By Jungmin Park and Akhilesh Tyagi

Using Power Clues to Hack IoT Devices

The power side channel provides for instruction-level disassembly.

The field of consumer electronics (CE) has gone digital, inheriting all the security problems of the digital world in the process. The Internet of Things (IoT) class of CE devices is still in a very early adoption phase. Physical side channels are a bigger vulnerability for this class of devices than traditional security threats that exploit protocol, algorithm, or program weaknesses.

Digital Object Identifier 10.1109/MCE.2017.2684982
Date of publication: 14 June 2017

Interestingly, the assembly-level program executing within a device can be reconstructed only through power side-channel observations. This instruction-level disassembly is an emerging power side-channel threat.

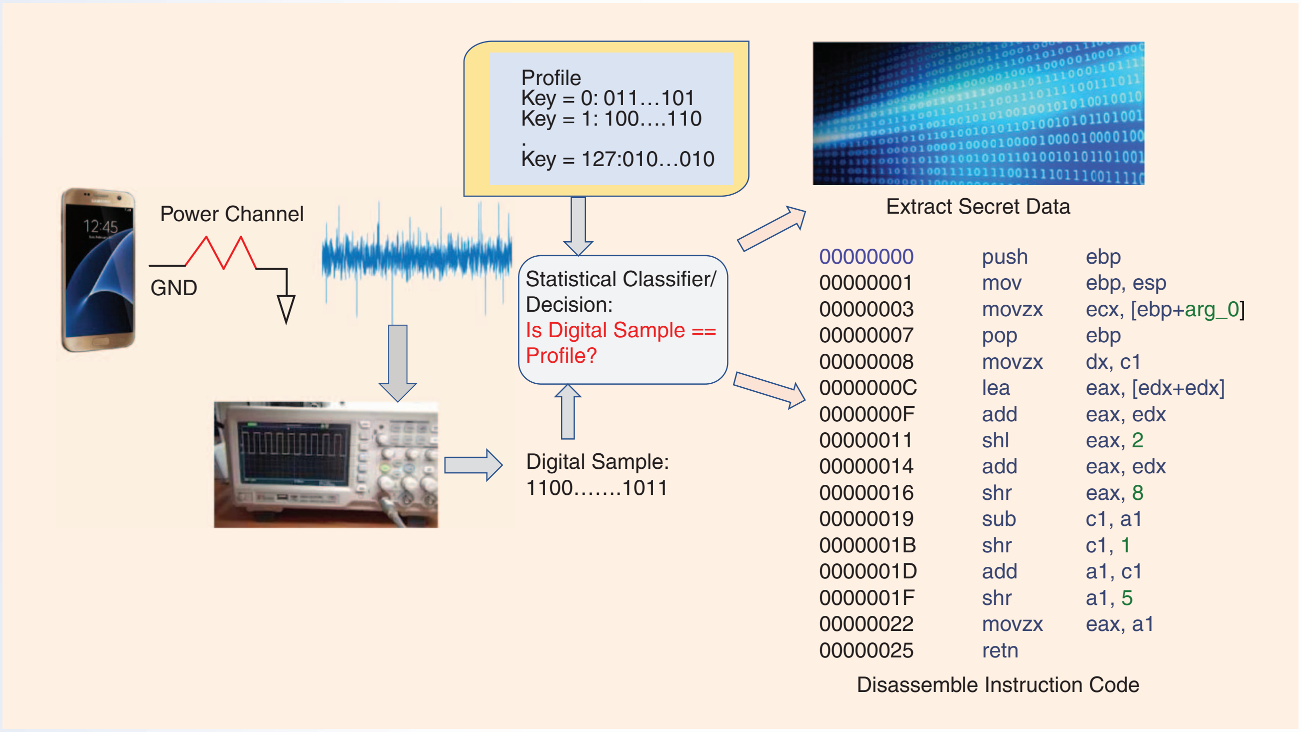
Traditional power side-channel attacks target secret data or keys within crypto hardware or algorithms. Instruction disassembly through power side channels is a significantly more difficult problem than side-channel data leakage for the following reasons: 1) The problem dimensionality for disassembly is on the order of thousands as opposed to an order of 128 or data size for data leakage. The execution time for the statistical classifiers is a function of the dimensionality of the classification problem. 2) Data are at rest, and hence it is not unusual for side-channel data attacks to conduct thousands of experiments with the same secret data. An instruction is in flight within a processor, taking only a fraction of a nanosecond. This leaves little time for a classifier to function. The power side-channel instruction disassembly creates a new kind of digital rights management threat for software-based intellectual property. It also threatens the privacy of user data on these devices.

On 21 October 2016, the recent Dynamic Domain Name System (Dyn DNS) infrastructure was attacked with a distributed denial-of-service (DDoS) assault through the Mirai botnet. As the Dyn DNS DDoS attack demonstrated, IoT devices such as set-top digital video recorders (DVRs), home routers, and personal surveillance cameras can be provisioned for botnet attacks. The instruction disassembly side channel offers another way to probe a device for vulnerabilities against DDoS attacks by future adversaries, even when it is designed as a closed system. In this article, we introduce side-channel analysis tools, instruction-level disassembly issues, and our preliminary experiences with instruction-level disassembly through power side channels.

ASPECTS OF POWER SIDE-CHANNEL THREATS

For many decades, CE devices were designed primarily in an analog manner. Within the last ten years, however, the digitization of CE has caused the resulting products to inherit the digital world's security vulnerabilities. Traditional cyberattacks will probe for the operating system and other software weaknesses through the network. But in the modern CE landscape, physical side-channel attacks exist that are not mounted through the Internet interface. These attacks require physical possession of the device, and they extract private data hosted by these devices through side channels such as power. Such secret-data leakage can compromise private keys embedded within a device. This, in turn, either can violate access rights within an embedded device (unauthorized firmware upgrades or unauthorized content access) or, for networked devices, can enable forged identities.

The problem is exacerbated for IoT-class CE devices that contain sensitive private information. Side-channel vulnerabilities are especially pronounced for this class of devices. The secret-data leakage through power side channels has been studied quite extensively over the last decade [1]–[3]. An emerging threat, however, is assembly-level disassembly solely through power side channels [4]–[8]. Figure 1 illustrates these threats.



```
00000000    push    ebp
00000001    mov     ebp, esp
00000003    movzx   ecx, [ebp+arg_0]
00000007    pop     ebp
00000008    movzx   dx, c1
0000000C    lea     eax, [edx+edx]
0000000F    add     eax, edx
00000011    shl     eax, 2
00000014    add     eax, edx
00000016    shr     eax, 8
00000019    sub     c1, a1
0000001B    shr     c1, 1
0000001D    add     a1, c1
0000001F    shr     a1, 5
00000022    movzx   eax, a1
00000025    retn
```

FIGURE 1. The mobile device is under a power channel attack. The current flow through the GND terminal is correlated with secret data or instructions within the device. The current profile is sampled at high frequency to create a digital signature of the data or instruction. This sampled signature is matched against predetermined data-value or instruction-specific signatures. The presence of noise makes the matching statistical rather than deterministic.

