The Safety versus Security Dilemma

Hanne Riis Nielson Flemming Nielson Kebin Zeng

DTU Compute, Technical University of Denmark, Denmark

{hrni,fnie,keze}@dtu.dk

June 20, 2013

Abstract

Often systems need to be both safe (so that no harm comes from using the systems) and secure (so that no one can damage the operation of the system). Sometimes these demands conflict with one another, because safety dictates quick and timely responses, while security necessitates time consuming cryptographic operations. We show how to mediate this dilemma by introducing redundancy into the system thereby showing how methods and techniques from the safety culture can be used to achieve the coexistence of safety and security.

1 Introduction

Safety and security are both desirable goals of systems. *Safety* [4] concerns making sure that no harm can arise from using the system; it often requires that the system responds within short time bounds. As an example, the airbag in a car needs to react quickly to signals from the collision sensors in order to achieve its purpose. *Security* [3] has many facets including confidentiality, making sure that nobody can learn data not intended for them, and integrity, making sure that only properly authorized agents can influence the data upon which we need to act. As an example, an automatic brake function for slowing the car when a tyre explodes should make sure to react to the sensors of the car rather than the sensors on neighboring cars.

Sometimes safety and security seem to counteract each other. As we illustrated above, safety often requires that the system responds quickly whereas security can only be achieved by using cryptography that slows down the signaling from sensors. This sometimes calls for using quicker and more expensive hardware or

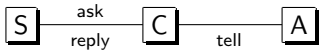

Figure 1: A sensor S, a control unit C and an actuator A together with three channels ask, reply and tell used for communication.

even for redesigning the entire system. In this paper we show that *redundancy* sometimes suffices for solving the dilemma.

Section 2 introduces a very simple system that suffices for our discussion; it consists of a sensor, a control unit and an actuator. To model the system we make use an extension of a fragment of our Quality Calculus [9, 10, 12]. Next in Section 3 we describe the expectations to safety as expressed in standards such as [4] and show how to choose the system parameters to achieve the desired level of safety. In Section 4 we briefly survey methods for achieving security and choose the particular scheme we will be advocating; as an unfortunate consequence our system now fails in achieving the desired level of safety and in the following sections we discuss ways of modifying the system to rectify this. In Section 5 we show how one of the classical techniques from the safety community, namely that of *redundancy*, can be used to regain the desired level of safety. While this involve introducing more sensors into the system, another possibility is to increase the power of the sensor either in its ability to perform the required measurements or in its compute power and this is discussed in Section 6. Finally, in Section 7 we consider the possibility of strengthening the cryptography by using longer keys. We conclude in Section 8.

2 The System

We shall consider a simple system consisting of a sensor S, a control unit C and an actuator A as illustrated on Figure 1. The control unit will poll the sensor for a certain measurement and then wait for a reply for a fixed number of time units, in our example it will be $200\mu s$. The sensor is ready to deliver its measurements at any time but will do so with a small delay so we shall assume that on average it will take $6.2\mu s$ before the measurement is delivered. If the control unit has received the measurement before the $200\mu s$ has passed it will process it and send a message to the actuator before recursing. In the event that the measurement is not received within the $200\mu s$, the control unit still have to take appropriate actions and it will therefore send an alert message to the actuator before recursing. Finally, we shall assume that the actuator is always ready to receive the message.

We shall focus on the operations of the control unit and model it in a version of the Quality Calculus [9, 10, 12]. This is a calculus in the π-calculus [8] family that allows us to express not only what should happen when communications

are successful but also what should happen when they are unsuccessful. In our example this will be useful in the case where the sensor fails to deliver the measurements within the required time limit. We shall use the stochastic version of the Quality Calculus [13] that integrates stochastic delays (and real-time waiting times) with the actions unlike for example the IMC process algebra [2]. The control unit is a recursive process that can be specified as follows:

$$\begin{array}{lcl} \mathsf{C} & \triangleq & \mathsf{ask!}p.\ \&_{\mathsf{true}}^{[200\mu s,200\mu s]}(\mathsf{reply?}x). \\ & & \mathsf{case}\ x\ \mathsf{of}\ \mathsf{some}(y) : \mathsf{tell!}(\mathsf{f}\, y).\ \mathsf{C} \\ & & \mathsf{else}\ \mathsf{tell!alert}.\ \mathsf{C} \end{array}$$

