOR the last 15 years, cache-based side-channel attacks (CSCAs), countermeasures (CMs), and subsequent counterattacks (CTAs) have been a popular topic in computer architecture research. They have far-reaching implications in terms of the design of secure hardware. Unfortunately, this is a cat-and-mouse game, where after a CM is published, a few years later a CTA is created, and the sequence continues.

Consider the timeline in Figure 1 – it shows a sequence of CMs for CSCAs and the corresponding CTAs. In this paper, we show that while designing the CM, the developers violated a basic axiom, which made the subsequent development of a CTA possible. In some cases, the original authors of the CM indicated the difficulty of developing a CTA based on the prevailing levels of computational power in CPUs; however, once we had access to faster hardware, this assumption broke, and in some other cases, there were basic flaws in the design of the CMs.

Our key contribution in this short paper is to list three properties that any noise-based CM (one that does not rely on strict partitioning of the caches) must satisfy to make it immune to attacks (explained with examples). We formally analyze the problem and show that for timing-based CMs, a trivial algorithm is also optimal. Furthermore, address-space based CMs are *inferior* and can be easily broken in $O(n^{\log(\log(n))})$ steps for popular encryption algorithms like AES.

II. Background

Cache-based Side-Channel Attacks (CSCAs): The key idea behind such attacks is that it is possible for an attacker process to deliberately induce a miss in a cache line. Subsequently, the time required to access the cache line can be monitored with a high-resolution nanosecond-level timer (HRT). This can give us some information about the memory behavior of the victim process [28], [31], [32] particularly if it is executing a cryptographic algorithm that has data-dependent accesses.

The authors are with the Department of Electrical Engineering, Indian Institute of Technology Delhi, Delhi, India, 110016.E-mail: nivedita.shrivastava@ee.iitd.ac.in, srsarangi@cse.iitd.ac.in

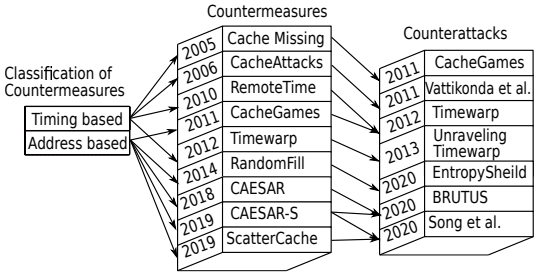

Fig. 1: A timeline of CMs and CTAs

Countermeasures (CMs): Early approaches proposed strict partitioning based CMs where there was no interaction between the attacker and victim processes. However, this was too restrictive and the overheads were high; hence, the current thinking is to increase the noise with various obfuscation techniques such that it is hard for the attacker to derive any useful information from the quantities that she measures. For example, we can add noise to the return value of the HRT, run decoy processes [3], [37], randomize the address space [9], [24], alter the frequency and use a prefetcher [10] to introduce randomness. The efficacy of such a CM can be quantified using information-theoretic results (standard approach).

Mutual Information (MI): Let the quantity of interest that the attacker seeks be the random variable X (cache set address or T-table (for AES-based encryption), time between two events, etc.). From a mathematical perspective, the choice of X (uni or multivariate [7]) can be left to the designer of the CM or CTA. It is assumed that if the attacker receives a sufficient number of samples of X, a successful attack can be mounted.

The attacker actually gets $Y = X + N = g(X)$, N is the noise added by the CM using the noise-adding function g. Information theoretically, we wish to minimize the mutual information (MI) between X and Y defined as $I(X;Y) = H(X) - H(X|Y)$, where $H(X) = -\Sigma p(X) log_2(p(X))$; $H(X|Y) = -\Sigma p(X,Y) log\left(\frac{p(X,Y)}{p(Y)}\right)$. The MI captures all statistical dependences and is the gold standard for estimating the strength of side-channel information [39]. There is a close correlation between the MI and other statistical measures, such as the chi-squared function [41]

Table 1 - List of papers violating the properties. The CTA supports the fact that the failure of a CM is due to a property violation.

