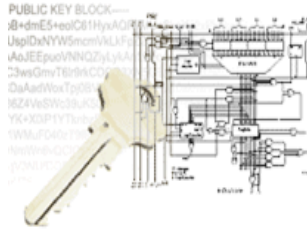


Information Security Course Work



A report on Timing Attacks

VERSION 1.11

Vasileios Lampos vl7342,
Stavros Syrimis ss7482,
and Christos Tsiotras ct7627

Article
IS - 4

<http://www.cs.bris.ac.uk/>

A report on Timing Attacks

Vasileios Lampos, Stavros Syrimis, and Christos Tsiotras

December 14, 2007

Abstract

A side channel attack tries to exploit specific properties of the implementation and operating environment of a cryptosystem rather than its mathematical specification [HMV04]. Timing attack is a class of side channel attack where the attacker tries to break an encryption algorithm by using information about the execution times of its encryption or decryption queries. In general, a timing attack tries to exploit private information from a system by timing specific system's operations. In this report, a vast amount of different types of timing attacks is described.

Keywords: timing attack, RSA, Montgomery reduction, CRT, OpenSSL.

1 Introduction

During the last eleven years several timing attacks have been proposed and most of them implemented successfully on various security protocols. The starting point was given by Kocher in 1996 when he first analyzed the possibility of breaking into a cryptosystem by using timing measurements on cryptographic operations. His idea was later on adopted as the basis for most of the forthcoming timing attacks on cryptographic schemes such as RSA in OpenSSL protocol. In 2000, a different type of timing attacks was presented; they were unrelated with Cryptography but able to impose user's privacy and security.

This report is organized as follows: In Section 2, starting from Kocher's first timing attack, we make a scroll through time presenting various types of timing attacks. In Section 3, we focus on timing attacks on SSL protocol and especially on the initiating attack of Brumley and Boneh whereas in Section 4 we refer to timing attacks on the World Wide Web. Finally, in Section 5, we come up with some conclusions regarding this class of side channel attack.

2 Timing attacks through time

2.1 The first timing attack

In 1996, Paul C. Kocher proved that knowing the amount of time required to perform private key operations on RSA or Diffie-Hellman protocol may result in breaking a cryptosystem. Kocher's attack is outlined in Figure 1 where the modular exponentiation algorithm that computes $m = c^d \pmod{N}$ is implemented; m , c , d , x , and N denote the message, the cipher text, the private exponent, the length of the private exponent d , and the public modulus respectively. This attack allows someone who knows the values of decryption key's bits $0 \dots b-1$ to decide the value of bit b .

```
s[0] = 1;
for (i = 0; i < x - 1; i++){
    if (d[i] == 1){ m[i] = (s[i] * c)%N; }
    else{ m[i] = s[i]; }
    s[i+1] = (m[i] * m[i])%N;
}
return m[x-1];
```

Figure 1: Kocher's timing attack on RSA, [Koc96]

Depending on the execution time t of $m_i = s_i \cdot c \pmod{N}$ two cases exist:

- (a) t is extremely slow whereas the modular exponentiation time is fast and,
- (b) the slow execution time t results in slow modular exponentiation times.

In case (a), the prediction for the next bit of the private exponent d_{i+1} is 0 when in case (b) it is 1. This method's correct guess probability is equal to $\Phi(\sqrt{\frac{j(b-c)}{2(w-b)}})$, where j is the number of timing measurements, b is the number of the private exponent bits for which a guess has been made, c is the number of the private exponent bits that are guessed correctly, w is the bit-length of the private exponent and $\Phi(x)$ is the area under the standard normal curve from $-\infty$ to x (*e.g.*, the expected Φ for $j = 250$, $b = 1$, $c = 0$, $w = 127$ is ≈ 0.84).

Kocher's attack was inefficient against RSA implementations that used Montgomery multiplications or the Chinese Remainder Theorem (CRT) to optimize their operations. However, he suggested that an attack on these may be possible as well. Countermeasures to Kocher's attack could be the execution of all operations in constant time or the enforcement of blinding techniques [Koc96].

