Template Attacks

Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi

IBM Watson Research Center,
P.O. Box 704
Yorktown Heights, NY 10598
{schari,jrrao,rohatgi}@us.ibm.com

Abstract. We present *template attacks*, the strongest form of side channel attack possible in an information theoretic sense. These attacks can break implementations and countermeasures whose security is dependent on the assumption that an adversary cannot obtain more than one or a limited number of side channel samples. They require that an adversary has access to an identical experimental device that he can program to his choosing. The success of these attacks in such constraining situations is due manner in which noise within each sample is handled. In contrast to previous approaches which viewed noise as a hindrance that had to be reduced or eliminated, our approach focuses on precisely modeling noise, and using this to fully extract information present in a single sample. We describe in detail how an implementation of RC4, not amenable to techniques such as SPA and DPA, can easily be broken using template attacks with a single sample. Other applications include attacks on certain DES implementations which use DPA–resistant hardware and certain SSL accelerators which can be attacked by monitoring electromagnetic emanations from an RSA operation even from distances of fifteen feet.

1 Introduction

In the past few years, side channel attacks [13,12] have shown to be extremely effective as a practical means for attacking implementations of cryptographic algorithms. Adversaries can obtain sensitive information from side channels such as timing of operations[13], power consumption [12], electromagnetic emanations [19,9,20] etc. In constrained devices such as chip–cards, straightforward implementations of cryptographic algorithms can be broken with minimal work.

Since Paul Kocher's original paper [12], a number of devastating attacks, such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) have been reported on a wide variety of cryptographic implementations [15,18, 6,11,17,8,3,4,10,16,7,5,21]. In SPA, keying information is easily extracted from a single sample due to leakage from the execution of key dependent code and/or the use of instructions which leak substantial information in the side channel over the noise. When the leakage relative to noise is much less, statistical techniques such as DPA are applicable. DPA relies on a statistical analysis of a *large number of samples* where the same keying material is used to operate on different data. A large number of samples is used to reduce noise by averaging.

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 13–28, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In this paper, we show that these attacks are not optimal as they do not take advantage of all information available in *each* side channel sample. Consequently, some implementations believed to be immune to side channel attacks simply because the adversary is limited to one or at most a few compromising samples, can in reality be broken by harnessing all available information.

Consider an implementation of the RC4 stream cipher. While there are recent reports of cryptanalytic results highlighting minor statistical weakness, there are no major statistical biases to be easily exploited by side–channel attacks. To our knowledge, no successful side channel attack on a reasonable RC4 implementation has been reported[1]. Initializing the 256-byte internal state of RC4 using the secret key is simple enough to be implemented in a key independent manner. While implementations of this will certainly leak some information about the key, the individual steps do not leak *enough* information. Thus, simple side channel attacks such as SPA are not possible. After initialization, the rapidly evolving internal state of the stream cipher, independent of adversarial action (due to the absence of any external inputs), offers innate defense against statistical attacks such as DPA. One can at most hope to obtain a *single* sample of the side channel leakage during the key initialization phase of RC4. Figure 1 is based on side channel samples from the RC4 key initialization phase: the upper trace is the difference between two single power samples when the keys are the same, the lower trace when they are different. Contrary to expectation, the first case shows larger differences. This ambiguity exists even when one looks at differences of averages of upto five invocations (as shown in Figure 3 in Section 3). Clear and consistent differences emerge only when one considers averages of several tens of samples. Therefore, it would appear that such a carefully coded RC4 implementation cannot be attacked using only *one* available sample.

Consider a smart card which has a fast[2] hardware DES engine. These have become very popular especially because they have been evaluated and shown to be highly resistant to side channel attacks. In conjunction with protocols limiting adversaries to only few DES invocations with the card's secret key, it would appear that this is immune to side channel attacks. The third case we consider is where an adversary is able to position sensitive (and bulky) electromagnetic (EM) eavesdropping equipment in the proximity of a server with a commercial RSA accelerator inside. In this environment, due to a risk of detection, it is likely that only a few samples can be obtained.

In all these cases, the adversary has to work with far fewer signals than is believed is necessary for side channel attacks based on known techniques. The *template attacks* introduced in this paper can break all of these implementations. In fact, as we will see, the template attack extracts all possible information available in each sample and is hence the strongest form of side channel attack possible in an information theoretic sense given the few samples that are available.

A key requirement for the template attack is that the adversary has an identical experimental device which can be programmed. While such an assumption

[1] IEEE 802.11 uses RC4 in a mode which makes implementations vulnerable to DPA.
[2] Typically, such engines perform the entire DES in a few cycles.