Property Violation	Countermeasure	Basic Technique/Claim	Counterattack
$\mathcal{P}1$: No functionality disruption	Disable RDTSC [21], [22]	HRT unavailable	CacheGames [21]
	CacheAttacks: Add an unbounded, random delay to RDTSC [20]	Completely obfuscated RDTSC instruction (time can appear to go backwards)	Timewarp [1]
$\mathcal{P}2$: A large num. of tries	Address randomization for caches: ScatterCache [9]	Requires 33.5 million LLC evictions to find the victim's cache lines	Song et al. [14]
$\mathcal{P}3.1.1$: Det. obfuscation (many-to-one)	Hide the LSBs of RDTSC [20]	Degrade the clock granularity, the observed value of RDTSC follows a step function	Vattikonda et al. [5]
$\mathcal{P}3.1.2$: Det. obfuscation (one-to-one)	CEASER: Address randomization for the caches [8], [24]	The mapped address is generated from the real address using a linear block cipher	BRUTUS [17]
$\mathcal{P}3.2.1$: Unpredictable statistical properties (one-to-many)	Timewarp: Add crafted delays to the output of RDTSC [1]	Difficult to find the noise distribution using statistical analysis	Unraveling timewarp [18]
	RemoteTime: Execute a dummy *'for'* loop with the actual program [3]	Obscured timing information of the actual program	-
	CacheAttacks: Add an unbounded random delay to RDTSC [20]	Forces the attacker to perform and average many measurements	-
	RandomFill: Randomize the cache lines [12]	Randomly fill the cache lines and thus the attacker will find it hard to guess the mapping	EntropyShield [6]

and inter-class distance [42].

III. Properties

In this section, we shall list down the properties that need to be followed for designing a successful noise-based CM (refer to Table 1).

Property 1: No functionality disruption The externally introduced noise should not affect the functionality of the system – no HRT can be turned off.

Property 2: Requires a large number of tries

The number of tries M (to recover the key) needs to overwhelm the computational capability of a hacker (today or in the near future). This has to be set based on technological projections. A large number such as 2^{64} is considered to be safe with today's technology. To derive an upper bound on the information leakage, we rely on the modified Massey's inequality derived in [7]; it says that given the plaintext, $M \geq (2^{-I(X;Y)+log(K)})/e$, where K is the number of bits in the key. We thus have an upper bound ϵ for $I(X;Y)$ given M. Note: For the ease of readability, we will write $I(X;Y|T)$ as $I(X;Y)$ everywhere. Assumption: The plaintext is given.

Property 3.1: Deterministic obfuscation In this case, the mapping from X to Y is decided a priori. There are two mutually exclusive subclasses that are exhaustive. In all cases, the designers need to ensure $I(X;Y) \leq \epsilon$.

[3.1.1] Many-to-one mapping: Different values of X may map to the same value of Y. In the extreme case, Y can be a constant. In this case, $I(X;Y) = 0$.

[3.1.2] One-to-one mapping: There is a one-to-one mapping between X and Y. The mapping is not known in advance and is hard to compute because we have $|X|!$ possible permutations. Now, by not considering unmapped values of Y, and by suitably relabeling values, we can create an equivalent bijective mapping from X to itself. This is also a 1-way permutation.

Property 3.2: Nondeterministic obfuscation This captures the one-to-many scenario, where the mapping between X and Y possibly changes for every measurement. For example, we can just add a random number to X to get a value of Y. We need to still ensure that $I(X;Y) \leq \epsilon$, (refer to Property 2). Let us now discuss an important subclass of this property, which a lot of CMs have violated in the past; this has made it easier to design CTAs.

[3.2.1] Unpredictable statistical properties An attacker should not be able to estimate any useful statistical properties of the noise distribution, even with multiple measurements. For example, if the mean of the noise can be estimated with a large number of samples, then it can be subtracted from a given Y to get its corresponding X. Given that the MI takes into account all statistical dependences, it automatically means that the statistical properties of the noise N or Y are not known and cannot be predicted.

Summary: Any CM needs to satisfy Properties 1, 2, and any one of the sub-properties of Property 3. Then, it is guaranteed that we have a normally performing system (Property 1), the information leaked per measurement is $\leq \epsilon$ (Property 3), and thus M (# tries required) is much more than the available computational capacity (Property 2).

IV. Examples of Property Violations

Let us now evaluate popular CMs and CTAs on the basis of our properties (as mentioned in Table 1), and look at the properties they violate.

A. Property 1: No Functionality Disruption

Percival [22] recommended to completely disable HRTs. However, they are required by various applications such as network drivers and games; disabling them will disrupt their functionality [1], [21]. In comparison, Bangerter et al. [21] altered the working of RDTSC (inst. to access HRT) by adding an unbounded random offset. However, it was pointed out that this CM may make time flow backward [1]. As a result of this issue, a process that is switching its context from one core to another may find a negative offset in the system time [1]. Operating systems

may not work with this change. *It is thus essential that two consecutive RDTSC instructions always respect the wall clock time [1].* Vattikonda [5] et al. also supported this maxim by saying that inconsistent time will lead to inconsistent system performance, for example, "inconsistent file modification timestamps could affect applications like make or kernel daemons". The takeaway point is no CM should morbidly impair the functioning of the system.