2.2 The first timing attack on a smart card

In 1997, Dhem *et al* implemented a timing attack able to expose a 512-bit RSA key of a CASCADE smart card in a few minutes. Their method attacked the squaring operation of the square and multiply algorithm. Suppose that the first $i-1$ bits of the decryption key K have been extracted and a guess has to be made for the i^{th} bit. Firstly, the $i-1$ steps of the square and multiply algorithm (Figure 2) are executed; we stop just before the possible but unknown multiplication by M due to the bit K_j .

```
M_temp = M;
for (i = N - 2; i <= 0; i--){
    M_temp = power(M_temp,2);
    if(K[j] == 1){
        M_temp = M_temp * M;
    }
}
return M_temp;
```

Figure 2: Attack on Square and Multiply, [DKL⁺98]

If $K_j = 1$ then the multiplication $M_{temp} \cdot M$ is performed and before squaring it is decided whether an extra reduction is needed. The samples are divided in two subsets S_1 (extra reduction) and S_2 (no reduction). If $K_j = 0$ then there is no multiplication and just M_{temp}^2 is performed. The samples are divided again in two subsets S_3 (extra reduction) and S_4 (no reduction). If timing difference between the execution times of S_1 , S_2 ($T_{S_1-S_2}$) is significantly bigger than this of S_3 , S_4 ($T_{S_3-S_4}$) then $K_j = 1$, else $K_j = 0$. A major advantage of this attack is the 'embedded' error detection property. If $T_{S_1-S_2}$ is abnormally close to $T_{S_3-S_4}$, then the decision for K_j is not correct and thus the algorithm could go back one step or more correcting the wrong decisions for K 's bits. Dhem *et al* managed to extract an 128-bit key by using an average of 10.000 timings [DKL⁺98].

2.3 A timing attack on RC5

In 1999, Handschuh and Heys (HH) based on Kocher's statement that 'RC5 is at risk on platforms where rotations run in non-constant time' [Koc96], managed to perform a timing attack on RC5 in order to obtain the total amount of rotations carried out during an encryption. RC5 is a block, Feistel-like cipher that holds 3 parameters, w , r , and b which are the word size, the number of rounds performed, and the number of bytes for the secret key respectively. HH attacked on an RC5 algorithm with 64-bit block size (two words of 32-bit each), 12 rounds and 128-bit key. Their attack was concluded in two phases; in the first phase, they managed to extract the $\log_2 w$ least significant bits of the last half-round subkey by generating samples and using them for detecting encryption timings. In the

second phase, by applying the same techniques, they managed to extract all the bits of the sub-keys except for the first three one's which could be trivially determined using a small number of plaintexts and ciphertexts. The total complexity of the attack does not exceed $(2r-2) \cdot 2^{30+\log_2 w}$ [HH98].

2.4 Timing attacks against RSA with Chinese Remainder Theorem

In 2000, Schindler presented an attack on RSA versions that used CRT for exponentiation with the secret exponent and Montgomery for squaring. Schindler's attack was able to factorize the public RSA modulus and was based on the observation that the probability of an extra reduction step during a Montgomery multiplication (e.g., $g \cdot R^{-1} \bmod p$) is $P(g) = \frac{g \cdot \text{mod } p}{2 \cdot R}$. Using the ciphertext as an input, he measured the processing time of its exponentiation.

In Figure 3 we present the main routine of Schindler's attack. Suppose that R denotes the Montgomery multiplication parameter, N is the public RSA modulus, and $t_{reduction}$ denotes the extra time for a reduction during a Montgomery multiplication. In order to conform with Schindler's algorithm the following assumptions have to be made:

$$\beta = \sqrt{\frac{N}{R^2}}, \beta \cdot R \leq a < N, C = -t_{reduction} \cdot \frac{\log_2 \beta}{16}, \Delta = 2^{-6} \cdot R (= D).$$

```

a2 = a; a1 = a2 - D;
while( Time(a2) - Time(a1) > C ){ a2 = a1; a1 = a1 - D; }
while(u2 - u1 > 1000){
    a3 = floor((a1 + a2)/2);
    if ( Time(a2) - Time(a3) > C ){ a2 = a3; }
    else{ a1 = a3; }
}
for (i = 1; u1 > u2; i++){
    check_prime[i] = gcd(u1 + i, N);
}

```

Figure 3: Schindler's attack on RSA with CRT [Sch00]

Schindler's attack is a chosen plaintext attack which means that the attacker should be able to use the encryption / decryption device. A further explanation of the timing measurements is presented in Section 3.1 (Brumley and Boneh attack on OpenSSL, [BB05]). Schindler managed to factorize an 1024-bit RSA modulus by taking an average of 570 timing measurements. The most effective countermeasure against Schindler's attack would be

the application of blinding techniques (described in Section 3.1) [Sch00].