IoT devices such as set-top digital video recorders, home routers, and personal surveillance cameras can be provisioned for botnet attacks.

DATA CONFIDENTIALITY VIOLATION (SECRET-DATA LEAKAGE)

A classic cyberattack exploits a vulnerability in the software and protocol interfaces through the Internet. A side-channel attack, on the other hand, observes a physical by-product of computation to infer some property of a computation. This is analogous to a detective inferring something about criminal events (computation) from the clues (a side channel). In that sense, it is statistical reasoning, similar to Sherlock Holmes's.

For example, a computational event C, such as an XOR of a secret key, with the plaintext data provided by the adversary, occurs within the processor. A derived power event $\mathcal{P}$ occurs at the V_{dd} pin. The adversary can infer C from the clue $\mathcal{P}$. The power event $\mathcal{P}$ could be highly correlated or weakly correlated with the computation event C, depending on how good a model the adversary deploys. The adversary is also not able to synchronize the power event measurement with the computation event trigger perfectly.

Moreover, in addition to the measurement apparatus noise, all the contextual computation not controlled by the adversary is also modeled as noise. This noise makes the adversary's matching of the power event signature of the computation event with the observed power signal difficult. On the plus side for the adversary, though, the latter can repeat this experiment by injecting a trigger to force the computation event and then measure the corresponding power event multiple times. Since the secret data, such as an Advanced Encryption Standard (AES) key, do not often change, such attack experiments can be repeated thousands of times until the adversary has a statistically valid match. The unvarying secret data are sitting ducks inviting such attacks.

CODE CONFIDENTIALITY VIOLATION (INSTRUCTION LEAKAGE)

Instruction-level disassembly or program-level reverse engineering solely through the power side channel is relatively new. It is harder than private data leakage for the following reasons. An instruction from the program executes only once per execution path. Hence, the adversary has little time to make the match. The person does not have the luxury of repeating thousands of experiments that were available with targeting secret data. Moreover, the adversary instruction-level classifier or distinguisher has only as much time to classify as the processor's throughput. If a processor executes four instructions every clock cycle at 3 GHz, the distinguisher has only approximately 0.08 ns/instruction. The secret-data distinguisher is often viewed as an offline activity since the data are not going anywhere!

PROBLEM DIMENSIONALITY/COMPLEXITY

It may appear at first glance that the 128-b AES secret key has a large dimensionality (on the order of 128), whereas a simple instruction like `ADD  R2, R3` has a relatively lower dimensionality of perhaps three (one for the opcode `ADD`, and one each for the two operands). However, a better way to capture the dimensionality of the instruction disassembly is as follows. The first parameter is the number of power samples per clock cycle per instruction s. The second relevant parameter is the number of pipeline stages p. The number of instructions in flight i is important as well, since these instructions are occupying the processor at the same time. Hence, their power signatures are entangled. The overall dimensionality of the problem then is $s * p * i$. Even a simple five-stage pipeline leads to $p = 5$ and $i = 5$. A typical value for s would be 5–10. Even with $s = 5$, the dimensionality of this simple five-stage processor is 125. The midrange ARM Cortex A9, typically used in low-end IoT and consumer devices, has eight to 11 pipeline stages $(p = 8)$ [9], [10]. Since it is an out-of-order dynamically scheduled, dual-issue processor, the number of in-flight instructions is much larger than the number of pipeline stages. Let us estimate this number to be approximately 24. One instruction at each pipeline stage for dual lanes leads to a number of 16–22. Dynamically scheduled processors have large buffers between pipeline stages to hold instructions, leading to our conservative estimate of 24. The dimensionality of this processor then is $5 * 8 * 24$, which is enormous, at 960!

A typical high-end processor, such as the Intel Core i7, would have an overwhelmingly large dimensionality, with $p = 14$ and $i = 128$. The Intel Core i7 has 14 pipeline stages [9]. It is a speculatively scheduled out-of-order processor, with a reorder buffer on the order of 128 instructions, leading to our in-flight instructions count estimate of 128. This leads to a dimensionality of $5 * 14 * 128$, which is 8,960. These dimensionality estimates consider only one core. When multiplied by the number of cores in a modern processor, the dimensionality easily quadruples, leading to a range of 1,000–35,840.

The reason this dimensionality matters, particularly for online instruction identification, is that the complexity of a statistical classifier strongly depends on the dimensionality of the feature set. Given that the classifier may have only a fraction of a nanosecond for the instruction classification, it is of critical importance to reduce the dimensionality of the problem.

In a complex pipeline, the individual instruction power signatures start running into each other due to the signal spread caused by the high capacitance on the V_{dd} power pin. In modern dynamically scheduled superscalar pipelines, the schedule of an instruction window of 128 instructions (the ones that occupy the reorder buffer at the same time) is, at best, nondeterministic. Even if a simulation-driven stage-by-stage power model of individual instructions were to be developed, sequencing 128 power models of 128 in-flight instructions is still combinatorially challenging.

Additional complexity arises from the complex instruction set architecture (ISA) of Intel processors, such as an Intel Core i5 or Core M. A nonuniform length instruction (e.g., 1–17 B) leads to ambiguity in instruction reconstruction. Additional complexity arises as Intel processors perform a dynamic translation from x86 instructions to internal μ-ops. Interpreted execution models, such as JVM or Android's Dalvik VM, also complicate the disassembly of the byte code. In modern Android apps with Android Runtime, a certain fraction of the code remains as Dex Dalvik byte code, but a certain fraction also gets translated into the native ISA code during the app install time through the dex2oat layer. Such a hybrid execution model of native and interpreted code makes this disassembly even more challenging.

The most recent 26 October 2016 DDoS attack on Dyn DNS servers was primarily mounted with over 100,000 infected IoT devices, such as DVRs. Mirai malware is used as part of a botnet in DDoS attacks. The insertion of malware affects the program that is running and causes IoT devices to behave abnormally. One can imagine a scenario where IoT vulnerabilities are probed through power side-channel-based reverse engineering, even for the closed, tamper-proofed class of IoT devices, to develop such botnets.

In some situations, such instruction-level disassembly through the power side channel might actually be desirable. In a recent Defense Advanced Research Projects Agency program, Leveraging the Analog Domain for Security, the explicit goal is to maximize this leakage. The thought process is that, for traditional monitoring-based security, the monitored and the monitor are cohosted on the same platform, and if malware corrupts the monitored, it is also likely to corrupt the monitor. Therefore, if the monitor could be separated from the monitored through a power pin—i.e., if the monitor monitors the program execution of the monitored solely through the power side channel—the existing malware cannot travel across a power pin to corrupt the monitor. The goal in this program is to expose as much of the program execution state through the power side channel as is available locally.