B. Property 2: A Large Number of Tries

Even if the CM does not disrupt the system's functionality, assuming a constraint on the attacker's capacity is not wise. The constraint will be violated as soon as the technology catches up. ScatterCache [9] proposes to randomize the addresses. This makes it hard to find the eviction sets (set of lines evicted by the victim). Sadly, we can use a variety of search techniques to reduce the number of tries and also try more using faster hardware (done in this case by Song et al. [14]). They proposed to find the eviction set in only $O(wn)$ time, where w is the number of ways in a cache and n is the number of addresses being tracked. This made the CTA possible.

The number of tries should be in accordance with Property 2; this will ensure that in the foreseeable future it will be difficult to mount such attacks.

C. Property 3.1: Deterministic Obfuscation

Let us now look at cases where the mapping between X and Y is deterministic. The basic precept is that the mapping should not be easily discoverable.

Property 3.1.1: Many-to-One Mapping: Osvik et al. [20] propose to obscure RDTSC's return value by masking the LSB bits; this makes the measured time a step function. The hidden *bit mask* can be easily discovered via statistical analyses [5]. An attacker can also determine the beginning of a timing epoch (for which the output remains constant because of the bit masking). Then she will keep on invoking RDTSC and accordingly increment a proxy counter that will help to estimate the real clock cycle (also highlighted in Timewarp [1]).

Property 3.1.2: One-to-One Mapping: CEASER [24] and CEASER-S [8] are cache architectures in this category where a simple, linear block cipher is used to convert the real address into a mapped address. The linear block cipher uses the uptime, machine id, and key as variables, which ensure that its output varies across different instances of a secure process. However, the linear cipher has limited confusion and diffusion properties – flipping a certain number of bits called 'invariant bits' in the input address will not alter some of the encrypted output bits. As a result, it is possible for an attacker to study Y and find some bits of X. This tremendously reduces the search space. The problem arose because bytes in X were being mapped to a very constrained set of Y values. This fact was used to break this technique by the Brutus paper [17].

The only way to solve this is by increasing the search space such that a given X can possibly map to a large number of Y values making the process of learning the mapping intractable. Learning the mapping should require a *large* number of tries in accordance with Property 2. Unfortunately, as shown in Section V, the search space is quite small for commonly used ciphers such as AES (industrial-strength encryption) and PRESENT (lightweight encryption). This is why such methods are inherently weak.

D. Property 3.2: Nondeterministic Obfuscation

The idea is that the distribution of Y should not provide any information about X and therefore should be unpredictable. Note that if the noise is zero, $X = Y$.

Property 3.2.1: Unpredictable Statistical Properties: This property is an important subset of the parent property. An attacker should not have any estimate about the statistical properties of the noise distribution or Y. Information theoretically, we need to ensure $I(X, Y|\eta) = I(X, Y)$, where η represents any statistical property (e.g. mean, variance, etc.), i.e., there should be no change in the amount of MI between the leaked and actual data, even if an adversary learns some statistical properties.

Assume that the mean of the noise is constant. Now, if X remains constant (key and plaintext remaining the same), we can just average out the noise and subtract it from Y, we will get X. This was the problem with Timewarp [1], which was later exploited by Sarani et al. [18] to design a CTA. Consider Random Fill Cache [12], where several dummy requests are made to addresses that are in the same neighborhood as the original address. Dhavlle et al. [6] leverage the knowledge of the statistical distribution to design a CTA.

V. Discussion: Impossibility Results and Design of a Universal CM

CMs can be grouped into two broad categories [1], [17]: timing and address-based. The former obfuscates the timing information captured by the HRT and the latter obfuscates the addresses. X in the first case is the time [1] and in the latter case is the accessed address in the data structure of interest [9]. For AES, it is the T-table and for PRESENT, it is the S-box.

Given the plaintext T, key K, we have a Markov chain $\langle key(K), T\rangle \rightarrow X \rightarrow Y$ [7]. This will lead us to the equation $I(K;Y) = I(X;Y)$ (see Lemma 6 in [7]). This basically means that the MI between X and Y determines the amount of information that can be extracted out of Y about the key. The two assumptions are that this is a Markov process and the noise distribution N is independent of the key, K (standard assumption). In the generic case, we can use the data processing inequality in information theory and write $I(K;Y) \leq I(X;Y)$ (derived from basic results in [7], please refer to Appendix A for the complete proof.) Note that our **aim** is to minimize $I(K;Y)$ because we need to learn as little as possible about the key from different observations (values of Y). Considering both cases, this translates to minimizing $I(X;Y)$.