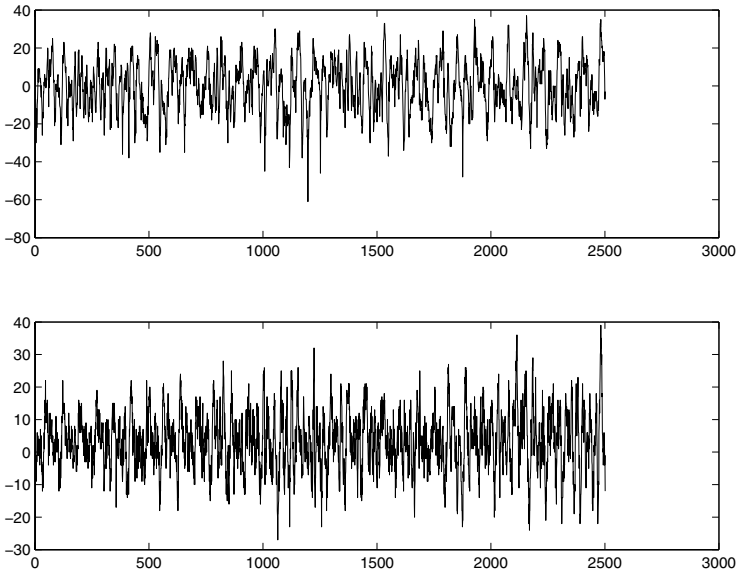

Fig. 1. Differences of side channel samples: upper figure is for the same key while the lower is for two different keys

is limiting, it holds in many cases and has been used in other side channel attacks [8,18] before. The template attack derives its power from using the experimental device to derive a precise multivariate characterization of the noise. In sharp contrast, prior approaches focussed on eliminating noise by averaging. We argue that, especially for cryptographic algorithms implemented in CMOS devices, the use of such a characterization is an extremely powerful tool to classify even a single sample. The situation is analogous to the manner in which very weak signals are extracted in signal communications. Even though the received signal strength is very weak, it can be extracted by a receiver who has a very good characterization of the signal and the ambient noise.

We refer to the precise, detailed models of the signal and noise as the *template* of the computation. The concept of a template is based on Signal Detection and Estimation Theory and in particular, the use of information theoretic techniques such as likelihood ratios for hypothesis testing. Although other techniques such as DPA, can also be viewed as coarse approximations of likelihood ratios, the use of multivariate noise statistics is key to extracting the maximum information from a single sample. Empirically, we have observed that in several situations, univariate statistics are not sufficient and yield poor results.

The template attack works by a process of iterative classification. At each step, we unroll one more segment of the sample which uses more bits of the unknown key. Correspondingly, larger templates are used to prune the space of possible hypotheses for the values of key bits, while controlling error probability.

While minor differences in the keys can possibly confuse a classifier, this attack is effective on cryptographic algorithms because the natural diffusion properties of cryptographic algorithms actually aid in eliminating precisely such mistakes.

The paper is organized as follow: Section 2 introduces the theory behind template attacks. Section 3 describes the application of template attacks to extract keys from an implementation of RC4 using a single sample. Section 4 describes two other cases where template attacks are feasible. In Section 5, we describe the implications of template attacks and discuss potential countermeasures.

2 Theory

In this section, using Signal Detection and Estimation Theory we derive the template attack and describe some heuristics to make the attacks practical. Essentially, we have a device performing one of K possible operation sequences, $\{O_1, \ldots, O_K\}$: for example, these could be to execute the same code for different values of key bits. An adversary who can sample the side channel during this operation wishes to identify which of the operations is being executed or to significantly reduce the set of possible hypotheses for the operation.

In signal processing, it is customary to model the observed sample as a combination of an intrinsic signal generated by the operation and noise which is either intrinsically generated or ambient. Whereas the signal component is the same for repeated invocations of the operation, the noise is best modeled as a random sample drawn from a noise probability distribution that depends on the operating and other ambient conditions. It is well known[22] that the optimal approach for the adversary, who is trying to find the right hypothesis given a single sample S, is to use the maximum likelihood approach: The best guess is to pick the operation such that, the probability of the observed noise in S is maximized. Computing this probability requires the adversary to model both the intrinsic signal and the noise probability distribution for each operation accurately.

Template attacks meld this basic principle with details of the cryptographic operation being attacked. The adversary uses an experimental device, identical to the device under test, to identify a small section of the sample S depending only on a few unknown key bits. With experimentation, he builds templates corresponding to each possible value of the unknown key bits. The template consist of the mean signal and noise probability distributions. He then uses these templates to classify that portion of S and limit the choices for the key bits to a small set. This is then repeated with a longer prefix of S involving more key bits. We will retain only a small number of possibilities for the portion of the key considered thus far. Thus template attacks essentially use an extend-and-prune strategy directed by the single sample S to be attacked: we use increasingly longer prefixes of S and the corresponding templates to prune the space of possible key prefixes. The success critically depends on how effectively the pruning strategy reduces the combinatorial explosion in the extension process.

Template attacks are particularly effective on implementations of cryptographic algorithms on CMOS devices due to their *contamination* and *diffusion*

properties. Contamination refers to key dependent leakages which can observed over multiple cycles in a section of computation. In CMOS devices, direct manipulation of the key bits makes them part of the device state and these state leakages can persist for several cycles. Additionally, other variables affected by the key, such as key dependent table indices and values, cause further contamination at other cycles. The extent of contamination controls the success of the pruning of the fresh key bits introduced in the expansion phase. It is to be expected that if two keys are almost the same, that even with the effects of contamination, pruning at this stage, may not be able to eliminate one of them. Diffusion is the well-known cryptographic property wherein small differences in key bits are increasingly magnified in subsequent portions of the computation. Even if certain candidates for key bits were not eliminated due to contamination effects, diffusion will ensure that closely spaced keys will be pruned rapidly.