EXISTING RESEARCH

Side-channel attacks correlate information leakage through channels other than the input–output of the system. Timing analysis [11] attacks use the execution time difference of different control paths in an application. Power analysis attacks [1] use the differences between the power consumption of different control paths to learn the secret keys. More than one side-channel model [2], such as the use of the power side channel along with the electromagnetic (EM) radiation side channel, can increase the classification accuracy. A few research projects have used side-channel traces for the disassembly of instructions executing in a device. Statistical techniques such as Bayesian classifiers and principal component analysis (PCA) can be used to construct classification models from the known power consumption traces. Eisenbarth et al. [4] achieved a recognition rate of 70.1% on 35 test instructions and 50.8% on real code by applying a hidden Markov model.

The instruction disassembly side channel offers another way to probe a device for vulnerabilities against DDoS attacks by future adversaries.

Msgna et al. [5] accomplished a 100% recognition rate on a chosen set of 39 instructions in ATMega163-based smart cards running at a clock frequency of 4 MHz. They classified the power traces by applying PCA in combination with the k-nearest neighbors algorithm. The side-channel disassembler of Strobel et al. [6] has a recognition rate 96.24% on test code and 87.69% on real code on a PIC16F687 using multiple EM channels (antennas) with a decapsulated package. Standaert and Archambeau [3] propose linear discriminant analysis (LDA)-based template attacks using power and EM side-channel information. The work described in [5] and [6] uses only instruction classes as the distinguishing feature to classify power traces. Liu et al. [7] use a semantic model of instructions to investigate the data dependency effect on the observed power traces of executing instructions. More recently, McCann et al. [8] have used instruction-level power models to model secret-data leakage. This is unlike the traditional secret-data power side-channel models, where only data-specific switching is modeled.

SIDE-CHANNEL ANALYSIS ATTACKS ON DATA

Common side-channel analysis attacks tap a current path at the V_{dd} or GND pin (as shown in Figure 1) or EM radiation at a specific location in the chip to reveal a secret key. The current or voltage across a shunt resistance, as shown in Figure 1, is sampled at a high rate by a sampling oscilloscope. A model of the secret data can be developed in advance through profiling. For instance, a specific power sample value might correspond to the secret data `Key = 10110111`. Note, however, that a more complete power signature would be a time sequence of sampled digital values for power: $[l_0 = 01110110; l_1 = 01110111; l_2 = 01111011; \ldots; l_{N-1} = 10111111]$ for some N sampling times that overlap with the targeted computation. For the discussion that follows, we will continue to assume that we are dealing with a single time sample.

POWER SIGNATURE MATCHING

If the collected power signature is $l = 01110011$, and our profile suggests the 8-b `Key = 221` with power signature 01110011 after searching all the profile entries, it can safely conclude that the secret key value is 221. The situation is not quite that straightforward, though. The computation environment is very noisy. The power noise (variability) arises from other computations occurring within the processor. It can also

Common side-channel analysis attacks tap a current path at the V_{dd} or GND pin or EM radiation at a specific location in the chip to reveal a secret key.

result from the measurement apparatus. It could also be ambient noise. The timing of the computation event may not be aligned with the digital sampling, which also appears to be noise. Hence, the match between the digital sample and the profiled sample is not an exact, deterministic match.

Given this, we tend to think of the profile signature 01110011 as a distribution rather than a single-point value. The distribution could be viewed as a normal distribution around the decimal value 115 as a mean, with some variance σ^2. Or it could be viewed as a distribution over each bit of 01110011. The second view of bit-level distribution unnecessarily increases the dimensionality of the problem and hence is rarely used. This makes the profile values a bubble with a decimal value 115 sitting in the center. The bubble could be defined by a 2σ or 3σ envelope around 115. If the collected sample falls within this bubble, we could declare the secret data to be 221.

This is where it gets even more tricky. However, the intuitive underpinnings remain the same. First, the profile and collected sample is a time vector rather than a single value. Each component of this vector is called a *feature*. The feature space can be large—on the order of a few hundred to a thousand. Each feature vector component is a probability distribution within the profile to account for noise. Moreover, the feature space need not be defined within a time domain. It is not unusual to consider the power signatures in a frequency domain or in a hybrid time–frequency domain. An additional complication is that the bubbles around each competing hypothesis (such as `Key = 221` and `Key = 95`) are not necessarily disjoint. They often overlap, creating ambiguity. When can we say that there is a match between the profile feature vector and the digital sample vector?

This is where statistical matching, better known as *statistical machine learning*, can help us. This becomes a classification problem. For a secret 8-b data Key, there are 256 possible classes: `Key = 0; Key = 1;` …; `Key = 255`. Given a digital sample vector of N time samples, which Key class is most likely to have given rise to this sample vector? If it is class `Key = 250`, then we classify this sample vector into class `Key = 250`. We will describe these concepts in a bit more detail in the following sections.

FURTHER COMPLICATIONS

When we develop a profile based on secret-data values, we often need to understand the underlying (crypto) algorithm under computation. This is often called a *template-based attack*. What if we do not know anything about the underlying algorithm? Do we have any algorithm-agnostic general models of power induced by certain secret-data values? Since pretty much every computing device is based on complementary metal–oxide–semiconductors (CMOSs), we could consider general CMOS switching characteristics as parameters for power models.

In general, in many function units, the number of 1s in a data word determines the number of switching events. Imagine a precharged (domino) logic, where precharged nodes are discharged if a specific input bit driving the gate of an n-type metal–oxide–semiconductor transistor is 1. This will lead to a number of switching events proportional to number of 1s in the input or secret data. The number of 1s in a data word X is called the *Hamming weight* (HW) of X or $HW(X)$. Now we can develop a model parameterized by $HW(X)$ for secret data X. The classification classes can also be labeled by the HW of each class. In this case, we would be able to classify the HW of the secret data, but not necessarily the complete value. But that is close enough in many instances, allowing for some other, simpler attack to determine the rest of the information.

Hamming distance (HD) is another parameter often used in these algorithm-blind attacks. The HD between two values X and Y, denoted as $HD(X, Y)$ is the number of bit positions in which X and Y differ, mathematically captured as $\sum_{i=0}^{n}(X_i \oplus Y_i)$. In CMOS static logic, if a communication bus or function unit uses X followed by Y, the number of induced switching events equals $HD(X, Y)$, leading to the intuition behind the model.

SIMPLE POWER ANALYSIS

A simple power analysis attack [12] models the power measurements during cryptographic operations with detailed knowledge of the cryptographic algorithm implementation. A skilled adversary monitors only one or a few power signature vectors during cryptographic operations to infer the secret key. This scenario is not practical for modern complex hardware implementations, since it is very difficult to obtain detailed information on the effective capacitance and resistance of the internal nodes of a processor or the application-specific integrated circuit implementation.

Profiling, however, makes the power attacks feasible. In the profiling phase, an adversary estimates the probability distribution of the power consumption vector for the secret key value space. The adversary records many power trace vectors at the specific times when cryptographic operations with intermediate values related to the secret key are performed. Multiple power trace vectors capture ambient noise, measurement apparatus noise, and timing uncertainty noise to create a robust distribution for the profile model.