A. Timing-based CM

Consider a timing-based CM first. Here, the key idea is to find all the times (values of Y) that are above a

threshold τ across a set of measurements. At least one of them will be more than τ because it will correspond to a cache miss. Based on these values of Y, a guess is made about X. If we wish to minimize $I(X;Y)$, then we need to ensure that the domain (list of allowed values) of Y is as large as possible, which means that all the measurements are $\geq \tau$. Now, assume the limiting case where all values of Y are equal to τ. This can easily be enforced in hardware by pinning cache lines or by adding a dummy delay. We now have $I(X;Y) = 0$ because Y is a constant. In this case, this is a necessary and sufficient condition for a universal CM. If instead, the attack is based on cache hits, then also the same scheme will work. Furthermore, we are minimizing the additional delay added to each measurement by setting $Y = \tau$. This scheme is thus optimal because it minimizes the MI and $(\sum Y - \tau)$.

Conclusion: Alternatively, this means that we do not accrue any benefit by delaying any request for a duration more than τ, and thus any non-trivial CM will perform strictly worse than our trivial solution assuming that $(\sum Y - \tau)$ monotonically determines performance.

B. Address-based CM

For addresses, the many-to-one mapping is not relevant because it violates the basic notion of memory – one address contains one datum. The one-to-many mapping is possible if we have redundant copies; however, the redundancy can be discovered with multiple measurements, and because we need to limit the storage space, we cannot provision a large number of copies. Hence, in practice, only the one-to-one mapping is used. While describing Property 3.1, we proved that this can be made equivalent to a 1-way permutation. Now, note that a keyed 1-way permutation is a block cipher – the key determines the mapping.

We have two options: either we use a block cipher to map X to Y or use an explicit mapping table. Consider the first approach. It means that to remove side channels, we need to map $X \rightarrow Y$ using another block cipher, which will have its own side channels – a never-ending process.

Now, consider the case where we have an explicit mapping table ($X \rightarrow Y$) stored in hardware. This is feasible because most T-tables or S-boxes contain up to 256-1024 entries. If there are n entries, we can have $n!$ possible mappings and thus it may appear that discovering the right mapping is a difficult problem. This is false. Algorithms such as AES and PRESENT have an interesting property for two pieces of plaintext T and T', where $d(T,T') = 1$. Definitions: d is the Hamming distance and $\langle w, w' \rangle$ is a neighboring pair if $d(w,w') = 1$. The corresponding values of Y have the same property, $d(Y,Y') = 1$, because for accessing T-tables or S-boxes we need to XOR plain text bytes with key bytes. A XOR operation preserves the Hamming distance (for the same key).

Let us use this information to represent the entries as a hypercube. Recall that in any labeling of a hypercube, adjacent vertices have a Hamming distance of 1. By choosing plaintext pairs or by observing them in a large sequence, we can find neighboring pairs of Y values. With $O(n.log(n))$ observations we can find all the neighboring pairs and we can construct a hypercube with n vertices. We will however not know the labeling – $Y \rightarrow X$ mapping. The total number of possible labelings of a hypercube is equal to $n(\log n)!$. Now, according to the Stirling's approximation: $n! \approx \sqrt{2\pi n}(n/e)^n$. $(\log n)!$ can be represented as $\sqrt{2\pi \log n}\ (\log n/e)^{\log n}$, which is bounded by $n^{\log\log n}.n^{O(1)}$. Hence, $n(\log n)!$ is bounded by $n^{O(\log\log n)}$: total number of possible labelings ($Y \rightarrow X$ mappings) in a hypercube. For the complete proof, please refer to Appendix B.

Conclusion: This basically means that we need to try out $n^{O(\log\log n)}$ candidate mappings to find out the correct $X \leftrightarrow Y$ mapping. This function has a very slow growth rate and we can exhaustively consider all the combinations. For practical values of n (< 1024), this is computationally feasible and thus even with an explicit mapping table, this family of approaches is very weak, especially, as compared to the trivial yet optimal timing-based scheme.

References

[1] R. Martin *et al.*, "Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks," in *ISCA*, 2012.

[2] D. Page, "Partitioned cache architecture as a ėide-channel defence mechanism," 2005.

[3] D. Jayasinghe *et al.*, "Remote cache timing attack on advanced encryption standard and countermeasures," in *ICIAfS*, 2010.

[4] M. Alam *et al.*, "How secure are deep learning algorithms from side-channel based reverse engineering?" in *Proc. of the 56th Annu. Design Automation Conf. 2019*, 2019, pp. 1–2.

[5] B. C. Vattikonda *et al.*, "Eliminating fine grained timers in xen," in *Cloud computing security workshop*, 2011.

[6] A. Dhavlle *et al.*, "Entropy-shield: Side-channel entropy maximization for timing-based side-channel attacks," in *ISQED'20*.

[7] E. de Chérisey *et al.*, "Best information is most successful," *CHES*, 2019.

[8] M. K. Qureshi, "New attacks and defense for encrypted-address cache," in *ISCA*, 2019, pp. 360–371.