The implementation of an algorithm on a particular device inherently places theoretical bounds on the success of the template attack. The best any adversary can do to approach this theoretical bound is to have extremely good and accurate characterizations of the noise. While one get elaborately sophisticated with such characterizations, in practice approximations such as a multivariate Gaussian model for the noise distributions yields very good results.

2.1 The Multivariate Gaussian Model Approach

The steps in developing a Gaussian model are as follows:

1. Collect a large number L (typically one thousand) of samples on the experimental device for each of the K operations, $\{O_1, \ldots, O_K\}$.
2. Compute the average signal $M_1, \ldots, M_K$ for each of the operations.
3. Compute pairwise differences between the average signals $M_1, \ldots, M_K$ to identify and select only points $P_1, \ldots, P_N$, at which large differences show up. The Gaussian model applies to these N points. This optional step significantly reduces the processing overhead with only a small loss of accuracy.
4. For each operation O_i, the N–dimensional noise vector for sample T is $N_i(T) = (T[P_1] - M_i[P_1], \ldots, T[P_N] - M_i[P_N])$. Compute, the *noise covariance matrix* between all pairs of components of the noise vectors for operation O_i using the noise vectors N_is for all the L samples. The entries of the covariance matrix Σ_{N_i} are defined as:

$$\Sigma_{N_i}[u, v] = cov(N_i(P_u), N_i(P_v))$$

Using this we compute the templates (M_i, Σ_{N_i}) for each of the K operations. The signal for operation O_i is M_i and the noise probability distribution is given by the N–dimensional multivariate Gaussian distribution $p_{\mathbf{N_i}}(\cdot)$ where the probability of observing a noise vector $\mathbf{n}$ is:

$$p_{\mathbf{N_i}}(\mathbf{n}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma_{N_i}|}} \exp(-\frac{1}{2}\mathbf{n}^T \Sigma_{N_i}^{-1}\mathbf{n}), \quad \mathbf{n} \in \mathcal{R}^N \tag{1}$$

where $|\Sigma_{N_i}|$ denotes the determinant of Σ_{N_i} and $\Sigma_{N_i}^{-1}$ is its inverse.

In this model, the optimal technique to classify a sample S, is as follows: for each hypothesized operation O_i, compute the probability of observing S if indeed it originated from O_i. This probability is given by first computing the noise $\mathbf{n}$ in S using the mean signal M_i in the template and then computing the probability of observing $\mathbf{n}$ using the expression for the noise probability distribution and the computed Σ_{N_i} from the template. If the noise was actually Gaussian, then the approach of selecting the O_i with the highest probability is optimal.The probability of making errors in such a classification is also computable. If we use this approach to distinguish two operations O_1 and O_2 with the same noise characterization Σ_N, the error probability is given by:

Fact 1 *[22]For equally likely binary hypotheses, the error probability of error of maximum likelihood test is*

$$P_\epsilon = \frac{1}{2}\,\mathrm{erfc}\Big(\frac{\Delta}{2\sqrt{2}}\Big) \tag{2}$$

where $\Delta^2 = (M_1 - M_2)^T \Sigma_N^{-1}(M_1 - M_2)$ *and* $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$.

To implement the pruning process of the template attack, we deal with multiple hypotheses and bound the probability of classification errors by judiciously selecting a small subset of possible operations as most likely candidates.

2.2 The Pruning Process

In the extend-and-prune paradigm of the template attack, each extension results in several hypotheses about the operation being performed. For the attack to be tractable, the pruning process has to reduce the set of possible hypotheses to a very small number while ensuring with high probability that the correct hypothesis is *not* discarded. To achieve this we ensure that the cumulative probability of the hypothesis not retained is within the desired error bound. While this can be done exactly given the precise noise characterizations, several heuristic methods are easier to implement and give good results.

One approach that works well is to scale the probabilities so that the noise probabilities under all of the hypotheses add up to one. We then discard those hypotheses with the lowest scaled probabilities till the cumulative probability of error due to the discarded hypotheses reaches the desired error bound.

Another heuristic that is easy to implement is to fix a constant factor c and only retain those hypotheses in the pruned set whose noise vector probabilities are within this constant fraction c of the highest noise probability: that is, if P_{max} is the maximum noise probability, we keep all hypotheses whose noise probability is at least $\frac{P_{max}}{c}$. We refer to this pruning process as the *ball approach*. The intuition for this heuristic is that if the noise characterization is approximately the same for all hypotheses then the logarithm of the noise probability for a hypothesis is a measure of the distance between the received signal and that hypothesis. The misclassification error is an inverse exponential function of the distance between hypotheses. The heuristic is to create a ball centered at

the received sample whose radius is $d_{min} + \log(c)$, where d_{min} is the shortest distance between the sample and the nearest hypothesis. We then retain only those hypotheses that fall into this ball. Under the assumption of approximately similar noise characterizations, the worst case probability of error can be shown to be bounded by $\mathcal{O}(\frac{\infty}{\sqrt{c}})$ [22]. In practice, the error is much better. In subsequent sections, we will illustrate how this theory can be applied to a number of case studies including an RC4 implementation.