A nonprofiling attack exploits a model with generic parameters such as the HD and HW of the secret data. This attack needs little knowledge of the implementation of the cryptographic algorithm and can be performed with relatively cheap equipment. It is known to be a major threat to cryptographic devices such as smart cards or embedded systems. Figure 2 shows a summary of these attack classes. $\mathcal{L}$ models the leakage function based on the unknown secret key k^* and the adversary-picked plaintext P. ϵ models all the noise. $\mathcal{L}$ is just the sampled power vector.

DIFFERENTIAL POWER ANALYSIS ATTACK

There is a general attack strategy that is used by all differential power analysis (DPA) attacks. Figure 2(b) illustrates it. We will discuss this through the example of the AES encryption algorithm. In the 128-b AES, the plaintext and the key are 16 B or 128 b long and are organized into a 16-B 4 × 4 array. We will denote the ith key block by KB_i for $0 \leq i \leq 15$. The plaintext data P to be encrypted can also be organized as 16-B blocks PB_i. In a typical round of AES, key block KB_i is XORed with the plaintext block PB_i or, in general, state block SB_i as $SB_i = SB_i \oplus KB_i$. This operation is called `AddRoundKey`. XOR is a low-energy operation, since there is not even a carry ripple between bits. Hence, the power observability for this operation is low; it can be drowned in the noise.

Hence, the first step of the DPA attack is to determine an intermediate value of the AES cryptographic algorithm that is a function of both PB_i and KB_i but generates a stronger power signature. In AES, `AddRoundKey` is followed by `SubBytes`. `SubBytes` substitutes a block of state $SB_i = SB_i \oplus KB_i$ with a table look-up where $SB_i = SBox[SB_i]$. The current value of the state block is used as an address to look up a table called $SBox$. Note that the address for this lookup $SB_i = PB_i \oplus KB_i$ in the first round is a known function of the plaintext the adversary picks and the secret key block KB_i. Hence, knowing $PB_i \oplus KB_i$ also reveals KB_i. Addresses to a memory (where $SBox$ lookup tables are kept) switch considerably more capacitance than the XOR operation of `AddRoundKey`, generating a more observable power signature. Hence, from the adversary's viewpoint, it is easier to observe the `SubBytes` operation than the `AddRoundKey` operation.

In general, the intermediate value $v_i = f(d_i, k^*)$, as a function of d_i (the ith plaintext or cipher text) and k^* (the secret key), is identified for a more robust power observation.

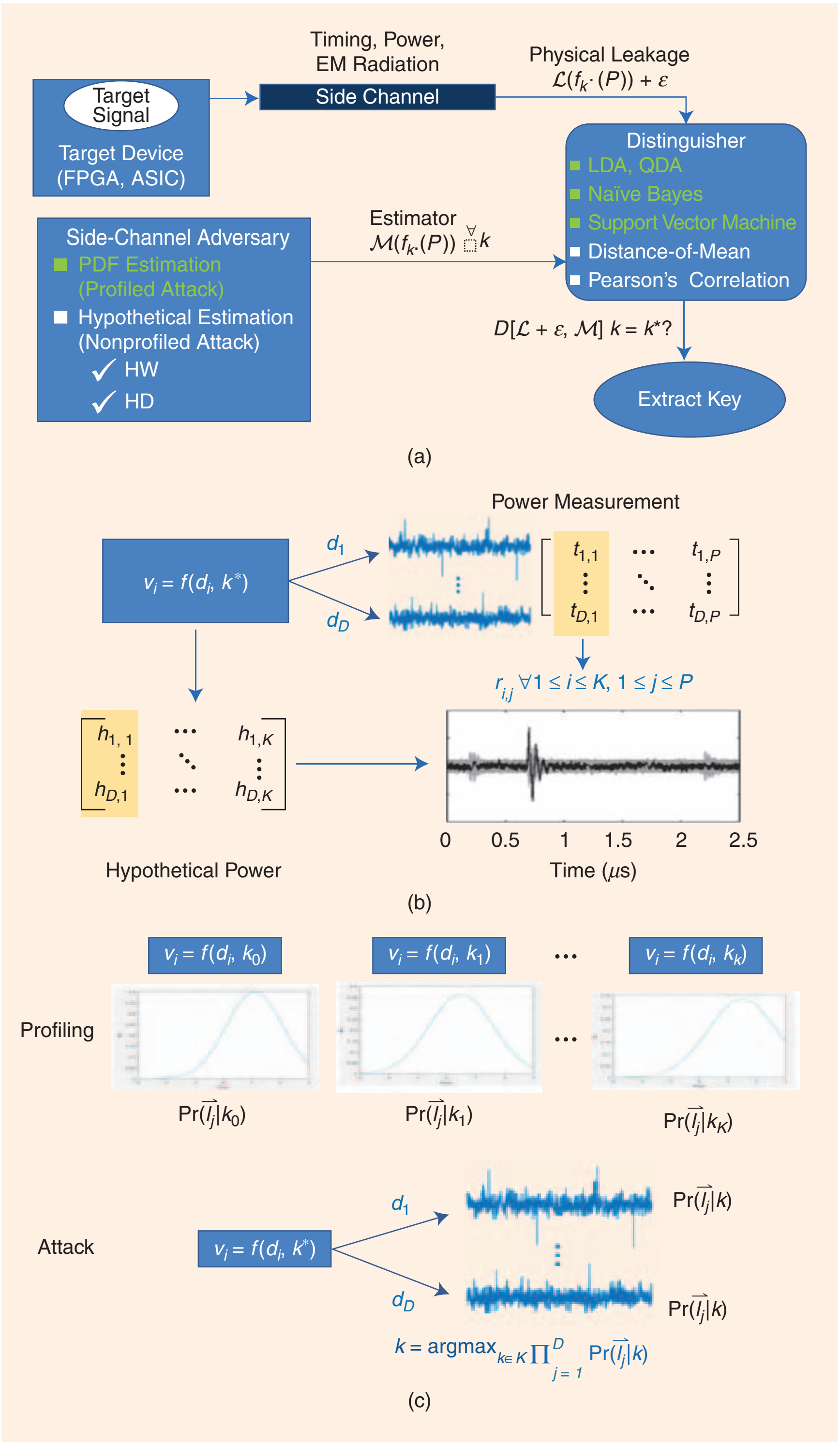


FIGURE 2. Some schematics showing (a) side-channel attacks, (b) a differential power analysis (DPA) attack, and (c) a profiling attack. FPGA: field-programmable gate array, ASIC: application-specific integrated circuit, PDF: probability density function, QDA: quadratic discriminant analysis.