In 2005, Tomoeda *et al* by combining power analysis with timing measurements presented an improvement on Schindler's attack able to factorize the RSA modulus even if some blinding techniques were used. They did not measure the timing of the whole exponentiation process but only of the decryption process. Again, using a long-period random number for blinding could possibly make the attack useless [TMSK05].

2.5 A timing attack on SSH

In 2001, Song *et al* presented a timing attack on Secure Shell (SSH) network protocol. Their attack was based on two 'weaknesses' of SSH:

- the padding of the transmitted packets is not strong enough (if a block cipher is used, then the padding is only an 8-byte boundary) and information about the size of the original data is revealed,
- each keystroke of the user is transmitted in a different IP packet after the significant key is pressed and thus by monitoring a communication channel one could derive the exact length of *e.g.*, a user's password.

To perform this type of attack a large amount of statistical training data is essential. Song *et al* performed a statistical analysis by timing the latencies between two keystrokes for all the possible pairs of characters. They categorized each character pair (*e.g.*, pairs that could be typed with the right or left hand or both with high probability) and then formed a model in which the Latency L between two keystrokes of a Character Pair C has a Gaussian distribution:

$$P(L|C) = \frac{1}{\sqrt{2\pi}\sigma_C} e^{-\frac{(L-\mu_C)^2}{2\sigma_C^2}},$$

where μ_C is the mean value of the latency and σ_C the standard deviation for a character pair C . From their experiments it occurred that most of the times L was between 50 and 250 *ms* whereas if L was higher than 150 *ms* it was more possible that the key pair had been typed with one hand. This analysis was able to reveal 1 bit of information per password if the password's length was up to 7 - 8 characters. Knowing that the entropy of written English is 0.6 - 1.3 bits per character [Sha51] as well as the entropy of a password is 4 - 8 bits per character the information retrieved was significant. Song *et al* implemented a software (named as *Herbivore*) that monitors a communication channel and by using a Hidden Markov Model performs an exhaustive, improved search for users' passwords. They managed to exploit a password in 1.3 days when a real exhaustive search could take approximately 65 days (an Intel Pentium III was used) [SWT01].

2.6 Timing analysis on low-latency mix systems

In 2004, Levine *et al* presented a timing analysis on low-latency mix systems which are defined as systems that exchange messages in human-unnoticeable times by using communication proxies that attempt to mask the relation between incoming and outgoing messages. Their attack tried to correlate messages sent through a low latency channel consisting of multiple intermediate proxies (defined as 'mixes') and decide if their initiator is the same. They used a timing window of fixed duration T , and measured the number of packets that arrived to each proxy categorized by their initiator. By controlling the first mix of a communication channel and the last mix of another, an attacker who observes messages going through these mixes, is able to decide if the initiator of these messages is the same. The attack could be improved by dropping 'dummy' packets in the first mix which will create timing gaps and thus enhance the distinction between the first mixes of two different initiators. A possible defence to this enhancement would be to make an intermediate mix able to distinguish 'dummy' packets (by using *e.g.*, one more bit in the encryption layer) and discard them [LRWW04].

2.7 A timing attack on CIKS-1

In 2005, Furlong and Heys presented an attack on CIKS-1. CIKS-1 is a symmetric block cipher which consists of an 64-bit block, eight rounds and eight 32-bit subkeys. It uses four operations: fixed permutations, XOR and mod 2^2 addition, all of which take constant time to run, and Data-Dependent Permutation (DDP) operations which take variable times. DDP, when implemented in software (*e.g.*, a microcontroller found in smart cards), can reveal information about the Hamming weight of the control vector applied to it. As a result, in this attack, when a subkey is used as a control vector, its Hamming weight can be obtained through accurate measurements of the timing associated with each encryption. In order to attack CIKS-1 we focus on the DDP₂₁ block where two input bits i_0 and i_1 are considered for swapping under the control of a bit b :

```
if (b == 0) swap(i0,i1);
```

If b equals to 0, then the time to execute the instructions will be the time to evaluate the instruction plus the time to complete the swap; otherwise, the timing measurement refers to the evaluation of the Boolean expression. Consequently, if the timing information for a plaintext/ciphertext pair is known, then the relation of this pair with the number of DDP block swaps during the encryption process will reveal information about the Hamming weight of the expanded key [FH05].