[9] M. Werner *et al.*, "Scattercache: Thwarting cache attacks via cache set randomization," in *USENIX*, 2019.

[10] H. Wang *et al.*, "Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis," in *DATE*, 2020.

[11] J. Seo *et al.*, "Sgx-shield: Enabling address space layout randomization for sgx programs." in *NDSS*, 2017.

[12] F. Liu *et al.*, "Random fill cache architecture," in *MICRO'14*.

[13] D. Gruss *et al.*, "Cache template attacks: Automating attacks on inclusive last-level caches," in *USENIX*, 2015, pp. 897–912.

[14] W. Song *et al.*, "Randomized last-level caches are still vulnerable to cache side-channel attacks! but we can fix it," in *S&P*, 2021.

[15] W. Song and P. Liu, "Dynamically finding minimal eviction sets can be quicker than you think for side-channel attacks against the {LLC}," in *RAID*, 2019.

[16] L. Zhang *et al.*, "Brute force attack on block cipher algorithm based on distributed computation," *Comput. Eng.*, vol. 34, no. 13, pp. 121–123, 2008.

[17] R. Bodduna *et al.*, "Brutus: Refuting the security claims of the cache timing randomization countermeasure proposed in ceaser," *IEEE CAL*, vol. 19, no. 1, pp. 9–12, 2020.

[18] S. Bhattacharya *et al.*, "Unraveling timewarp: What all the fuzz is about?" in *HASP*, 2013.

[19] Y. Zhang and M. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proc. of the ACM SIGSAC conf. on Comput. & commun. security*, 2013, pp. 827–838.

[20] D. A. Osvik *et al.*, "Cache attacks and countermeasures: the case of aes," in *Cryptographers' track at the RSA conf.*, 2006.

[21] D. Gullasch *et al.*, "Cache games–bringing access-based cache attacks on aes to practice," in *S&P*, 2011.

[22] C. Percival, "Cache missing for fun and profit," 2005.

[23] S. Hong *et al.*, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.

[24] M. K. Qureshi, "Ceaser: Mitigating conflict-based cache attacks via encrypted-address and remapping," in *MICRO*, 2018.

[25] W. Song and P. Liu, "Dynamically finding minimal eviction sets can be quicker than you think for side-channel attacks against the {LLC}," in *RAID*, 2019.

[26] P. Vila *et al.*, "Theory and practice of finding eviction sets," in *S&P*, 2019.

[27] M.-M. Bazm *et al.*, "Side-channels beyond the cloud edge: New isolation threats and solutions," in *1st Cyber Security in Networking Conf.*, 2017, pp. 1–8.

[28] Y. Yarom and K. Falkner, "Flush+ reload: A high resolution, low noise, l3 cache side-channel attack," in *USENIX*, 2014.

[29] I. Intel, "and ia-32 architectures software developer's manual," *Volume 3A: System Programming Guide, Part*, vol. 1, no. 64, p. 64, 64.

[30] A. Saxena and B. Panda, "Dabangg: Time for fearless flush based cache attacks." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 637, 2020.

[31] D. Gruss *et al.*, "Flush+ flush: a fast and stealthy cache attack," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016.

[32] F. Liu *et al.*, "Last-level cache side-channel attacks are practical," in *S&P*, 2015.

[33] D. A. Osvik *et al.*, "Cache attacks and countermeasures: the case of aes," in *Cryptographers' track at the RSA conf.*, 2006, pp. 1–20.

[34] D. Sanchez and C. Kozyrakis, "The zcache: Decoupling ways and associativity," in *2010 43rd Annual IEEE/ACM Int. Symp. on Microarchitecture*, 2010, pp. 187–198.

[35] A. Seznec, "A case for two-way skewed-associative caches," *ACM SIGARCH computer architecture news*, vol. 21, no. 2, pp. 169–178, 1993.

[36] A. González, M. Valero, N. Topham, and J. M. Parcerisa, "Eliminating cache conflict misses through xor-based placement functions," in *Proceedings of the 11th international conference on Supercomputing*, 1997, pp. 76–83.

[37] D. Page, "Partitioned cache architecture as a side-channel defence mechanism," *Cryptology ePrint Archive*, 2005.

[38] Q. Ge and et al., "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *JCEN*, 2018.

[39] L. Batina *et al.*, "Mutual information analysis: a comprehensive study," *J. of Cryptology*, 2011.

[40] M. Randolph *et al.*, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, 2020.

[41] S. Weisberg, *Applied linear regression*. J. Wiley & Sons, 2005.

[42] A. Heuser *et al.*, "Information theoretic comparison of side-channel distinguishers: Inter-class distance, confusion, and success," in *Trusted Computing for Embedded Systems*, 2015.