The first action is $\mathsf{ask!}p$ and it will send a message on the channel ask shared with the sensor and ask for the measurement of the parameter p. The control unit will then enter the real-time wait expressed by the binding construct $\&_{\mathsf{true}}^{[200\mu s,200\mu s]}(\mathsf{reply?}x)$; it expresses that the process wants to input a value (to be bound to the variable x) on the channel reply shared with the sensor and that it will wait for exactly $200\mu s$. This construct is an instance of a general binding construct of the form

$$\&_q^{[t_1,t_2]}(c_1?x_1, \cdots, c_n?x_n)$$

stating that we are waiting for a number of inputs $c_1?x_1, \cdots, c_n?x_n$ on channels $c_1, \cdots, c_n$. The construct expresses that will be waiting for at least t_1 time units and at most t_2 time units and within this time interval we will proceed if the inputs received satisfy the predicate q – thus we will never proceed before t_1 time units have passed and, at the very latest we will proceed after t_2 time units independently of whether q holds or not. As an example, the predicate q may be $\forall$ meaning that we will only proceed before t_2 time units have passed if all inputs have been received. Another example is $\exists$ meaning that we proceed as soon as one of the inputs is received but only if at least t_1 time units have passed. In the specification of the control unit above, we are only waiting for one input and the time interval is $[200\mu s, 200\mu s]$ meaning that we are waiting for exactly $200\mu s$; the predicate q is true meaning that we will proceed independently of whether some input has been received or not.

The binding construct of the specification above will result in binding the variable x to a value of the form $\mathsf{some}(c)$ if the input on reply was successful within the time bound and it will bind x to the value none otherwise. The case construct of the second line of the specification then determines which of the two situations apply. In the first case the variable y will be bound to the value received (denoted by c above) and it will be processed (using the function f) before being output on the channel tell shared with the actuator and then the control unit will recurse. Alternatively the communication with the sensor was not successful and the value alert is sent to the actuator before recursing.

Safety Integrity Level	Probability of Failure per Hour
1	10^{-5} – 10^{-6}
2	10^{-6} – 10^{-7}
3	10^{-7} – 10^{-8}
4	10^{-8} – 10^{-9}

Table 1: Safety Integrity Levels and the Probability of Failure per Hour.

3 The Safety Analysis

We shall now assume that the system has to be certified according to one of the safety standards, in particular we shall be interested in determining its *Safety Integrity Level* [4]. Such analyses are based on a probabilistic analysis of the device and since our system is supposed to operate continuously we shall be interested in the *Probability of Failure per Hour*. Depending on this probability the system may be certified with one of four Safety Integrity Levels as shown in Table 1.

As explain in the previous section, the control unit of our system will continuously cycle through the following behaviour. At the beginning of the cycle, the sensor is queried and on average it takes it $6.2\mu s$ to produce a result. It is considered a hazard if a value is not received from the sensor so in the interest of safety, the control unit waits a full $200\mu s$ before actually reading the result from the sensor. Then it will output the result to the actuator before starting all over again.

We shall make the assumption that the delays on part of the sensor in providing the answer is governed by a Markov chain with just one transition, that is by an *exponential distribution*. Taking seconds as our unit of time this gives rise to a parameter λ that is the reciprocal of $6.2 \cdot 10^{-6}s$ which gives $\lambda = 1.613 \cdot 10^{5}s^{-1}$. If more complex distributions are needed we would model them as continuous phase type distributions or Continuous Time Markov Chains and rely on stochastic model checking [5, 6]; however in the following we shall stick to the exponential distribution.

For each cycle we can therefore compute the probability p_c that the sensor signal is *not* available when required. It is given by the formula

$$p_c = e^{-\lambda \cdot t}$$

where $t = 200\mu s$ is the waiting time. In our case this works out to $p_c = 9.8 \cdot 10^{-15}$.

In order to compute the Safety Integrity Level we shall be interested in the probability that at least one error takes place during one hour; it is given by the formula

$$p_h = 1 - (1 - p_c)^n$$

where n is the number of cycles in an hour, that is $n = 3600/(200 \cdot 10^{-6})$. For our example this works out to $p_h = 1.8 \cdot 10^{-7}$; this probability suffices for achieving the Safety Integrity Level 2.

Higher levels can be achieved by replacing the sensor with a more responsive one. As an example, if we replace our current sensor (requiring $6.2\mu s$ on average to return a measurement) with one that only requires $6.0\mu s$ on average, we will obtain the Safety Integrity Level 3.