3 Timing attacks on SSL

3.1 Brumley and Boneh timing attack on OpenSSL

Brumley and Boneh (BB) managed to attack OpenSSL 0.9.7. protocol by measuring the time an OpenSSL server takes to respond to decryption queries. The attacker manages to extract a 1024-bit RSA private key stored in a server by using approximately a million queries in a 2 hours process.

Their attack was based on two key facts. RSA in OpenSSL uses sliding windows exponentiation (SWE) algorithm to compute $c^d \bmod N$. During this algorithm there are many multiplications by c which are performed using Montgomery reduction algorithm. At the end of each Montgomery reduction it is checked whether the output is greater than the modulus used. If this is the case, then the modulus is subtracted from the output. This is an extra reduction step that may reveal timing information about the execution of the decryption algorithm. Schindler proved that the probability of an extra reduction in Montgomery is:

$$P[\text{reduction}_c] = \frac{c \bmod N}{2 \cdot R},$$

where $R > 2^{32 \cdot t}$, $t \in \mathbb{N}$ [Sch00] *Theorem 1*.

The first fact was based on the following hypotheses:

- if c is close to modulus, the number of Montgomery reductions increases.
- if c is a multiple of modulus, the number of Montgomery reductions tends to zero.

The second fact was based on the multi-precision integer multiplication routine performed in RSA. For an operation $x \cdot y \bmod N$, where the size of x , y is m , n words respectively, there are two hypotheses:

- if x and y have the same number of words, RSA uses Karatsuba routine which completes after $O(n^{1.58})$ operations.
- if x and y have different number of words, RSA uses a normal routine which completes after $O(m \cdot n)$ operations.

BB attack on OpenSSL uses Schindler's attack method on extra Montgomery reductions [Sch00, *Section 4*] and is based on building approximations of q , where q is a prime factor of modulus N . After recovering the half most significant bits of q , Coppersmith's algorithm is applied to complete factorization of N [Cop97] (*Section 11*). The initial guess of q is a number g between 2^{511} and 2^{512} . They measured the decryption times of all the possible combinations of the top 2 or 3 bits of g . The main routine used in the attack is presented in Figure 4. Supposing that the first $i-1$ bits of q are recovered, g is created by setting its first $i-1$ bits equal to the relevant bits of q and its rest bits equal to 0. In

Figure's 4 routine, g_{hi} is equal to g , with the i^{th} bit equal to 1. The representations of g and g_{hi} in Montgomery's 'world' are u_g and $u_{g_{hi}}$. The decryption times t_1, t_2 of $u_g, u_{g_{hi}}$ respectively are used to determine the value of the i^{th} bit of q .

```

g_hi = g; g_hi[i] = 1;
u_g = g * power(R,-1)%N; u_g_hi = g_hi * power(R,-1)%N;
t_1 = decrypt_time(u_g); t_2 = decrypt_time(u_g_hi);
if (large(t_1 - t_2) == TRUE){ q[i] = 0; }
else { q[i] = 1; }

```

Figure 4: BB timing attack on OpenSSL, [BB05]

The fact that sliding windows exponentiation algorithm uses a small number of multiplications results in inaccurate timing measurements. Thus BB used numbers close to g to form a neighborhood $g, g+1, g+2, \dots, g+n$ where a reasonable value of n is 100, and measured the total decryption time T_g . Then they replaced t_1, t_2 with $T_g, T_{g_{hi}}$ and improved the accuracy.

An attack scenario would be to have a client that sends a key-exchange message (t_{send}) to an OpenSSL server and substitutes the private key with g . Then the server responds with an alert message ($t_{receive}$) indicating that g is not valid. The time $(t_{receive})-(t_{send})$ corresponds to the decryption time of g .

There are two mentioned defences for the proposed attack. The most widely accepted defence is the use of blinding techniques. Let a be an integer such as $\gcd(a, N) = 1$. The blinding function B for a message m is $B(m) = a^e \cdot m \bmod N$, where e is the public key exponent. The unblinding function is $B^{-1}(m) = a^{-1} \cdot m \bmod N$ [Men97]. Since a is a random number, timing the decryption of $B(m)$ should not reveal information about the key. The second defence is the addition of 'dummy' extra reductions in the Montgomery reduction algorithm combined with the execution of multiplications using only Karatsuba's algorithm [BB05].