[43] R. Spreitzer and T. Plos, "Cache-access pattern attack on disaligned aes t-tables," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2013, pp. 200–214.

[44] D. Gruss, R. Spreitzer, and S. Mangard, "Cache template attacks: Automating attacks on inclusive {Last-Level} caches," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 897–912.

[45] M. Neve and J.-P. Seifert, "Advances on access-driven cache attacks on aes," in *International Workshop on Selected Areas in Cryptography*. Springer, 2006, pp. 147–162.

[46] D. Wang, Z. Qian, N. Abu-Ghazaleh, and S. V. Krishnamurthy, "Papp: Prefetcher-aware prime and probe side-channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[47] A. Saxena and B. Panda, "Dabangg: time for fearless flush based cache attacks," *Cryptology ePrint Archive*, 2020.

[48] G. Didier and C. Maurice, "Calibration done right: Noiseless flush+ flush attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2021, pp. 278–298.

[49] M. A. Mukhtar, M. Mushtaq, M. K. Bhatti, V. Lapotre, and G. Gogniat, "Flush+ prefetch: A countermeasure against access-driven cache-based side-channel attacks," *Journal of Systems Architecture*, vol. 104, p. 101698, 2020.

[50] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2003–2020.

[51] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering cnn models using side-channel attacks," *IEEE Design & Test*, 2022.

[52] M. Yan, R. Sprabery, B. Gopireddy, C. Fletcher, R. Campbell, and J. Torrellas, "Attack directories, not caches: Side channel attacks in a non-inclusive world," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 888–904.

[53] A. Dhavlle, R. Mehta, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "Entropy-shield: Side-channel entropy maximization for timing-based side-channel attacks," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 161–166.

[54] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 827–838.

[55] D. Sanchez and C. Kozyrakis, "The zcache: Decoupling ways and associativity," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2010, pp. 187–198.

[56] A. Jaamoum, T. Hiscock, and G. Di Natale, "Scramble cache: An efficient cache architecture for randomized set permutation," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 621–626.

[57] D. Page, "Defending against cache-based side-channel attacks," *Information Security Technical Report*, vol. 8, no. 1, pp. 30–44, 2003.

[58] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "Dawg: A defense against cache timing attacks in speculative execution processors," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 974–987.

[59] M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, V. Lapotre, and G. Gogniat, "Nights-watch: A cache-based side-channel intrusion detector using hardware performance counters," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, 2018, pp. 1–8.

[60] M. Mushtaq, A. Akram, M. K. Bhatti, R. N. B. Rais, V. Lapotre, and G. Gogniat, "Run-time detection of prime+ probe side-channel attack on aes encryption algorithm," in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–5.

[61] M. Mushtaq, J. Bricq, M. K. Bhatti, A. Akram, V. Lapotre, G. Gogniat, and P. Benoit, "Whisper: A tool for run-time detection of side-channel attacks," *IEEE Access*, vol. 8, pp. 83 871–83 900, 2020.

[62] M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, M. Yousaf, U. Farooq, V. Lapotre, and G. Gogniat, "Machine learning for security: The case of side-channel attack detection at run-time," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 485–488.

[63] M. Mushtaq, A. Akram, M. K. Bhatti, V. Lapotre, and G. Gogniat, "Cache-based side-channel intrusion detection using hardware performance counters," in *CryptArchi 2018-16th International Workshops on Cryptographic architectures embedded in logic devices*, 2018.

[64] J. Kong, O. Aciiçmez, J.-P. Seifert, and H. Zhou, "Hardware-software integrated approaches to defend against software cache-based side channel attacks," in *2009 IEEE 15th international symposium on high performance computer architecture*. IEEE, 2009, pp. 393–404.

[65] T. Kim, M. Peinado, and G. Mainar-Ruiz, "{STEALTHMEM}:{System-Level} protection against {Cache-Based} side channel attacks in the cloud," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 189–204.

[66] J. Kong, O. Aciicmez, J.-P. Seifert, and H. Zhou, "Deconstructing new cache designs for thwarting software cache-based side

channel attacks," in *Proceedings of the 2nd ACM workshop on Computer security architectures*, 2008, pp. 25–34.

[67] Z. Lv, Y. Zhao, and C. Zhang, "Degradetimer: Mitigating dedicated thread timer based microarchitectural timing channels," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[68] Y. Cao, Z. Chen, S. Li, and S. Wu, "Deterministic browser," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 163–178.

[69] W.-M. Hu, "Reducing timing channels with fuzzy time," *Journal of computer security*, vol. 1, no. 3-4, pp. 233–254, 1992.