3 Case Study: RC4

We describe a template attack on an implementation of RC4. RC4 is a stream cipher operating on a 256-byte state table. The state table is used to generate a pseudo-random stream of bytes that is then XOR'ed with the plaintext to give the ciphertext. It is a popular choice in a number of products and standards. RC4 uses a variable key length (from 1 to 256 key bytes) to update the 256-byte state table (initially fixed) using the pseudo code below:

```
index1 = index2 = 0;
for (counter = 0; counter < 256; counter++)  {
  index2 = (key[index1] + state[counter] + index2) % 256;
  swap_byte(&state[counter], &state[index2]);
  index1 = (index1 + 1) % key_data_len;
}
```

A portion of the corresponding side channel sample, in this case the power consumption, is shown in Figure 2. The repeated structure observed is exactly five successive iterations of the loop. As described in Figure 1, two keys cannot be distinguished on the basis of a single sample. In fact, this remains true if one were to consider the averages of five samples as illustrated in Figure 3. However, significant and widespread differences become apparent when examining averages of several dozen samples (see Figure 4).

A well-designed system using RC4 is unlikely to permit an attacker to repeatedly obtain samples of identical state initialization computations[3]. Thus the real challenge is to break this implementation using a single sample. The figures clearly show that traditional attacks like SPA will not work and DPA is clearly not an option since we can not obtain more than a single sample.

RC4 is, however, an ideal candidate for template attacks. It is evident from inspecting the code snippet above, that the key byte used in each iteration causes substantial contamination. The loading of the key byte, the computation of `index2` and the use of `index2` in swapping the bytes of the state table all contaminate the side channel at different cycles. The extent of this contamination is easily visible as significant and widespread once averages of a large number of samples are taken. Further, the use of `index2` and the state in subsequent

[3] Note that in devices implementing the 802.11 standard the key initialization is done repeatedly with a fixed secret key and a variable part.

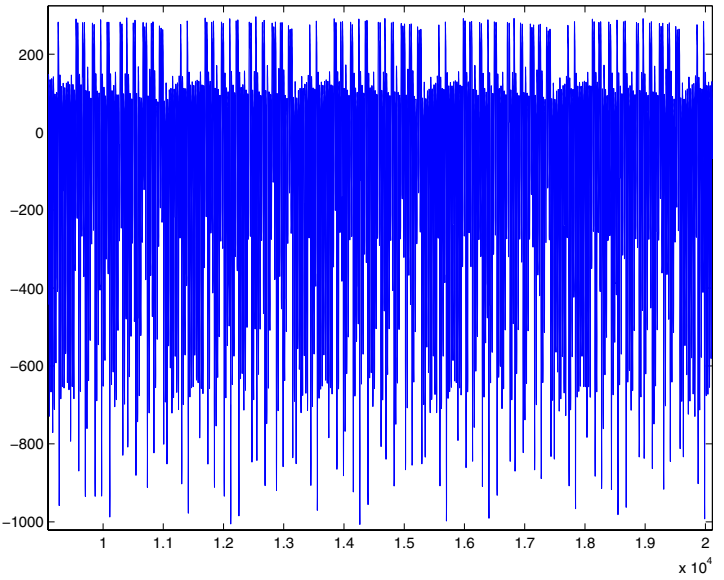

Fig. 2. Power sample during first 5 iterations of RC4 state initialization loop.

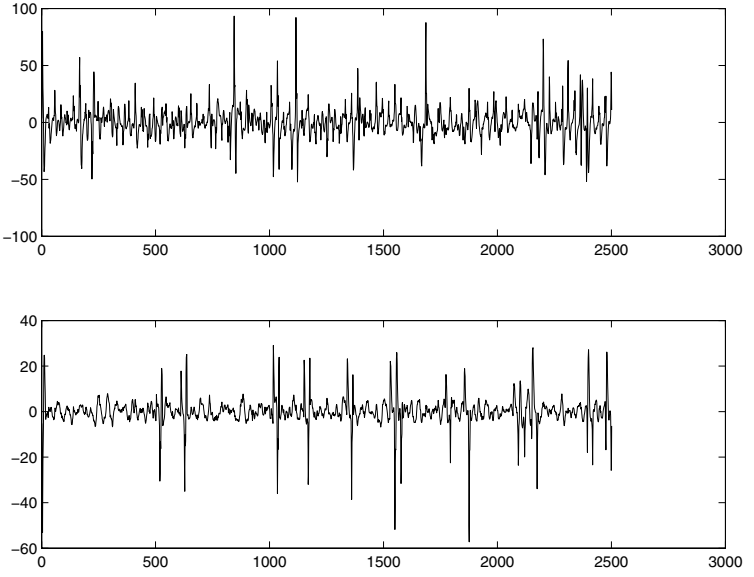

Fig. 3. Differences of averages of 5 side channel samples: upper figure is for the same key while the lower is for two different keys

