On Modelling Real-time and Security properties of Distributed Systems – Extended Abstract

Ricardo Corin, Sandro Etalle, Pieter Hartel, Angelika Mader*

Abstract

We discuss a simplified version of the timing attack to illustrate a connection between security and real-time properties of distributed systems. We suggest several avenues for further research on this and similar connections.

1 Introduction

A functionally correct system must satisfy a range of systemic (i.e. non-functional) requirements to be fit for purpose. Systemic requirements include energy efficiency, low noise emission, low EMF radiation, (real)-timeliness, security, cost effectiveness etc. The main difficulty is that while functional correctness can often be modularised, systemic 'correctness' cannot be modularised. The reason is that systemic properties are not compositional, i.e. the systemic properties of individual components are rarely preserved when the components are integrated because the components tend to interfere with each other in many unexpected ways. Engineers usually over-dimension designs to cater for the worst case scenario, which makes systems more costly than strictly necessary.

We offer insight into the interaction of two systemic properties: security and real-timeliness, where we use the timing attack [12] as the prime example of an undesirable interaction. Mounting a timing attack requires the ability to measure time with predictable accuracy, which is of course exactly what real-time systems are all about. Therefore we claim that the security of a system can be weakened in principle by making the system suitable for real-time applications. It is precisely this kind of interaction that is horribly difficult to predict, and which ultimately determines whether a system is fit for purpose.

2 Timing attacks

Timing attacks can be used to discover a secret key by measuring the time taken by cryptographic operations. Other systemic properties can be exploited in the same way, for instance to mount power attacks [7], and attacks based on fault induction [5] etc.

The basic assumptions of timing analysis are:

1. The run time of a cryptographic operation depends to some extent on the key. With present hardware this is likely to be the case, but note that there are various efficient hardware based proposals to make the timing attack less feasible through 'noise injection' [17]. Software approaches to make the timing attack infeasible are based on the idea that the computations in two branches of a conditional should take the same amount of time ('branch equalisation'). This is costly [2].

2. A sufficiently large number of encryptions can be carried out, during which time the key does not change. A challenge response protocol is ideal for Timing Attacks.

3. Time can be measured with known error. The smaller the error, the fewer time measurements are required.

Different versions of timing attack have been reported to be effective with RSAREF [15], DES [14], and RC5 [13].

Timing attacks usually take place in a controlled environment, where the system under attack (often a smart card) is connected to measurement equipment, particularly to control the errors in the timing measurements. A recent report [6] describes a timing attack on OpenSSL. However, the report acknowledges that mounting this attack over the network is unlikely to be successful because time measurement is too inaccurate. We believe that in a distributed system with real-time properties, timing attacks on the security protocols of such system may become a threat. This is the focus of our paper.

3 Modelling

To predict the kinds of attacks that could be mounted on the security of a distributed system would appear to be difficult in its full generality. However, by singling out specific systemic properties we believe that progress can be made by modelling the cause and effects of the attacks. We adhere to the usual Dolev-Yao model of attackers [11],

*Dept. of Electrical Engineering, Mathematics and Computer Science, Univ. of Twente, Email: {corin, etalle, pieter, mader}@cs.utwente.nl

where we take into account timing information. This requires that we model both the protocol and the encryption scheme.

Timing aspects of encryption schemes are typically analysed using statistical methods and tools such as Matlab, where attacks are modelled as signal detection problems. The signal is in some way correlated with the secret keys. The noise originates from timing inaccuracy, unknown key bits etc [15]. The level of abstraction is generally too low to take protocol aspects into account.

Security protocols are typically analysed using formal methods and tools such as Casper [16], and CoProVe [9], where attacks are modelled by an attacker who learns secrets by inspecting, deleting and repeating messages. The level of abstraction is generally too high to take timing information into account.

We have not been able to find related work that addresses the combined modelling of secrecy and timing.