3.2 An improvement on BB attack

Aciımez *et al* improved the efficiency of BB timing attack on SSL by modifying two major operations of the algorithm. Firstly, they replaced g, g_{hi} with $\lfloor \sqrt{g} \rfloor, \lfloor \sqrt{g_{hi}} \rfloor$ in order to create a neighborhood. By preprocessing odd powers of the base c (SWE algorithm uses an odd sized window) and storing them in a table in order to use them repeatedly, Montgomery operations are completed faster. These procedures increase the efficiency of BB attack by a factor of ≈ 5.8 . Secondly, they managed to implement a better strategy for computing

the decryption time differences between two neighborhoods. The observation that a timing measurement of a previous round could be used in the next round with high probability resulted in removing the redundant queries from BB-attack and hence doubled the current performance. Consequently, they improved BB-attack by a factor of 10 [ASK05].

3.3 Branch prediction analysis

Computers translate conditional commands (*e.g.*, if-then-else) as branching points. During the execution of a program, the processor has to evaluate a conditional branching command and decide the correct execution path. For the period of this evaluation, a branch prediction algorithm tries to predict the path with the highest probability and 'in a way' continues the execution of the program. If the prediction is correct, then the program execution continues. Otherwise, we have a miss-prediction and the branch prediction algorithm steps are ignored. A branch buffer is a buffer where the CPU stores the addresses of the previously used branches.

Aciğmez *et al* presented a trace driven attack against the Branch Target Buffer [ASK07]. Their attack addressed the branch buffer by implementing an iterative spy process able to capture it. When the processor evaluates a branch command (*e.g.*, during the execution of an encryption/decryption process), it requests access to the branching buffer. The spy process frees the branching buffer for the execution of the branching command. When the branching buffer is free again, the spy process will retake control and thus the execution time of the branching command could be revealed to the attacker.

Suppose that there are n possible branches on a branching point. A better idea would be to execute a sequence of n branches to evict the target branch's entry out of the branch target buffer and measure the overall execution time of each branch. This type of attack is named as Simple Branch Prediction Attack (SBPA) [AKS07]. It is able to extract 508 bits from a 512-bit OpenSSL 0.9.7e RSA key by using ten measurements which proves that the blinding or randomization techniques of the OpenSSL protocol need improvements.

4 Timing attacks on World Wide Web

4.1 Exploiting web caching

Felten and Schneider in [FS00] carried out various types of timing attacks on web based systems. Their target was to determine whether a user has visited a specific web page in the past, and thus to retrieve private information about the user (*e.g.*, an insurance company would be delighted to know whether their web site visitors and possible customers have been visiting web sites which are related to specific categories of health problems). This attack was embedded in a web site or a web service and its most important forms were exploiting web or DNS caching. The implementation of the attack could be JavaScript

dependent or not. Suppose that A is the attacker's web site, T is the target web site and the naive user is denoted by U . Two attack scenarios are presented below:

- *JavaScript dependent - Jd*: A visits T and picks a file that is displayed to the majority of the web page visitors (*i.e.*, a logo image). Then, writes a Java applet and embeds it in his home page. Sometime in the future, U visits A and automatically downloads the applet which measures the time required to access the logo image on U 's machine. The Java applet reports the timing results to A .
- *JavaScript non dependent - Jnd*: When U visits A , A loads a fake file located on itself (t_1), then loads a file (*i.e.*, a logo image) from the target web site and finally loads again a fake file located on itself (t_2). The computation of t_1-t_2 is a good estimation of the file's load time on the target web site.

By collecting a reasonable amount of timing estimations and deploying statistical methods they managed to guess with an accuracy of 98.5% (*Jd*) or 96.7% (*Jnd*) whether the user has previously visited the target web site. Countermeasures to web based attacks would be turning off caching or JavaScript on the browser [FS00].

4.2 Direct and Cross-site timing attacks

Recently, Bortz *et al* presented two timing attacks which exposed private information on web application implementations. The direct timing attack collects timing information directly from a web site by measuring response times for specific operations while the cross-site timing attack uses a 'malicious' page to obtain information about user's activities on another site.

Direct timing attacks could be applied with two different forms:

- A Boolean test: *e.g.*, an attack on a web site's login page by imposing 'fake' usernames and taking measurements of the web site's response times. The web site will respond in less time if the username is invalid.
- Hidden data size estimation test: by measuring the load time of a web page we determine whether private or hidden data exist and estimate an approximation on the number of hidden elements, *e.g.*, the attacker is able to reveal information about the size and number of probable hidden pictures in a web page's photo gallery.