The second step is profiling to measure the power consumption of the cryptographic device while it encrypts or decrypts D different data blocks. This power observation must include the selected intermediate function from

Typical power vectors can have hundreds of sampling points or features, resulting in a 100-plus-dimensional space for classification!

the first step. We denote the power trace vector as $\vec{t_i} = (t_{i,1}, t_{i,2}, \ldots, t_{i,t^*}, \ldots, t_{i,P})^T$, corresponding to data block d_i, where P denotes the number of time samples in the vector, and t_{i,t^*} is the power consumption when the selected intermediate function from the first step is computed. Note that the power event that occurs at time t^* spreads in the time domain due to the capacitance of the V_{dd} and GND nodes—hence, the need to measure the power in a time window around the actual computation event. An adversary measures a trace for each of the D data blocks, and hence the traces can be written as matrix $\mathbf{T}$ of size $D \times P$:$\mathbf{T} = (\vec{t_1}, \vec{t_2}, \ldots, \vec{t_{t^*}}, \ldots, \vec{t_P})$, where $\vec{t_j}$ for $j = 1, \ldots, P$ is a column vector of size $D \times 1$.

For the AES-based example, let us perform this profiling with a length of three traces $(P = 3)$ and two data values $(D = 3)$. Let the power trace vector for the first data value of 100 (plaintext block $PB_i = 100$; binary: 01100100) be [50,75,100]. Note that we assume 8-b power samples with a range of [0,255]. Let the second data value be 200 (binary: 11001000), with the corresponding power trace vector being [52,78,99]. Then the trace matrix is ([50,75,100], [52,78,99]).

The third step is to calculate the hypothetical intermediate values for all possible combinations of the secret key block KB_i and data values PB_i. In general for the AES algorithm, KB_i will take on all 256 possible values of 0–255. But if we know something about the key selection algorithm, we may be able to limit our search to fewer possible values. Let us assume that our hypothesis for the key block KB_i is limited to two choices 127 (binary: 01111111) and 65 (binary: 01000001). The intermediate-value matrix $v_{i,j}$, where rows correspond to data values and columns to key values, can be computed for AES as the binary matrix $([00011011, 00100101]^T, [10110111, 10001001]^T)$. Note that each entry is the XOR of the row data value and the column key value in binary. This intermediate-value matrix in decimal will be $([27,37]^T, [183,137]^T)$.

The fourth step is to map the hypothetical intermediate values $v_{i,j}$ to the hypothetical power consumption values. The power consumption of the targeted computation cycle when the intermediate value is produced is targeted. Let $g(v)$ be the abstract function that gives us a power trace vector of the intermediate value v's computation. Then we compute $g(v_{i,j}) = g(f(d_i, k_j))$ for $i = 1, \ldots, D$ and $j = 1, \ldots, K$. This is where either algorithm-specific power models or generic *HW* or HD-based power models can be used. The $D \times K$ matrix $\mathbf{H}$ is constructed at this step: $\mathbf{H} = (\vec{h_1}, \ldots, \vec{h_K})$. Note that each entry in this matrix is a power vector.

Let us assume that our power vectors have length 3. Let the power trace vector for $PB_i = 100$ and $KB_i = 127$ be [50,75,100]; for $PB_i = 200$ and $KB_i = 127$, let the power trace vector be [52,78,99]; for $PB_i = 100$ and $KB_i = 65$, let the power vector be [30,45,30]; and for $PB_i = 200$ and $KB_i = 65$, let the power trace vector be [32,48,30].

The fifth step is to compare the hypothetical power consumption model with the measured power trace vectors. This requires comparing each hypothetical power vector $\vec{h_i}$ with the collected power trace $\vec{t_j}$ for similarity. For a given set of data values {100, 200} in our AES example, we measure power vectors for each data value either once or multiple times to even out the noise contribution. In the following, we assume only one measurement per data value. Let the trace vector matrix for data values {100, 200} be ([50,75,100],[52,78,99]). For a given data value, say, 100, we correlate its hypothetical power vector from step 4 for the column given by $d_i = 100$, which includes power vectors for all key value hypotheses [50,75,100],[30,45,30], against the power trace vector [50,75,100] corresponding to data value 100. If the correlation against one of the hypothesis power vectors [50,75,100],[30,45,30] is high, the corresponding key (the row in the $v_{i,j}$ hypothesis matrix from step 4) is guessed. In this case, the first row, corresponding to $KB = 127$, will have a perfect match and hence a correlation equal to 1. Hence, the attack will guess the key to be 127.

Mathematically, to measure the linear relationships between two vectors $\vec{h_i}$ and $\vec{t_j}$ for $i = 1, \ldots, K$ and $j = 1, \ldots, P$, the Pearson correlation coefficient is calculated: $r_{i,j} = \left[\sum_{d=1}^{D} (h_{d,i} - \overline{h_i})(t_{d,j} - \overline{t_j})\right] \Big/ \left[\sqrt{\sum_{i=1}^{D} (h_{d,i} - \overline{h_i})^2 \sum_{i=1}^{D} (t_{d,j} - \overline{t_j})^2}\right]$. This correlation reports a perfect match (a coefficient value of 1) if each component of the two vector variables x, y are linearly related. First, each variable is normalized by using $(x_i - \bar{x})/\sigma(x)$. This measures the distance from the mean as a multiple of the standard deviation of each component. This removes dc shifts in the measured power. If, for each component i, $(y - \bar{y})$ equals $k * (x_i - \bar{x})$, the linear relationship between normalized components, then the Pearson correlation coefficient will be a perfect 1: $\left(\sum k * (x_i - \bar{x}) * (x_i - \bar{x})\right) \Big/ \left[\sqrt{\sum (x_i - \bar{x})^2} * \sqrt{\sum k^2 * (x_i - \bar{x})^2}\right]$. This equals 1.

In general, if r_{k^*,t^*} of the correct key k^* and the specific time t^* has the distinct peak value, the DPA attack succeeds with key guess k^*. These attacks are sometimes also called nonprofiling attacks, since they need not develop a detailed power profile. The next section addresses profiling-based attacks.

PROFILING ATTACKS

As the name connotes, a power profile for the attacked crypto algorithm's secret data is built in advance. If detailed implementation capacitances and resistances are known, this profile could even be constructed through simulation program with integrated circuit emphasis-level simulations. The simplest profile will develop the distribution of the power trace

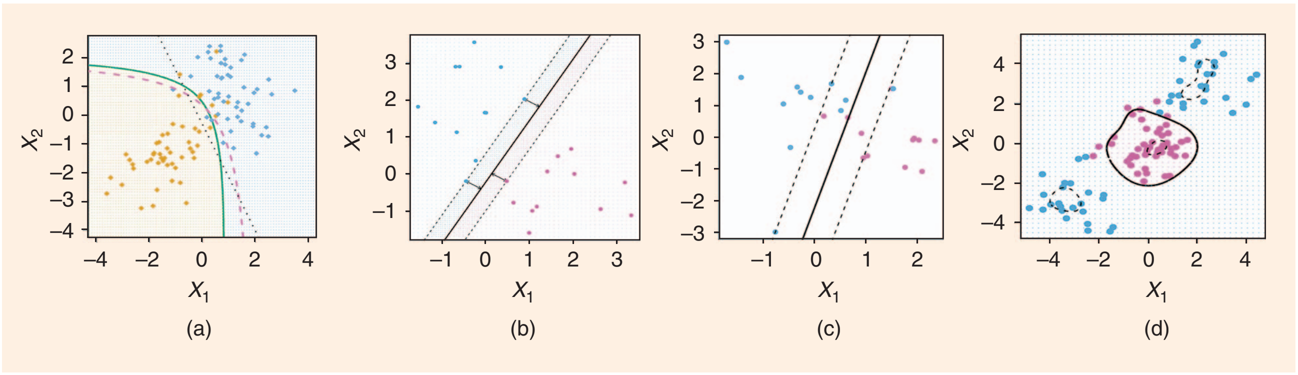


FIGURE 3. The classification performance of various machine-learning classifiers: (a) naïve Bayes (purple dashed line), LDA (black dotted line), and QDA (solid green line), and three support vector machines: (b) a linear classifier (separable), (c) a linear classifier (nonseparable), and (d) a nonlinear classifier.