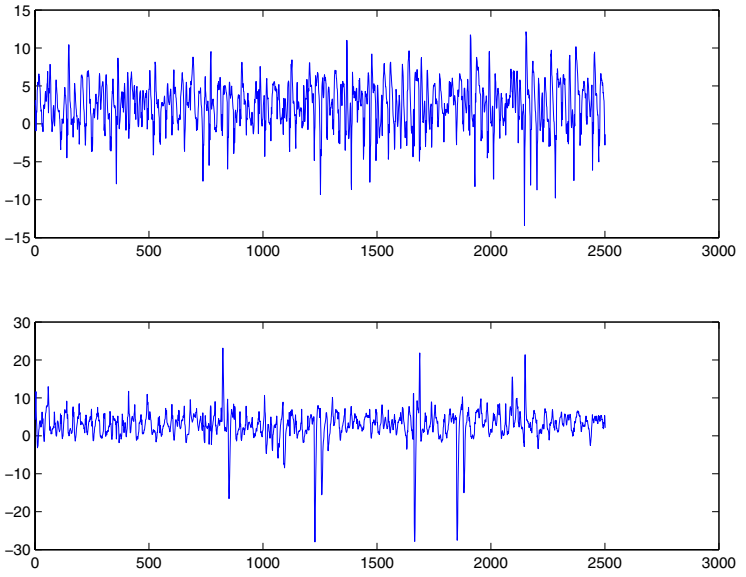

Fig. 4. Differences of averages of 50 side channel samples: upper figure is for the same key while the lower is for two different keys

iterations, and the fact, that RC4 is a well-designed stream cipher, quickly propagates small key differences to cause diffusion. Thus, one expects that templates corresponding to different choices of key bytes are very different and can be used to efficiently and effectively classify a single sample.

3.1 Template Attack on RC4

Inspecting the averaged RC4 side channel samples using several different keys, we identified 42 points in the side channel sample for each iteration of the loop for classification purposes. Our first attempt used statistical measures that treated these 42 points independently, i.e, we only looked at means and standard deviations of the samples at each of the points. Although encouraging, the results have high classification errors (as much as 35%) for pairs of keys with few bit differences. Some empirical classification results of samples with five different keys using this approach are shown in Fig 5. When key bits are very different, even this simplistic approach gives us 100% success rate. However, in general, this approach is unsuitable for an extend-and-prune attack due to high errors. In the worst case, a large number of keys close to the actual key would be retained. Empirically, we have observed that this could be of the order of a few tens of keys. This is because even if keys have the same Hamming weight, the state table addressing part distinguishes keys with different higher order bits. We use the approach described in Section 2 to launch an attack based on multivariate

Key Byte	1111 1110	1110 1110	1101 1110	1011 1110	0001 0000
1111 1110	0.86	0.04	0.07	0.03	0.00
1110 1110	0.06	0.65	0.10	0.20	0.00
1101 1110	0.08	0.16	0.68	0.09	0.00
1011 1110	0.10	0.11	0.08	0.71	0.00
0001 0000	0.00	0.00	0.00	0.00	1.00

Fig. 5. Classification Probability of 5 competing hypotheses using univariate statistics. Entry (i,j) is probability of classifying samples with key i as one with key j.

statistics with the Gaussian noise. For our experiment, we used 10 choices for the first key byte, as shown in Fig. 6. They are carefully chosen to be very close and yielded poor results with the univariate statistics. For each key byte, we computed the mean of 2000 samples of the side channel. We used the same 42 points of interest as in the univariate experiment. The templates consisted of the means and the noise covariance at these points. To obtain statistics on how well this approach would work, we used the templates to classify tens of thousands drawn using one of the 10 choices as the first key byte. Figs. 6 and 7 summarize the results of the classification experiments for this set of 10 key choices. Since the values were carefully chosen to reflect the worst case, these results can be extrapolated to the case of 256 different values of the key byte. Fig. 8 is an extrapolation of our results for the case of 256 different templates by making pessimistic assumptions about the number of "close" keys. In practice the actual results should be much better.

Our first classification heuristic was to retain only the most likely hypothesis i.e. with highest likelihood probability. Even with such a drastic pruning approach, average classification success probability was 99.3% with these 10 hypotheses and worst–case probability was 98.1%. Detailed results are described in column 1 of the Fig. 6. We can get reasonable results even if we use this extensive pruning strategy in each iteration of the extend-and-prune approach. Extrapolating, as shown in Fig. 8 we expect average error probability of the closest hypothesis approach to be about $5-6\%$ when we consider all 256 possible values, since we pessimistically expect around $50-60$ keys to be "close" to any key. Bounding the error probability over many iterations by the sum of error is in each iteration we note that when the number of key bytes is small this can be used to extract all key bytes. For example, we can do better than 50% for about 8 bytes of key material.