As an initial step towards combined modelling we study a simple protocol using the timed automata based modelling method and tool Uppaal [4]. To offer insight into the problems that will arise, we describe:

- an abstraction of the timing attack in the 'security protocol verification' style (Section 4), followed by
- an analysis of the attack in the 'timed automata verification' style (Section 5).

4 Abstraction

In the design of cryptographic protocols, it is usually a good idea to adopt an abstract view of cryptographic operations. For example, given a message M representing a bit string, encryption can be represented by a *symbolic* operation denoted $\{M\}_K$, where K is a message representing the cryptographic key. The operator $\{\cdot\}_\cdot$ can be seen as a "black-box" operation, that hides all the details of the particular encryption algorithm. Typically, the only way to obtain M again from $\{M\}_K$ is by possessing the correct key K. To illustrate this point consider Figure 1. Cryptographic protocols *use* an encryption scheme to achieve some security goals. Analysing the protocol in an idealised setting, while assuming "black-box" cryptography, we can avoid considering the dashed box and arrow of Figure 1. This means that we can ignore any vulnerability of the encryption scheme and concentrate on the vulnerabilities of the protocol. However, in many cases it is unrealistic to ignore the vulnerabilities of the encryption scheme, thus we should be turning the "black-box" approach into a gray-box approach or even a white-box approach [8].

We focus on the case where the encryption scheme is vulnerable to timing attacks. In other words, we are interested in studying the security of protocols which are implemented using an encryption scheme that is subject

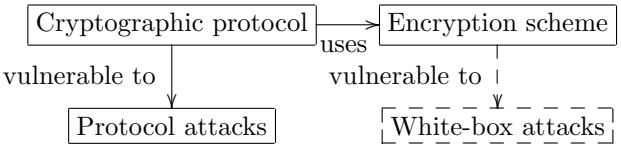

Figure 1: Protocols achieve goals by using cryptography. Ideal black-box cryptography avoids cryptographic vulnerabilities, i.e. ignore the dashed box and arrow.

to timing attacks. Consider, as an example, the following protocol:

1. $A \to B: \quad \{\{V\}_K, action\}_{K_B}$
2. $B \to A: \quad \begin{cases} \{V\}_{K_A} & \text{if } action = \text{"decrypt"} \\ \{ok\}_{K_A} & \text{if } action = \text{"store"} \end{cases}$

First, A prepares a message consisting of a value $\{V\}_K$ together with a value *action* chosen from {"decrypt", "store"}. Then, A encrypts this message with B's public key K_B and sends it to B in message 1. When B receives this message, it extracts and inspects *action*. If the *action* is "decrypt", B decrypts $\{V\}_K$ (we assume that B knows K) and obtains V. Then, B encrypts V with A's public key and sends it to A. If the *action* is "store", B simply stores $\{V\}_K$ and replies to A the value "ok" encrypted with A's public key.

Suppose that, for privacy's sake, we would like to keep the value of *action* secret. If we assume that encryption is non-deterministic, an attacker would not be able to distinguish between B's response, that is between $\{V\}_{K_A}$ and $\{ok\}_{K_A}$. Thus, in an idealised setting the secrecy of *action* would be preserved.

Now suppose that encryption is subject to timing attacks. More precisely, we suppose that an attacker can measure the time, say t, between receiving message 1 by B and sending message 2, also by B. Thus, an attacker would notice that on average, t is larger when *action* is "decrypt" than when *action* is "store". This is so since when *action* is "decrypt", B needs to perform an extra decryption of $\{V\}_K$, which is not carried out when *action* is "store". Therefore, the ability of the attacker to measure the time allows the attacker to mount a successful attack and obtain the value of *action*, intended to be secret. With the present tools and methodologies for security protocol verification we cannot model this attack because the notion of time is absent. Therefore we explore the possibilities of using Uppaal, which does offer timing analysis capabilities.