vectors denoted as leakage $\vec{l}$ for a given value for the secret k, $\Pr[\vec{l}_j \mid k]$. For instance, for the AES example discussed earlier, 256 such distributions would need to be developed, one for each possible value for KB_i (0–255). In fact, one distribution for each sampling point or the feature for each of the 256 KB_i values is needed. If we continue with three-dimensional power trace vectors such as $[50, 75, 100]$, we will need to profile $256 * 3 = 768$ such distributions. Figure 2(c) shows such distributions for $K+1$ different key values.

Bayesian inference allows one to develop a distribution for the key values, given a specific leakage—a power trace vector such as $[50, 75, 100]$. Specifically, the posterior probability that the secret key is equal to k, given any measured power ($\vec{l}_j$), can be computed using Bayes' theorem: $\Pr[k \mid \vec{l}_j] = [\Pr[\vec{l}_j \mid k]\Pr[k]] \big/ \left(\sum_{k=1}^{K} \Pr[\vec{l}_j \mid k]\Pr[k]\right)$.

Bayes' theorem derives the likelihood that the given power leakage $\vec{l}_j$ comes from the key value k as $\Pr[k \mid \vec{l}_j]$. For a causal relationship, causes result in effects. The distribution of effects can be measured by triggering the causes. Bayes' theorem derives the inverse relationship between effects and causes; given an effect, it infers which cause is likely to have triggered it.

Using the maximum-likelihood estimation, the best key guess is the key value k that leads to the maximum probability $k = \operatorname{argmax}_{k \in \mathcal{K}} \prod_{j=1}^{D} \Pr[k \mid \vec{l}_j]$. If the prior probability $\Pr[k]$ for $k = 1, \ldots, K$ is uniformly distributed, we get $k = \operatorname{argmax}_{k \in \mathcal{K}} \prod_{j=1}^{D} \Pr[\vec{l}_j \mid k]$. The likelihood probability $\Pr[\vec{l}_j \mid k]$ determines the kind of classifier. The successful classifier selects the correct key: $k = k^*$. We next discuss a variety of classifiers.

SIDE-CHANNEL ANALYSIS USING MACHINE-LEARNING CLASSIFIERS

In general, there exists a profile of power trace vectors that were derived from a key value $KB_i = 100$, and there are power vectors that were derived with key values $KB_i \neq 100$. Figure 3(a) shows yellow and blue dots representing these two classes $KB_i = 100$ and $KB_i \neq 100$ in two-dimensional power vectors. The question we need to ask is, given a measured two-dimensional (number of sampling points) power vector $[50, 75]$, does it belong to the yellow class ($KB_i = 100$) or the blue class ($KB_i \neq 100$).

The question gets more complicated when the yellow and blue dots run into each other, creating an ambiguous cloud. Machine-learning classifiers are designed to answer this question or to classify a new data point (e.g., power vector $[50, 75]$) into one of the classes. Of course, in general, there exist more than two classes—in fact, 256 classes for the AES key block value—and the dimensionality of the points is much larger than two. Typical power vectors can have hundreds of sampling points or features, resulting in a 100-plus-dimensional space for classification! A short description of some machine-learning classifiers follows.

The naïve Bayes classifier is the classifier we described for the profiling attack. Abstractly, a classifier defines a boundary to separate two classes. As we see in Figure 3(a), some points are misclassified (blue points in the mass of yellow points). Figure 3(a) shows the classification boundary for this classifier. Naïve Bayes is usually computationally expensive.

The LDA classifier defines a boundary between two classes that is a linear function of the feature axes. Figure 3(a) shows the black dotted line defined by an LDA classifier. In higher-dimensional spaces, the boundary would be a hyperplane. For some problems, a large number of points could be misclassified with a linear boundary. LDA is a poor match for those problems. LDA is usually one of the most computationally efficient classifiers, due to linear computation. The quadratic discriminant analysis (QDA) classifier defines a quadratic boundary as shown in Figure 3(a) as a solid green line. The classification accuracy depends on the data set. QDA performs well for instruction-level disassembly classification.

Support vector machines (SVMs) were introduced by Vapnik [13]. They became more important and popular in recent years when extensions to general nonlinear SVMs were made. The linear SVM classifier (separable case) defines an optimal separating hyperplane that maximizes the distance between the hyperplane and the nearest points on either side, as shown in Figure 3(b). The linear SVM classifier

We conducted preliminary experiments to check if similar-style instructions of AVR ATmega328p μC can be disassembled through power analysis.

(nonseparable case) handles cases where the two classes are not linearly separable; the hyperplane that incurs the least error should be searched.

Nonlinear SVM classifiers handle nonlinear data sets. They map the data set to a higher dimensional feature space by doing a nonlinear transformation. After the nonlinear mapping, a construction of the linear separating hyperplane is done in this high-dimensional feature space. SVM performs well for instruction disassembly classification but at a high computational cost.

SIDE CHANNEL-BASED DISASSEMBLER OF AVR MICROCONTROLLER

The main focus of the side channel-based disassembler is to extract assembly-level code along with the control flow graph from the side-channel leakage. The significant difference between side-channel analysis attacks and the side channel-based disassembler is the number of available power sample traces, assuming that both use profiled templates. Side-channel analysis attacks for secret-data leakage have more flexibility in the number of available sample traces, because the adversary can control the plaintext input of the target device. The adversary can perform multiple experiments with different plaintext data while the secret data remain invariant. But the side-channel disassembler does not have similar controllability of the target device. It should recognize a power or EM trace of each executed instruction. It does not get to repeat the power trace of the same instruction. In other words, the side-channel disassembler should estimate which instruction is executing, which register is used, or what value is processed with only one power sample. This makes the side-channel disassembler a more challenging problem. It requires more advanced estimation techniques.

There exist many challenging problems in complete disassembly. Identification of the destination register `Rd` and source register `Rs` for register transfer instructions or data for load or store instructions is difficult. For a more complete monitoring of programs, many variables, such as register names, register data, memory address, and values for load/store instructions, should be estimated.

PRELIMINARY EXPERIMENTS

We conducted preliminary experiments to check if similar-style instructions of AVR ATmega328p μC can be disassembled through power analysis. We considered six data transfer instructions (`add`, `sub`, `and`, `mov`, `or`, and `eor`) from the source register (`Rs16 ~ Rs25`) to the destination register (`Rd16 ~ Rd25`). The goal of this experiment is to identify which instruction is executed along with its destination register `Rd` identity and source register `Rs` identity.