With a little more effort, much better results can be obtained by using the *ball approach* to pruning as shown in columns 2, 3 and 4 of Fig. 6 showing success probability of retaining the correct hypothesis for balls with different values of ball size. Average success probability has improved and is better than 99.8% and the worst–case probability is 99.5%, for this set of samples. As shown in Fig. 7 the average number of hypotheses that we retain is still close to 1 for balls of size e^6 and e^{12}. Again, using an estimate of about $50-60$ close keys, we can extrapolate these results as done in Fig. 8. For example, choosing the ball size e^6, with good probability, at the end of one iteration we expect to retain at

Key Byte	Ball Size $c = 1$	Ball Size $c = e^6$	Ball Size $c = e^{12}$	Ball Size $c = e^{24}$
1111 1110	98.62	99.46	99.88	99.94
1110 1110	98.34	99.82	99.88	99.88
1101 1110	99.16	100.00	100.00	100.00
1011 1110	98.14	99.52	99.82	100.00
0111 1110	99.58	99.76	99.89	99.94
1111 1101	99.70	99.94	99.94	99.94
1111 1011	99.64	99.82	99.82	99.89
1111 0111	100.00	100.00	100.00	100.00
1110 1101	99.76	99.82	99.88	99.88
1110 1011	99.94	100.00	100.00	100.00
Average	99.29	99.81	99.91	99.95

Fig. 6. Percentage of samples for which the correct hypothesis is retained under different ball sizes with 10 competing hypotheses

most 2 hypotheses, yet we are guaranteed to retain the correct hypothesis with probability at least 98.67%. Using this approach independently in each iteration, we can correctly classify keys of size n bytes with expected probability around $(100 - 1.33n)\%$ and the number of remaining hypotheses would grow no more than $(1.5)^k$, which is substantially than the 2^{8k} (the entropy of the key). The next subsection describes experiments where we use larger templates comprising multiple iterations to get better pruning.

Ball Size $c = 1$	Ball Size $c = e^6$	Ball Size $c = e^{12}$	Ball Size $c = e^{24}$
1	1.041	1.158	1.842

Fig. 7. Expected number of hypotheses retained under different ball sizes for 10 competing hypothesis.

	Ball Size $c = 1$	Ball Size $c = e^6$	Ball Size $c = e^{12}$	Ball Size $c = e^{24}$
Success Prob.	95.02	98.67	99.37	99.65
Retained Hypotheses	1	1.29	2.11	6.89

Fig. 8. Extrapolated results for 256 competing hypotheses.

Iteration. Instead of using the above attack *independently* on subsequent portions of the sample which use the next key byte, it is advantageous to consider the basic attack on the whole prefix of the sample including previous iterations. In our RC4 implementation, we now use 84 points of interest in the sample spread over two loop iterations to prune possible candidates for the first two key bytes. After pruning hypotheses for the first byte (as described earlier), we extend extend each remaining hypothesis by all 256 possible values for the second

key byte. 84–point templates are created for this set of possibilities. Using these larger templates, we classify the sample and retain only those hypotheses for the first two bytes which remain in the ball around the sample.

Key Byte	Ball Size $c = 1$	Ball Size $c = e^6$	Ball Size $c = e^{12}$	Ball Size $c = e^{24}$
1101 1110 1111 1110	99.16	99.58	99.70	99.94
1101 1110 1110 1110	98.10	99.53	99.88	99.94
1101 1110 1101 1110	99.46	99.88	99.94	100.00
1101 1110 1011 1110	98.80	99.64	99.89	100.00
1101 1110 0111 1110	99.33	99.70	99.88	99.94
1101 1110 1111 1101	99.88	99.94	99.94	99.94
1101 1110 1111 1011	99.03	99.58	99.70	100.00
1101 1110 1111 0111	99.94	100.00	100.00	100.00
1101 1110 1110 1101	99.82	99.82	99.88	99.88
1101 1110 1110 1011	99.94	100.00	100.00	100.00
1011 1110 1111 1110	96.81	99.05	99.65	100.00
1011 1110 1110 1110	99.76	99.88	99.88	99.94
1011 1110 1101 1110	98.57	99.82	100.00	100.00
1011 1110 1011 1110	97.87	99.76	100.00	100.00
1011 1110 0111 1110	98.25	99.28	99.52	99.82
Average	98.98	99.70	99.86	99.96

Fig. 9. Percentage of samples for which the correct hypothesis is retained for 2 iterations and 15 competing hypotheses

Ball Size $c = 1$	Ball Size $c = e^6$	Ball Size $c = e^{12}$	Ball Size $c = e^{24}$
1	1.04	1.141	1.524

Fig. 10. Expected number of hypotheses retained under different ball sizes for 15 competing hypotheses.

To verify that this is better than using just 42 points of the second iteration, we performed the following experiment: We considered 15 possible combinations of the first and second key bytes. There were two possibilities for the first key byte which roughly simulates what we expect after the template attack on the first iteration. The classification results for different ball sizes are given in Figs. 9 and 10. To compare these results to the earlier case, we must scale the results from the tables for 10 hypotheses. Since there are 14 wrong hypothesis instead of 9 before comparison each figure from the earlier tables must be scaled by a factor of 14/9. This is indeed the case for the error probability of not retaining the correct hypothesis for ball sizes of 1,e^6 and e^{12}. The error probability is better for ball size of e^{24} when we use longer templates. More importantly, note that the average number of hypotheses retained is substantially better uniformly for each and every ball size. Thus, we retain fewer number of candidates by using a longer template. Empirically, we have observed that with a longer template, with

extremely high probability, all of the hypotheses remaining have the correct first key byte. After 2 iterations, we could only find 1 sample amongst 16000 where a hypothesis with the wrong first key byte was retained.