[70] M. Sabbagh, Y. Fei, T. Wahl, and A. A. Ding, "Scadet: A side-channel attack detection tool for tracking prime-probe," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2018, pp. 1–8.

[71] G. Sangeetha and G. Sumathi, "An optimistic technique to detect cache based side channel attacks in cloud," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 2473–2486, 2021.

[72] Z. He and R. B. Lee, "How secure is your cache against side-channel attacks?" in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 341–353.

[73] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitraş, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.

[74] W. Cheng, O. Rioul, and S. Guilley, "Guessing a secret cryptographic key from side-channel leakages," in *2019 IEEE European School of Information Theory (ESIT'19)*, 2019.

[75] D. Zhang, C. Zhou, S. Li, D. Yu, and K. He, "Evaluation of information leakage of cryptographic chip based on variance," *IEEE Letters on Electromagnetic Compatibility Practice and Applications*, vol. 2, no. 4, pp. 174–177, 2020.

[76] C. Su and Q. Zeng, "Survey of cpu cache-based side-channel attacks: systematic analysis, security models, and countermeasures," *Security and Communication Networks*, vol. 2021, 2021.

[77] C. Shen, C. Chen, and J. Zhang, "Micro-architectural cache side-channel attacks and countermeasures," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2021, pp. 441–448.

[78] W. Cheng, Y. Liu, S. Guilley, and O. Rioul, "Attacking masked cryptographic implementations: Information-theoretic bounds," *arXiv preprint arXiv:2105.07436*, 2021.

[79] J. Cho, T. Kim, S. Kim, M. Im, T. Kim, and Y. Shin, "Real-time detection for cache side channel attack using performance counter monitor," *Applied Sciences*, vol. 10, no. 3, p. 984, 2020.

[80] P. P. Bhade and S. Sinha, "Detection of cache side channel attacks using thread level monitoring of hardware performance counters," in *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2021, pp. 210–217.

[81] B. Gulmezoglu, A. Moghimi, T. Eisenbarth, and B. Sunar, "Fortuneteller: Predicting microarchitectural attacks via unsupervised deep learning," *arXiv preprint arXiv:1907.03651*, 2019.

[82] P. Vasilikos, H. R. Nielson, F. Nielson, and B. Köpf, "Timing leaks and coarse-grained clocks," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, 2019, pp. 32–3215.

[83] Z. Wang, S. Peng, X. Guo, and W. Jiang, "Zero in and time-fuzz: detection and mitigation of cache side-channel attacks," in *International Conference on Security for Information Technology and Communications*. Springer, 2018, pp. 410–424.

[84] G. Saileshwar and M. Qureshi, "{MIRAGE}: Mitigating {Conflict-Based} cache attacks with a practical {Fully-Associative} design," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1379–1396.

[85] M. Yan, B. Gopireddy, T. Shull, and J. Torrellas, "Secure hierarchy-aware cache replacement policy (sharp): Defending against cache-based side channel attacks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 347–360.

[86] T. Zhang, Y. Zhang, and R. B. Lee, "Cloudradar: A real-time side-channel attack detection system in clouds," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 118–140.

[87] S. Crane, A. Homescu, S. Brunthaler, P. Larsen, and M. Franz, "Thwarting cache side-channel attacks through dynamic software diversity." in *NDSS*, 2015, pp. 8–11.

[88] S. Enomoto and H. Kuzuno, "Flushblocker: Lightweight mitigating mechanism for cpu cache flush instruction based attacks," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2021, pp. 74–79.

[89] H. Chabanne, J.-L. Danger, L. Guiga, and U. Kühne, "Side channel attacks for architecture extraction of neural networks," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 3–16, 2021.

[90] F. Liu, H. Wu, K. Mai, and R. B. Lee, "Newcache: Secure cache architecture thwarting cache side-channel attacks," *IEEE Micro*, vol. 36, no. 5, pp. 8–16, 2016.

[91] H. Wang, H. Sayadi, T. Mohsenin, L. Zhao, A. Sasan, S. Rafatirad, and H. Homayoun, "Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1414–1419.

[92] F. Liu, Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee, "Catalyst: Defeating last-level cache side channel attacks in cloud computing," in *2016 IEEE international symposium on high performance computer architecture (HPCA)*. IEEE, 2016, pp. 406–418.

[93] M. Mushtaq, M. A. Mukhtar, V. Lapotre, M. K. Bhatti, and G. Gogniat, "Winter is here! a decade of cache-based side-channel attacks, detection & mitigation for rsa," *Information Systems*, vol. 92, p. 101524, 2020.

[94] A. Akram, M. Mushtaq, M. K. Bhatti, V. Lapotre, and G. Gogniat, "Meet the sherlock holmes' of side channel leakage: A survey of cache sca detection techniques," *IEEE Access*, vol. 8, pp. 70 836–70 860, 2020.