The AVR μC has two pipeline stages with a clock frequency of 16 MHz. A Tektronix DPO-4032 oscilloscope is used to sample the power pin at 1.25 GS/s, 20-MHz bandwidth, 1,000 sample points, and 128 average mode. Using this oscilloscope, the voltage of the shunt resistor between the GND pin and ground is measured. Each power trace is measured with the following program segment template: `sbi, 5 nops, targeted profiled instruction, 5 nops`. The `sbi` instruction is executed for the trigger signal. To remove power consumption of `sbi` instruction and electrical noise, we compute the difference between each power trace and the reference power traces of `sbi` and `10 nops` sequence. For profiling, 3,000 power traces for each instruction with randomly selected `Rs` and `Rd` (the values of the `Rs` and `Rd` also are randomly distributed) are sampled. We also measure 3,000 power traces per unique `Rd`, with randomly selected instruction opcode and `Rs`; and 3,000 power traces per unique `Rs`, with randomly selected instruction opcode and `Rd`. These training data will be used for the classification. There exist three different class groups. The first class group represents the instruction opcode: $C_{\text{inst}} = \{c_{\text{add}}, c_{\text{sub}}, c_{\text{and}}, c_{\text{mov}}, c_{\text{or}}, c_{\text{eor}}\}$. The second and third class groups represent the source register and the destination register-based separation, respectively: $\boldsymbol{C_{Rd}} = \{\boldsymbol{c_{rd16}}, \ldots, \boldsymbol{c_{rd25}}\}$, $\boldsymbol{C_{Rs}} = \{\boldsymbol{c_{rs16}}, \ldots, \boldsymbol{c_{rs25}}\}$.

INSTRUCTION CLASSIFICATION PROFILE

It is difficult to distinguish between the `add` and `sub` instructions in the time-domain power trace. To determine if the two instructions have distinct features in the frequency domain, two power traces are mapped into the two-dimensional time–frequency region by continuous wavelet transform (CWT). The power traces of the `add` and `sub` instructions with identical register operands and data values in the time domain lack any distinguishing features. However in the time–frequency domain, at the time interval $(320, 335), (358, 375)ns$, the two power signals have distinct, significant differences. Similarly, significant differences exist based on register names and data values in the time–frequency domain.

The wavelet transform has several advantages for side-channel analysis. It is used to remove noise from side-channel leakage traces obtained by oscilloscopes and to perfectly align collected traces [14]. Based on positive results with CWT, we use the wavelet transform to extract distinct features among all instruction classes to discriminate or classify instructions within a program segment. Principal components (time and frequency) are extracted from the wavelet transform of the collected traces. Only the principal time and frequency region features are kept, and all the other time and frequency domain signals are zeroed. An inverse CWT of the shaped time and frequency signal contains only the principal features in the time domain. A clean separation between the instructions exists after this transform. General classifiers, such as the maximum a posteriori classifier or the correlation coefficient classifier, can then be used to distinguish the inverse transformed signals in the time domain.

INSTRUCTION CLASSIFICATION

The next step is the feature selection to identify specific times that are significant. The total number of sampling points for each inverse-CWT power trace is 160. Assuming that each sampling point has a normal distribution, with the mean μ_i and the variance σ_i^2 for $i = 1, \ldots, 160$, the probability distribution of each class has the multivariate (160-dimensional) normal distribution. The computation complexity with this high-dimensionality feature space is very high and not practical. Thus, dimensionality reduction or feature selection is required.

The Kullback–Leibler (KL) divergence is a useful metric for the feature selection. The more the KL divergence between two random variables, the more distinguishable two random variables are. The specific sampling points should have a large KL-divergence value. An additional desirable property is that the specific sampling points do not have dependency (or collinearity). The specific sampling points have a locally maximum value to satisfy the two conditions. As a result, 160 dimensions can be reduced to about ten. The scatter plots of the raw power traces for the `add` and `sub` instructions are overlapped. After the CWT analysis and feature selection, the scatter plots do not have an overlapping region and are cleanly separated.

We classified the low-dimensional power signatures of `ADD`, `SUB`, `AND`, `OR`, and `MOV` instructions using a Bayesian classifier based on a multivariate Gaussian model. Table 1 shows the recognition rate when the power traces of the row's instruction are distinguished from the column's instruction. The row instruction's test vector is classified against the column instruction's reference vector.

We next discuss statistical machine-learning classifiers for instruction disassembly. Three thousand power traces, with the specific sample points for each class, are used for the training, depending on the classifier. The LDA, QDA, and naïve Bayes methods are executed. Each classifier is based on different assumptions. LDA assumes that the distribution of each class has a multivariate normal distribution with the same covariance matrix (Σ). QDA has more flexibility than LDA, since it assumes that the distribution of each class has a multivariate normal distribution with a different covariance matrix ($\Sigma_i \neq \Sigma_j \ \ \forall i \neq j$). The naïve Bayes classifier assumes that the probability distribution of each specific sampling point of each class can be various distributions independently. The marginal probability distribution of power traces at a specific sampling point resembles the normal distribution, and the marginal probability distribution of each class has a different variance. Since the characteristic of power traces satisfies the assumptions of QDA, the QDA classifier has the best performance among the three classifiers (LDA, QDA, and naïve Bayes). The successful recognition rates of the instructions (`add`, `sub`, `and`, `mov`, `or`, and `eor`) for these classifiers are shown in Table 2.

REGISTER CLASSIFICATION

The registers from `Rd16` (or `Rs16`) to `Rd25` (or `Rs25`) can be grouped into four classes depending on the *HW* of the binary address of the register. The *HW* of the register address is very related to the power consumption during the fetch and decoding of the instruction, since the addresses of the registers occupy 10-b length of the 16-b instruction code. The classification of registers (`Rd`, `Rs`) can be executed hierarchically. The *HW* of the address of the register is identified, and then the address of the register in the *HW* class is recognized.

The successful recognition rates of the *HW* of `Rd` and `Rs` using the QDA classifier are 80% and 69.6%, respectively. The address of the register `Rd` and `Rs` with the 2 *HW* is recognized at the rate of 77.8% and 67.5%, respectively. The address of the register `Rd` and `Rs` with the 3 *HW* is recognized at the rate of 83% and 73.6%, respectively. The destination register `Rd` has a better recognition rate than the source register `Rs`, probably because `Rd` is used twice for each instruction: once for one source operand and written once for the result.

SUPPORT VECTOR MACHINE

The least squares SVM (LS-SVM) [15] is used to classify instructions. Figure 4 shows the successful recognition rates of the LS-SVM and QDA to classify measured power traces into

Table 1. A Bayesian classifier's recognition rate when the row instruction's power traces are distinguished from the column's instruction.