5 Analysis

We modelled the system with timed automata [3] using the tool Uppaal. Timed automata extend finite state au-

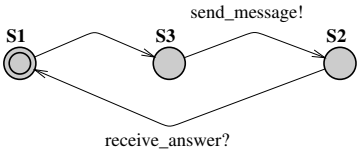

Figure 2: Timed automaton for a Sender

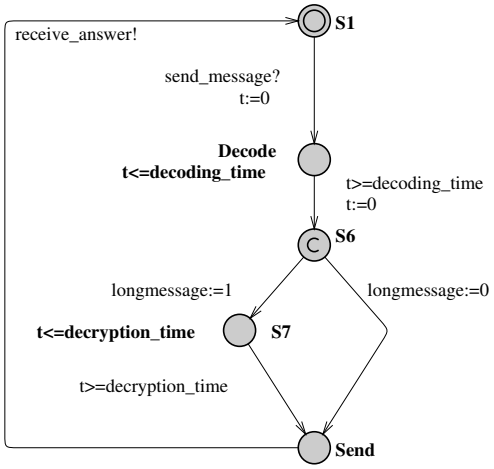

Figure 3: Timed automaton for a Receiver

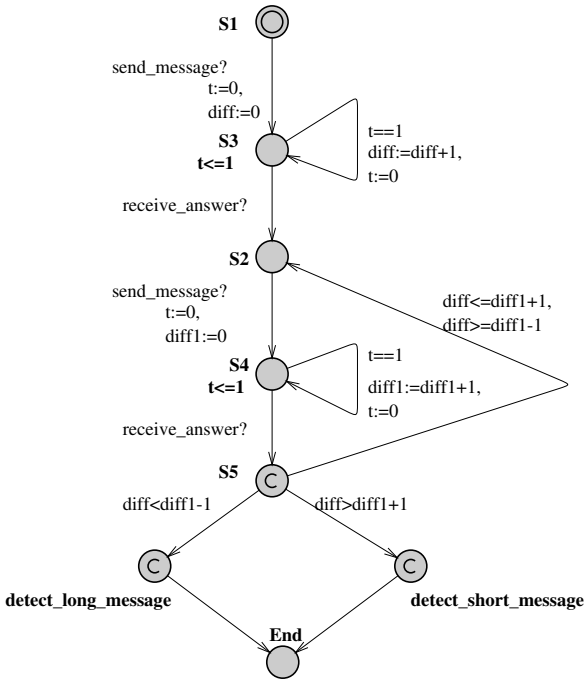

Figure 4: Timed automaton for an Attacker

tomata by the usage of clocks that can be reset and used in invariants on locations and guards on transitions.

The system consists of three processes, the Sender, the Receiver and an Attacker, each of them modelled as a timed automaton (see figures 2, 3, and 4).

The sender can wait an arbitrary amount of time, and then decide to go to a location where it immediately continues by synchronisation via the channel *send_message* with the receiver, which models the sending of a message.

The receiver then starts to decode the message. Decoding models the combined effect of the public key decryption and the choice of the conditional. After having finished decoding, the receiver decides which type of message it has received. Because we abstract from data-transmission in this model, the decision is nondeterministic. Depending on the decision, the local variable *longmessage* is assigned 1 and a decryption has to take place (corresponding to *action* = "decrypt"), before the location *Send* is reached. In the other case the local variable *longmessage* is assigned 0 and there is a direct transition to the location *Send* (corresponding to *action* = "store"). This location is left immediately synchronising on the urgent channel *receive_answer*, thus modelling that an answer is sent.

The attacker is able to synchronise on the actions *send_message* and *receive_answer*, modelled using broadcast channels. The attacker counts time units (variable *diff*) between a *send_message* and a *receive_answer*. The attacker also counts time units between a new pair of *send_message* and a *receive_answer* (variable *diff1*). The attacker then compares the two durations *diff* and *diff1*. If no difference is detected, the attacker goes back to observe a new message exchange. If the attacker detects a significant difference, the smaller is identified as *detect_short_message* and the longer as *detect_long_message*.