4 The Cost of Cryptography

We shall now assume that our system will be deployed in a setting where security is an issue. Safety is still an issue so the hard real-time bounds are unchanged and the system still has to be certified according to one of the safety standards so as above we are interested in determining the Safety Integrity Level.

To protect the communication between the sensor and the control unit we shall use cryptography. This means that the sensor will encrypt the message before sending it to the central unit that then will have to decrypt the message before it can be further processed. In order to comply with the overall safety requirements of the system, the hard timing constraints still have to be fulfilled meaning that the hard real-time bound of $200\mu s$ is unchanged. Basically this means that encryption as well as decryption have to happen within this very time interval as illustrated on Figure 2.

For cryptography we shall adopt AES – the Advanced Encryption Standard [1]. It is a symmetric key algorithm meaning that it uses the same key for encryption and decryption of data. The encryption and decryption algorithms proceed in a number of rounds determined by the key length; each round consists of several steps performing various operations influenced by so-called round keys that are extracted from the main key. If the main key is a 128 bit key then a total of 10 rounds are performed, if the key has length 192 bits we need 12 rounds whereas

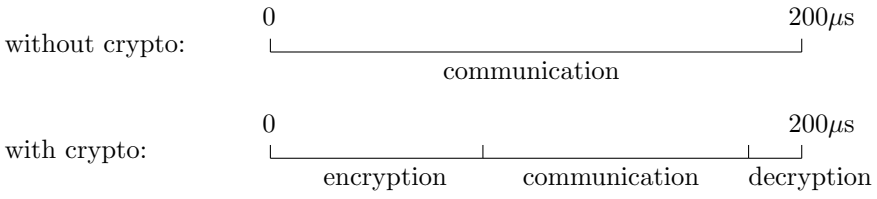

Figure 2: The cryptographic operations must be carried out within the fixed time interval.

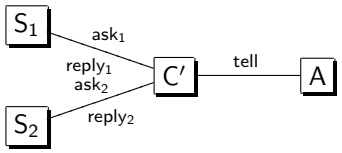

Figure 3: System with Dual Modular Redundancy.

14 rounds are required if the key length is 256 bits.

The operations performed in each of the rounds consist of a number of matrix operations that are very similar so in the following we shall assume that the individual rounds take a fixed amount of time. The time required will depend on the processing power of the device and here it is reasonable to assume that the sensor node and the control unit have different computational power. So in our computations below we shall assume that the sensor requires $8\mu s$ to complete a round whereas the central unit only requires $2\mu s$ to perform the similar task. With a 128 bit key the sensor will have to perform 10 such rounds thereby spending $80\mu s$ on encryption. Similarly the control unit will have to perform 10 rounds to perform the decryption and for this it will use $20\mu s$. As the overall real-time requirements are unchanged, this means that the required $100\mu s$ for cryptographic operations must be taken from the $200\mu s$ thereby leaving just $100\mu s$ for the communication.

In the case of 128 bit keys and 10 rounds the resulting control unit can be modelled as follows:

$$\begin{array}{lcl} \mathsf{C} & \triangleq & \mathsf{ask!}p.\ \&_{\mathsf{true}}^{[100\mu s,100\mu s]}(\mathsf{reply?}x). \\ & & \mathsf{case}\ x\ \mathsf{of}\ \mathsf{some}(y) : \mathsf{decrypt}\ y\ \mathsf{into}\ z :\ \mathsf{tell!}(\mathsf{f}\ z).\ \mathsf{C} \\ & & \mathsf{else}\ \mathsf{tell!alert}.\ \mathsf{C} \end{array}$$

It is not surprising that adding cryptography has safety implications. As before the formula $p_c = e^{-\lambda \cdot t}$ describes the risk that the message does not arrive within t time units. The sensor is unchanged so it still produces a measurement every $6.2\mu s$ on average so the parameter λ is unchanged but the time period t available for the communication is now just $100\mu s$. Therefore the risk of the failure of a single measurement will raise to $p_c = 9.9 \cdot 10^{-8}$.

We can now compute the probability that at least one error takes place during one hour; as before it is given by the formula $p_h = 1 - (1 - p_c)^n$ where n is the number of cycles per hour; with the new value of p_c it works out to $p_h = 0.83$ – which clearly is unacceptable with respect to the safety standards.