Reference Vector / Test Vector	ADD	SUB	AND	OR	MOV
`add`	—	100%	90.1%	97.2%	99.9%
`sub`	98.5%	—	86.7%	97.4%	67.2%
`and`	56.3%	100%	—	82.4%	98.7%
`or`	84%	89.4%	76.2%	—	71.2%
`mov`	97.3%	56.5%	97.1%	82%	—

Table 2. The successful recognition rate of instructions according to classifiers.

Classifier	Successful Recognition Rate
LDA	37%
QDA	70.1%
Naïve Bayes	37.1%

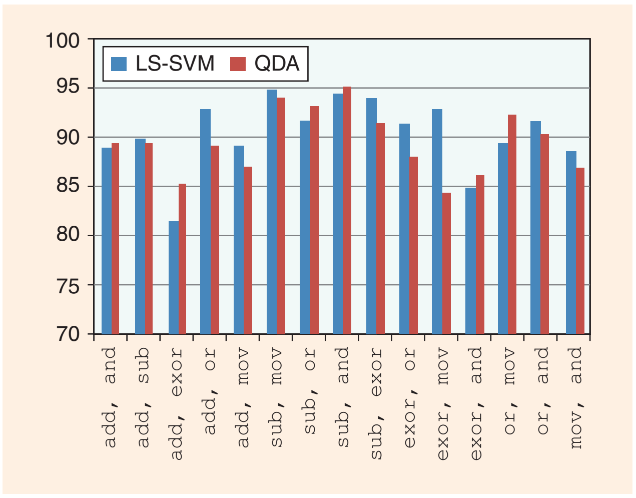


FIGURE 4. The LS-SVM classifier versus QDA.

two classes. The LS-SVM frequently exceeds the QDA classifier in terms of the successful recognition rate. In the case of $C = \{c_{\text{exor}}, c_{\text{mov}}\}$, the LS-SVM results in 8.5% better performance than the QDA classifier. The LS-SVM increases the successful recognition rates by 12% for the six instructions (`add`, `sub`, `and`, `mov`, `or`, and `eor`) compared with the QDA results.

CONCLUSION

In embedded devices, CE devices, and IoT devices, side channels pose at least as big a security threat as traditional network channel-mounted attacks. The power side channel exhibits highly correlated power signatures or events at the V_{dd} or GND pin of a processor caused by the secret cryptographic computation events occurring within a processor. We introduce the basic concepts relevant to power side-channel analysis. Many statistical machine-learning techniques have recently been applied to power side-channel analysis. We introduce some of these techniques. A more recent emerging threat within the power side-channel domain is that of instruction-level disassembly of the program solely through power signatures. Even a physically tamper-proofed processor with a strong operating system is not immune from such an attack. We describe our preliminary experiences with instruction-level disassembly of an AVR microcontroller. We are able to identify an instruction with over 80% accuracy through the power side channel.

ACKNOWLEDGMENT

We are grateful to the National Science Foundation for its support under grant CNS 1441640 and to the Semiconductor Research Corporation for its support under contract 2014-TS-2556.

ABOUT THE AUTHORS

Jungmin Park (jungminpark@ufl.edu) earned his B.S. and M.S. degrees in electrical engineering from Kyunghee University, Seoul, South Korea, in 2007, his M.S. degree in computer engineering from Kyunghee University in 2009, and his Ph.D. degree in computer engineering from Iowa State University, Ames, in 2016. He is currently working as a postdoctoral associate in the Department of Electrical and Computer Engineering, University of Florida, Gainesville. His research interests include side-channel attacks, side-channel analysis-resistant hardware design, and hardware Trojan and logic obfuscation.

Akhilesh Tyagi (tyagi@iastate.edu) earned his Ph.D. degree in computer science from the University of Washington, Seattle, in 1988. He was with the Computer Science Department at the University of North Carolina, Chapel Hill, before moving to Iowa State University in 1993. He is currently with the Department of Electrical and Computer Engineering, Iowa State University, Ames. His research interests include hardware security (power side channel), computer architecture, and low-energy computing.

REFERENCES

[1] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *J. Cryptographic Eng.*, vol. 1, no. 1, pp. 5–27, 2011.
[2] D. Agrawal, J. Rao, and P. Rohatgi, "Multi-channel attacks," in *Cryptographic Hardware and Embedded Systems—2003*, vol. 2779, C. D. Walte, Ç. K. Koç, and C. Paar, Eds. Berlin: Springer-Verlag, 2003, pp. 2–16.
[3] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems—2008*, vol. 5154, E. Oswald and P. Rohatgi, Eds. Berlin: Springer-Verlag, 2008, pp. 411–425.
[4] T. Eisenbarth, C. Paar, and B. Weghenkel, "Transactions on computational science x," *Building a Side Channel Based Disassembler*, M. L. Gavrilova, C. J. K. Tan, and E. D. Moreno, Eds. Berlin: Springer-Verlag, 2010, pp. 78–99.
[5] M. Msgna, K. Markantonakis, and K. Mayes, "Precise instruction-level side channel profiling of embedded processors," in *Information Security Practice and Experience*, vol. 8434, X. Huang and J. Zhou, Eds. Cham: Springer, 2014, pp. 129–143.
[6] D. Strobel, F. Bache, D. Oswald, F. Schellenberg, and C. Paar, "Scandalee: A side-channel-based disassembler using local electromagnetic emanations," in *Proc. Design, Automation, and Test in Europe Conf. and Exhibition (DATE)*, Mar. 2015, pp. 139–144.
[7] H. Liu, H. Li, and E. Y. Vasserman, "Practicality of using side-channel analysis for software integrity checking of embedded systems," in *Security and Privacy in Communication Networks 2015*, vol. 164, B. Thuraisingham, X. Wang, and V. Yegneswaran, Eds. Cham: Springer, 2015, pp. 277–293.
[8] D. McCann, C. Whitnall, and E. Oswald. (2016). Elmo: Emulating leaks for the arm cortex-m0 without access to a side channel lab. Cryptology ePrint Archive, Report 2016/517. [Online]. Available: http://eprint.iacr.org/2016/517
[9] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Mateo, CA: Morgan Kaufmann, 2011.
[10] ARM Corporation. (2012). Cortex-A9 MPCore technical reference manual revision: r4p1. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0388e/index.html
[11] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology, CRYPTO 96*, vol. 1109, N. Koblitz, Ed. Berlin: Springer-Verlag, 1996, pp. 104–113.
[12] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 19th Annu. Int. Cryptology Conf. Advances Cryptology*, M. J. Wiener, Ed. Berlin: Springer-Verlag, 1999, pp. 388–397.
[13] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[14] N. Debande, Y. Souissi, M. A. E. Aabid, S. Guilley, and J.-L. Danger. (2012). Wavelet transform based pre-processing for side channel analysis. *Proc. 2012 45th Annu. IEEE/ACM Int. Symp. Microarchitecture Workshops*. [Online]. pp. 32–38. Available: http://dx.doi.org/10.1109/MICROW.2012.15
[15] K. Leuven. (2011, Aug.). LS-SVMlab v1.8. [Online]. Available: http://www.esat.kuleuven.be/sista/lssvmlab/