By model checking we have verified that if the attacker identifies the last message as a long message, the local decision of the receiver is indeed *longmessage=1*, and similarly for short messages.

When changing the receiver automaton, such that it can wait an arbitrary time before sending the answer (See Figure 5), the type of message cannot be identified any more correctly. Model checking this changed system shows that we can reach a state where *longmessage=1*, and the Attacker is in state *detect_short_message*.

The Uppaal model illustrates that a timing attack can be modelled under ideal circumstances, where the probability of making the correct inference is either 0 or 1. An obvious extension would be to consider non-trivial probabilities, thus approaching a more realistic version of the Timing Attack. This is supported by e.g. the MoDeST approach, which is an extension of timed automata with stochastic concepts [10].

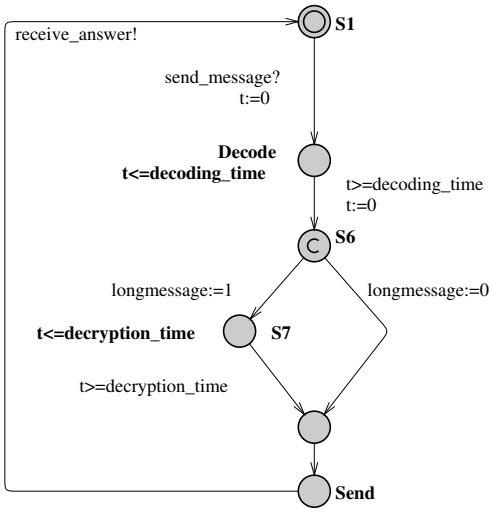

Figure 5: Timed automaton for a Receiver with an extra delay before sending

6 Conclusions and Future work

The notional separation of security protocols from encryption schemes has made it possible to make significant progress in security modelling and analysis in each separate domain. However, timing affects the protocols and protocols affect the timing, so that eventually security protocols and encryption schemes will have to be studied simultaneously. The development of the theory underpinning this has already been initiated [1], but work on modelling methods and tools is yet to start. We believe this work to be essential for the development of sound engineering methods for security in distributed systems.

We have studied a simplified version of the timing attack; we intend to take the statistics of the attack into account in future work.

One possible avenue of research is to use existing tools for the verification of both encryption schemes and security protocols. This would have the advantage that we can leverage the power of the tools, which, at the cost of many person years, have been engineered to be able to cope with sizeable models. It may also be possible to make a connection between different tools (for example Uppaal, MoDeST and CoProVe) so that results from one tool can be fed into the other and vice versa.

Another, equally important avenue of research is to develop modelling methods for protocols that are a little less abstract, and modelling methods for security schemes that are a little more abstract. Each represents a bridgehead, which, we hope, will eventually support a strong bridge between the two domains.

Countermeasures for timing attacks must be modelled so that we can study how effective the proposed measures are. The two measures that are currently in use (i.e. noise injection and branch equalisation) appear to be fundamentally different in the sense that noise injection weakens the power of the timing attack but it does not defeat it, whereas branch equalisation does defeat the attack but at significant cost.

Finally, the problems that we have discussed relating to timing will appear also in relation to the other systemic parameters, thus requiring the study of interaction between a multitude of systemic parameters. One might hope that eventually a general theory and associated methods and tools might emerge that will support the security engineer.

Acknowledgement

We thank Jeroen Doumen for his comments on the paper.