5 Redundancy

One of the classical approaches to reducing the failure rate of systems is to incorporate some form of redundancy [7]. We shall now show how to do this in the present setting where we have incorporated cryptographic operations in our system.

The idea is to extend the system with an additional sensor performing the same measurements as the original sensor and then to reprogram the control unit so that it communicates with both sensors. This is illustrated on Figure 3 and the actual code for the control unit C in the absence of cryptography is as follows:

$$\begin{array}{lcl} \mathsf{C} & \triangleq & \mathsf{ask_1}!p.\ \mathsf{ask_2}!p.\ \&_{\mathsf{true}}^{[200\mu s,200\mu s]}(\mathsf{reply_1}?x_1, \mathsf{reply_2}?x_2). \\ & & \mathsf{case}\ x_1\ \mathsf{of}\ \mathsf{some}(y_1) : \mathsf{tell}!(\mathsf{f}\ y_1).\ \mathsf{C} \\ & & \mathsf{else\ case}\ x_2\ \mathsf{of}\ \mathsf{some}(y_2) : \mathsf{tell}!(\mathsf{f}\ y_2).\ \mathsf{C} \\ & & \quad \mathsf{else\ tell}!\mathsf{alert}.\ \mathsf{C} \end{array}$$

The request for the parameter p is sent to both sensors and the binding construct $\&_{\mathsf{true}}^{[200\mu s,200\mu s]}(\mathsf{reply_1}?x_1, \mathsf{reply_2}?x_2)$ is now expressing that we wait for exactly $200\mu s$ time units and at that time we will inspect whatever inputs have been achieved on the two channels $\mathsf{reply_1}$ and $\mathsf{reply_2}$. The two nested case statements give priority to the input received from sensor $\mathsf{S_1}$ and only if none of the two sensors have succeeded in delivering a measurement we will send the alert to the actuator.

In the case of cryptography with 128 bit keys and 10 rounds, the control unit C may be defined as follows:

$$\begin{array}{lcl} \mathsf{C} & \triangleq & \mathsf{ask_1}!p.\ \mathsf{ask_2}!p.\ \&_{\mathsf{true}}^{[100\mu s,100\mu s]}(\mathsf{reply_1}?x_1, \mathsf{reply_2}?x_2). \\ & & \mathsf{case}\ x_1\ \mathsf{of}\ \mathsf{some}(y_1) : \mathsf{decrypt}\ y_1\ \mathsf{into}\ z_1 :\ \mathsf{tell}!(\mathsf{f}\ z_1).\ \mathsf{C} \\ & & \mathsf{else\ case}\ x_2\ \mathsf{of}\ \mathsf{some}(y_2) : \mathsf{decrypt}\ y_2\ \mathsf{into}\ z_2 :\ \mathsf{tell}!(\mathsf{f}\ z_2).\ \mathsf{C} \\ & & \quad \mathsf{else\ tell}!\mathsf{alert}.\ \mathsf{C} \end{array}$$

As in the previous section the time interval is just $100\mu s$.

Following Section 4 the formula $p_c = e^{-\lambda \cdot t}$ describes the risk that the measurement from a single sensor does not arrive within t time units. The risk that none of the two sensors are successful within t time units will therefore be p_c^2. In the case where $t = 100\mu s$ this amounts to $9.8 \cdot 10^{-15}$.

As before we can compute the probability that at least one error takes place during one hour; it is now given by the formula $p_h = 1 - (1 - p_c^2)^n$ that works out to $p_h = 1.8 \cdot 10^{-7}$. This probability suffices for achieving the Safety Integrity Level 2 – the same level that we achieved in Section 3 with the same sensor.

6 More powerful sensors

In the previous section we showed how redundancy could be used to improve the system when cryptography was added. In this section we shall now investigate to what extent similar effects can be obtained by replacing the (single) sensor with one with a better performance. We shall consider two possibilities: One is to improve on the measuring capabilities of the sensor so that the average time for obtaining a measurement becomes lower. Another possibility is to improve the computational power of the sensor so that it can perform the cryptographic operations faster.