[95] E. De Cherisey, S. Guilley, O. Rioul, and P. Piantanida, "An information-theoretic model for side-channel attacks in embedded hardware," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 310–315.

[96] A. Ito, R. Ueno, and N. Homma, "On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage," *Cryptology ePrint Archive*, 2022.

[97] B. Mao, W. Hu, A. Althoff, J. Matai, J. Oberg, D. Mu, T. Sherwood, and R. Kastner, "Quantifying timing-based information flow in cryptographic hardware," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 552–559.

[98] B. Gierlichs, L. Batina, and P. Tuyls, "Mutual information analysis–a universal differential side-channel attack," *Cryptology ePrint Archive*, 2007.

[99] A. Duc, S. Dziembowski, and S. Faust, "Unifying leakage models: from probing attacks to noisy leakage." in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 423–440.

[100] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A statistics-based fundamental model for side-channel attack analysis," *Cryptology ePrint Archive*, 2014.

Appendix A

Lemma 1: $I(K;Y|T)$ establishes a lower bound on $I(X;Y|T)$.

Proof: Given the plaintext T, key K, we have a Markov chain $\langle key(K), T\rangle \rightarrow X \rightarrow Y$ [7]. This basically means that the MI between X and Y determines the amount of information that can be extracted out of Y about the key. The two assumptions are that this is a Markov process and the noise distribution N is independent of the key, K.

In the generic case, we can use the data processing inequality in information theory and write $I(K;Y|T) \leq I(X;Y|T)$ (derived from Eqn.8 in [7]).

Using Equation 10 in [7]:

$$I((X,T);(Y,T)) = I(X;Y|T) + H(T) \tag{1}$$

$$I((K,T);(Y,T)) = I(K;Y|T) + H(T) \tag{2}$$

Now, using Equation 8 in [7]:

$$I((K,T);(Y,T)) \leq I((X;T),(Y;T)) \tag{3}$$

Using the previous equations, we get:

$$I(K;Y|T) \leq I(X;Y|T) \tag{4}$$

Appendix B

Lemma 2: *An adversary needs to try out* $\leq n^{O(\log\log n)}$ *candidate mappings to find out the correct X - Y mapping for encryption schemes like AES and PRESENT.*

Proof: In certain algorithms, such as AES, it is necessary to XOR plain text bytes with key bytes prior to accessing T-tables or S-boxes. The Hamming distance between two plaintexts is retained by the XOR operation. Section 3.2 of Reference [43] also emphasizes this point.

Assume, we have address mapping as a countermeasure. This means that the i^{th} entry of the T-table or S-box is actually mapped to the j^{th} entry. In other words, the tables are permuted and this permutation is not known. The attacker can use an ingenious strategy. She can provide two plaintexts that are a Hamming distance of 1 apart. She will then get the indices of the entries in the corresponding tables (using a traditional side-channel attack). These are mapped to T-table or S-box entries that are a Hamming distance of 1 apart. She can continue to do this and find the neighbors of every entry that are a Hamming distance of 1 away. This structure is nothing but a hypercube. In this case, we know the structure of the hypercube, but we still do not know which address is mapped to which entry of the T-table/S-box. We just have the Hamming distance information. In short, we need a labeling of the hypercube. A brute force approach is to go through all labelings and see if we can break the cipher.

Now, it is well known that a hypercube with n vertices has $n(\log n)!$ labelings. We apply the Stirling's approximation to expand $(\log n)!$:

$$m! < \sqrt{2\pi m}\left(\frac{m}{e}\right)^m e^{\frac{1}{12m}} \tag{5}$$

Now, $e^{\frac{1}{12m}} < m$ for $m \geq 2$, and $\sqrt{2\pi m} < 2.51m < m^2$ for $m \geq 3$. Hence, for $m \geq 3$.

$$m! < m^3\left(\frac{m}{e}\right)^m < m^{m+3} \tag{6}$$

Let us now replace m with $\log n$. Now $(\log n)^{\log n} = n^{\log\log n}$. This can be easily proven by taking the log of both sides.

We thus have:

$$(\log n)! < (\log n)^{\log n+3} = (\log n)^3 n^{\log\log n} < n^{\log\log n+3} \tag{7}$$

Given that the number of labelings is $n(\log n)!$, we have the following for $n \geq 8$ ($\log n \geq 3$).

$$n(\log n)! < n^{\log\log n+4} = n^{O(\log\log n)} \tag{8}$$

Hence, the number of labelings of the hypercube is upper-bounded by $n^{O(\log\log n)}$.

Now, we add another lemma to further back up our argument. The results are not included in the main manuscript due to space constraints.