References

[1] M. Abadi and Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002. http://www.springerlink.com/link.asp?id=q5jt65hx0u90rkae. 4

[2] J. Agat. Transforming out timing leaks. In *27th Principles of programming languages (POPL)*, pages 40–53, Boston, Massachusetts, Jan 2000. ACM Press, New York. 1

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical computer science (TCS)*, 126:183–235, 1994. 2

[4] T. Amnell, G. Behrmann, J. Bengtsson, P. R. D'Argenio, A. David, A. Fehnker, T. Hune, B. Jeannet, K. G. Larsen, M. O. Möller, P. Pettersson, C. Weise, and W. Yi. UPPAAL - now, next, and future. In F. Cassez, C. Jard, B. Rozoy, and M. D. Ryand, editors, *4th Summer School on Modeling and Verification of Parallel Processes (MOVEP)*, volume LNCS 2067, pages 99–124, Nantes, France, Jun 2000. Springer-Verlag, Berlin. http://www.springerlink.com/link.asp?id=8km1eyyvfqfm740f. 2

[5] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. Kaliski Jr., editor, *17th Advances in Cryptology (CRYPTO)*, volume LNCS 1294, pages 513–525, Santa Barbara, California, Aug 1997. Springer-Verlag, Berlin. 1

[6] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a SSL/TLS channel. In D. Boneh, editor, *Advances in Cryptology (CRYPTO)*, page to appear, Santa Barbara, California, Aug 2003. Springer-Verlag, Berlin. 1

[7] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi. A cautionary note regarding evaluation of AES candidates on Smart-Cards. In *2nd Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, 1999. http://csrc.nist.gov/CryptoToolkit/aes/round1/conf2/papers/chari.pdf. 1

[8] S. Chow, P. Eisen, H. Johnson, and P. C. van Oorschot. White-Box cryptography and an AES implementation. In *9th Int. Workshop on Selected Areas in Cryptography (SAC)*, volume LNCS 2595, pages 250–270, St. John's, Newfoundland, Canada, Aug 2002. Springer-Verlag, Berlin. 2

[9] R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In M. V. Hermenegildo and G. Puebla, editors, *9th Int. Static Analysis Symp. (SAS)*, volume LNCS 2477, pages 326–341, Madrid, Spain, Sep 2002. Springer-Verlag, Berlin. http://www.ub.utwente.nl/webdocs/ctit/1/00000096.pdf. 2

[10] P. R. D'Argenio, H. Hermanns, J.-P. Katoen, and J. Klaren. MODEST: A modelling language for stochastic timed systems. In L. de Alfaro and S. Gilmore, editors, *Joint Int. Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV)*, volume LNCS 2165, pages 87–104, Aachen, Germany, Sep 2001. Springer-Verlag, Berlin. http://www.springerlink.com/link.asp?id=ka9lrbfnj7g6a9dd. 3

[11] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. 1

[12] E. English and S. Hamilton. Network security under siege: the timing attack. *IEEE Computer*, 29(3):95–97, Mar 1996. http://www.computer.org/computer/co1996/r3095abs.htm. 1

[13] H. Handschuh and H. M. Heys. A timing attack on RC5. In *Selected Areas in Cryptography (SAC)*, volume LNCS 1556, pages 306–318, Kingston, Canada, Aug 1998. Springer-Verlag, Berlin. 1

[14] A. Hevia and M. Kiwi. Strength of two data encryption standard implementations under timing attacks. In C. L. Lucchesi and A. V. Moura, editors, *3rd Latin American Symp. on Theoretical Informatics (LATIN)*, pages 192–205, Campinas, Brazil, Apr 1998. Springer-Verlag, Berlin. 1

[15] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *16th Advances in Cryptology (CRYPTO)*, volume LNCS 1109, pages 104–113, Santa Barbara, California, Aug 1996. Springer-Verlag, Berlin. http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf. 1, 2

[16] G. Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop (CSFW)*, pages 18–30, Rockport, Massachusetts, Jun 1997. IEEE Computer Society Press, Los Alamitos, California. http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Casper/. 2

[17] D. May, H. Muller, and N. P. Smart. Non-Deterministic processors. In V. Varadharajan and Y. Mu, editors, *6th Australasian Conf. (ACISP) – Information Security and Privacy*, volume LNCS 2119, pages 115–129, Sydney, Australia, Jul 2001. Springer-Verlag, Berlin. 1