4 Other Case Studies

We briefly describe two other examples where template attacks can be used. The first example is a smart card with fast DES hardware. Use of such hardware is currently very popular, since they highly resistant to power attacks. The only exposure for such engines is the loading of the key bytes from EEPROM which usually leaks the hamming weight. However, implementors also have to worry about Differential Fault Attacks [2,1], and a card that we looked at addressed this problem using a checksum on key bytes, verified before the key was loaded in the DES engine. The checksum was calculated accessing key bytes in circular order starting from a random offset. This can be easily fixed using signal processing. Fig. 11 shows an averaged signal depicting the checksum computation loop in more detailfollowed by the enabling of the DES engine (the region with large current consumption). In the checksum calculation, the key bytes are loaded from EEPROM, placed in RAM and are then used as operands in the checksum computation, thus creating a large contamination. For example, Figure 12 shows that even key bytes with the same hamming weight can be distinguished by taking averages of a few samples. Such types of signals are prime candidates for a template attack.

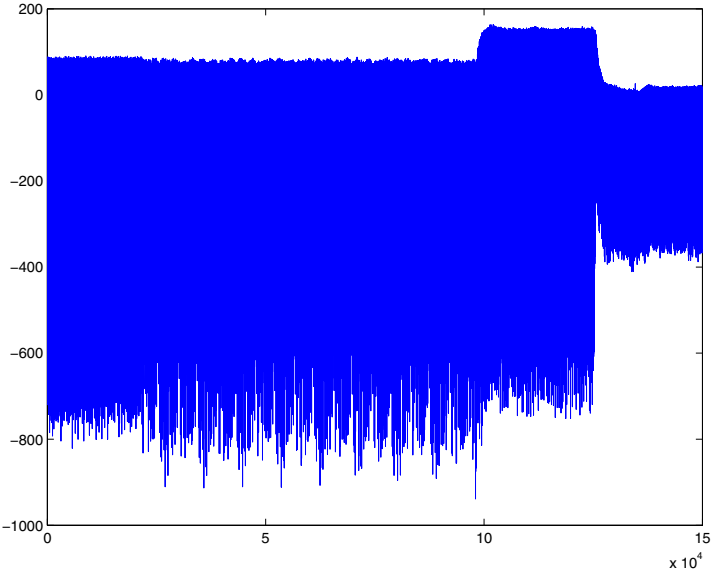

Fig. 11. Average signal showing details of checksum calculation for 8 key bytes.

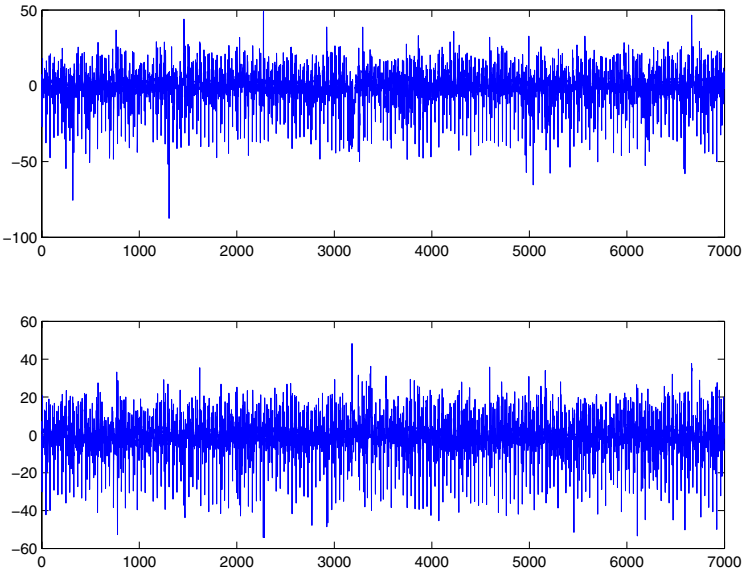

Fig. 12. Differences in average signals: lower figure is for same key and upper figure is for different keys but with same hamming weight.

The second example is a EM signal we collected from a distance of 15 feet away from an SSL accelerator inside a closed server. We programmed the SSL accelerator to do a 2048 bit exponentiation with a single-nibble exponent. The fact that the exponent leaks from this computation follows from Figure13 which shows the signal differences between exponentiation with two different nibble exponents B and D after taking averages of few signals and some signal processing. Thus template attacks are very feasible for this case as well. In this example, other attacks such as MESD [18] can be possible as well, if one can collect a large number of EM samples.