To implement a cross-site timing attack combined JavaScript and HTML code is used. By using an IMG tag that doesn't represent an existing image, an HTTP request to a target web server is processed. As the image does not exist, the *onerror* handler will respond to this request. The time interceded between the HTTP request and the response is measured and used for the timing attack purposes. In order to proceed with the attack the timing of another web-page is essential. This web page should ideally belong on the same domain with the target page and be a non-existing web-page (*e.g.*, a page request that

would trigger the '404 error' message web page) or generally a page with a little amount of data. The cross-site timing attack is able to estimate if a user is logged on a web site and furthermore to determine important private information values (*i.e.*, the number of items in a shopping cart, the number of transactions at a bank site or the number of e-mails in a web-mail account) [BBN07].

5 Conclusions

In this report, an effort to cover the most important, published timing attacks was made. Most of them are attacks that systems should not be afraid of as the defences against them are known and relatively easy to deploy. However, the power of a timing attack lies on the fact that there are no permanent countermeasures against it. From our point of view, even 'constant times' could not be a solution; the attacker will always be able to find something unstable, and thus measurable in a constant-time implementation for the simple, naive but true reason that nothing can be completely constant.

Reading for the first time Kocher's idea and then timing attack on RSA was the most interesting thing throughout this report; his work was the starting point for this research field. Of course, timing attacks got a lot more complicated than their first version. The most attracting thing about timing attacks is that they are based on a very 'smart' idea. An idea able to break into a well designed and extremely 'secure' system by performing 'two clicks': the first starts a stopwatch and the second stops it.

References

- [AKS07] O. Acicmez, Ç.K. Koc, and J.P. Seifert. On The Power of Simple Branch Prediction Analysis. *ACM Symposium on Information, Computer and Communications Security ASIACCS07*, 2007.
- [ASK05] O. Aciiçmez, W. Schindler, and Ç.K. Koç. Improving Brumley and Boneh timing attack on unprotected SSL implementations. *Proceedings of the 12th ACM conference on Computer and communications security*, pages 139–146, 2005.
- [ASK07] O. Acicmez, J.P. Seifert, and Ç.K. Koc. Predicting Secret Keys via Branch Prediction. *Topics in Cryptology-CT-RSA*, 2007.
- [BB05] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.
- [BBN07] A. Bortz, D. Boneh, and P. Nandy. Exposing private information by timing web applications. *Proceedings of the 16th international conference on World Wide Web*, pages 621–628, 2007.

- [Cop97] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.
- [DKL⁺98] J.F. Dhem, F. Koeune, P.A. Leroux, P. Mestre, J.J. Quisquater, and J.L. Willems. A Practical Implementation of the Timing Attack. *Proceedings of CARDIS*, pages 167–182, 1998.
- [FH05] M. Furlong and H. Heys. A timing attack on the CIKS-1 block cipher. *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 231–234, 2005.
- [FS00] E.W. Felten and M.A. Schneider. Timing attacks on Web privacy. *Proceedings of the 7th ACM conference on Computer and communications security*, pages 25–32, 2000.
- [HH98] H. Handschuh and H. Heys. A Timing Attack on RC5. *Proceedings of the Selected Areas in Cryptography*, pages 306–318, 1998.
- [HMOV04] D.R. Hankerson, A.J. Menezes, and S.A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.
- [Koc96] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Advances in Cryptology-CRYPTO*, 96:104–113, 1996.
- [LRWW04] B.N. Levine, M.K. Reiter, C. Wang, and M. Wright. Timing Attacks in Low-Latency Mix Systems. *Proceedings of Financial Cryptography: 8th International Conference (FC 2004): LNCS*, 3110, 2004.
- [Men97] A.J. Menezes. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [Sch00] W. Schindler. A Timing Attack against RSA with the Chinese Remainder Theorem. *CHES 2000*, pages 109–124, 2000.
- [Sha51] C.E. Shannon. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64, 1951.
- [SWT01] D.X. Song, D. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. 2001.
- [TMSK05] Y. Tomoeda, H. Miyake, A. Shimbo, and S. Kawamura. An SPA-Based Extension of Schindler’s Timing Attack against RSA Using CRT. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, pages 147–153, 2005.