5 Implications and Countermeasures

In principle, the template attack described is the strongest side channel attack possible from an information theoretic sense. The information present in each portion of the side channel signal is fully used for classification. This makes it a very powerful tool for attacking a wide range of cryptographic implementations. However, the effort required, in terms of creating a large number templates in an adaptive manner, make the task daunting. It also presumes the availability of an identical test device which can be programmed to the adversary's whim. Since the attack requires only a single invocation of the test device, all countermeasures to side channel attacks that rely on limiting the number of samples that an adversary can take with a fixed key may be vulnerable depending on the extent

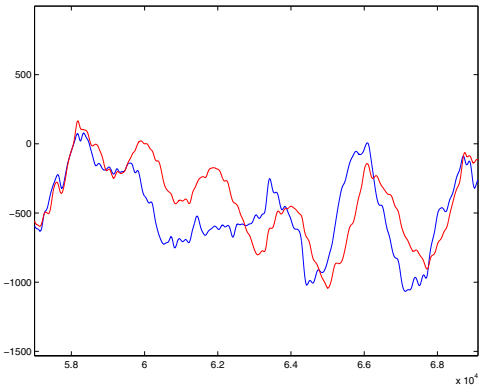

Fig. 13. Average processed EM signals for two diffrent private exponents.

of contamination. Such countermeasures include high level protocols to limit key usage and non-linear key update techniques of [12].

The requirement of having an identical experimental device is also the weakness of the template approach. Randomization in the computation such as address/data scrambling, blinding/masking of data and key bits and ensuring that the adversary cannot control the choice of randomness in *his own experimental device* is one way this attack can be mitigated. This countermeasure may not be feasible for highly programmable devices such as SSL accelarators. The overriding principle in building in securing implementations against templates is to minimize contamination caused by use of sensitive information in the clear.

References

1. Ross Anderson and Markus Kuhn. Low Cost Attacks on Tamper Resistant Devices. In Proc. Security Protocols, 5th International Workshop, Paris, France, Springer-Verlag LNCS Volume 1361 , pp 125–136, April 1997.
2. Dan Boneh, Richard DeMillo and Richard Lipton. On the Importance of Checking Cryptographic Protocols for Faults. Journal of Cryptology, Springer-Verlag, Volume 14, Number 2, pp 101–119, 2001.
3. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi. Towards Sound Countermeasures to Counteract Power–Analysis Attacks. Proc. Crypto '99, Springer–Verlag, LNCS 1666, August 1999, pages 398–412.
4. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi. A Cautionary Note Regarding the Evaluation of AES Candidates on Smart Cards. Proc. Second AES Candidate Conference, Rome, Italy, March 1999.
5. Christopher Clavier, Jean-Sebastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 2000, LNCS 1965, Springer-Verlag, pp 252–263.

6. Jean-Sebastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Proc. Workshop on Cryptographic Hardware and Embedded Systems , Aug. 1999, LNCS 1717, Springer-Verlag. pp 292–302.
7. Jean-Sebastien Coron, and Louis Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 2000, LNCS 1965, Springer-Verlag. pp 231–237.
8. P.N. Fahn and P.K. Pearson. IPA: A New Class of Power Attacks. In Proc.Workshop on Cryptographic Hardware and Embedded Systems, Aug. 1999, LNCS 1717, Springer-Verlag. pp 173–186.
9. K. Gandolfi, C. Mourtel and F. Olivier. Electromagnetic Attacks: Concrete Results. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Paris, France, May 2001.
10. L. Goubin and J. Patarin. DES and Differential Power Analysis. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 1999, LNCS 1717, Springer-Verlag. pp 158–172.
11. M.A. Hasan. Power Analysis Attacks and Algorithmic Approaches to Their Countermeasures for the Koblitz Curve Cryptosystems. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 2000, LNCS 1965, Springer-Verlag. pp 93–108.
12. P. Kocher, J. Jaffe and B. Jun. Differential Power Analysis: Leaking Secrets. In Proc. Crypto '99, Springer-Verlag, LNCS 1666, pages 388–397.
13. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In Proc. Crypto '96, LNCS 1109, Springer-Verlag, pp 104–113.
14. J. Kelsey, Bruce Schneier, D. Wagner and C. Hall. Side Channel Cryptanalysis of Product Ciphers. Journal of Computer Security, Volume 8, Number 2–3, 2000, pages 141–158.
15. Rita Mayer-Sommer. Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smart Cards. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, Aug. 2000, LNCS 1965, Springer-Verlag. pp 78–92.
16. Thomas S. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In Proc. Fast Software Encryption Workshop 2000, New York, NY, USA, April 2000, Springer-Verlag.
17. Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 2000, LNCS 1965, Springer-Verlag. pp 238–251.
18. T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smart Cards. In Proc. Workshop on Cryptographic Hardware and Embedded Systems 1999, Aug. 1999, LNCS 1717, Springer-Verlag. pp 144–157.
19. Jean–Jacques Quisquater and David Samyde. Simple Electromagnetic analysis for Smart Cards: New Results. Rump session talk at Cyrpto 2000.
20. Dakshi Agrawal, Bruce Archambeault, Josyula Rao, Pankaj Rohatgi. The EM Side-Channel(s). In Proc. Workshop on Cryptographic Hardware and Embedded Systems 2002, Aug. 2002,
21. Adi Shamir. Protecting Smart Cards from Power Analysis with Detached Power Supplies. In Proc. Workshop on Cryptographic Hardware and Embedded Systems, Aug. 2000, LNCS 1965, Springer-Verlag. pp 71–77.
22. H. L. Van Trees. Detection, Estimation, and Modulation Theory, Part I. John Wiley & Sons. New York. 1968.