Let us first consider improving the measuring abilities of the sensor. We shall repeat the calculations of Section 4 using a 128 bit key but with a sensor that performs the measurements faster. The first column of the table below shows the improvement in the average time needed for the sensor to produce a result, the second column shows the resulting probability for the message not being present when required by the control unit (within the $100\mu s$ time bound) and the last column shows the probability of missing a measurement within one hour.

improvement	p_c	p_h
$6.2\mu s(100\%)$	$9.9 \cdot 10^{-8}$	0.83
$5.58\mu s(90\%)$	$1.6 \cdot 10^{-8}$	0.26
$4.96\mu s(80\%)$	$1.8 \cdot 10^{-9}$	0.031
$4.34\mu s(70\%)$	$9.8 \cdot 10^{-11}$	0.0018
$3.72\mu s(60\%)$	$2.1 \cdot 10^{-12}$	$3.8 \cdot 10^{-5}$
$3.10\mu s(50\%)$	$9.8 \cdot 10^{-15}$	$1.8 \cdot 10^{-7}$

The table shows that a substantial improvement is necessary as only the last entry in the table will warrant a Safety Integrity Level, in this case level 2.

An alternative is to improve the computational capabilities of the sensor, that is, the time it requires in order to perform the cryptographic operations. So far we have assumed that it takes $8\mu s$ to perform each of the 10 rounds of computation required for a 128 bit key. The table below shows the results obtained by reducing the amount of time needed to perform each round of the encryption. The first column shows the time required for a single round and the next two columns list the resulting probability for a single message to be lost and the probability for missing a measurement within an hour.

improvement	p_c	p_h
$8\mu s$	$9.9 \cdot 10^{-8}$	0.83
$6\mu s$	$3.9 \cdot 10^{-9}$	0.068
$4\mu s$	$1.6 \cdot 10^{-10}$	0.0028
$2\mu s$	$6.2 \cdot 10^{-12}$	0.00011

Thus we see that even when the sensor is as fast as the control unit it is not possible to obtain any of the Safety Integrity Levels.

However, if we combine the two improvements we obtain more interesting results outlining a design space. The columns in the following table record the improvements of measuring capabilities whereas the rows record the improved computational capabilities of the sensor. For each combination we then list the probability of failure per hour:

	$6.2\mu s$	$5.58\mu s$	$4.96\mu s$	$4.34\mu s$
$8\mu s$	0.83	0.26	0.031	0.0018
$6\mu s$	$8.2 \cdot 10^{-3}$	$2.0 \cdot 10^{-2}$	$5.6 \cdot 10^{-4}$	$1.8 \cdot 10^{-5}$
$4\mu s$	$2.8 \cdot 10^{-3}$	$2.3 \cdot 10^{-4}$	$9.9 \cdot 10^{-6}$	$1.8 \cdot 10^{-7}$
$2\mu s$	$1.1 \cdot 10^{-4}$	$6.3 \cdot 10^{-6}$	$1.8 \cdot 10^{-7}$	$2.0 \cdot 10^{-9}$

Thus we see that the combination of $4.96\mu s$ and $4\mu s$ allow us to obtain the Safety Integrity Level 1 and so does the combination of $5.58\mu s$ and $2\mu s$. The combination of $4.34\mu s$ and $4\mu s$ as well as that of $4.96\mu s$ and $2\mu s$ allow us to obtain a Safety Integrity Level of 2.

7 Strengthening the cryptography

What happens if we strengthen the cryptography by using a 256 bit key? The first observation is that we need 14 rounds in the encryption and decryption algorithms rather than just 10. This means that $140\mu s$ will be used for cryptographic operations (rather than just $100\mu s$) and thus only $60\mu s$ are left for delays in communication. Without any redundancy the risk of failure within one hour is 1 and even with Dual Modular Redundancy as in Section 5 we cannot achieve any Safety Integrity Levels as the risk of failure within one hour amounts to 0.068.

Therefore we might consider adding yet another sensor and thereby go for Triple Modular Redundancy; this is captured by the following modification of the code considered earlier:

$$
\begin{array}{lll}
\mathsf{C} & \triangleq & \mathsf{ask}_1!p.\ \mathsf{ask}_2!p.\ \mathsf{ask}_3!p.\ \&_{\mathsf{true}}^{[60\mu s, 60\mu s]}(\mathsf{reply}_1?x_1, \mathsf{reply}_2?x_2, \mathsf{reply}_3?x_3). \\
& & \mathsf{case}\ x_1\ \mathsf{of}\ \mathsf{some}(y_1) : \mathsf{decrypt}\ y_1\ \mathsf{into}\ z_1 :\ \mathsf{tell}!(\mathsf{f}\ z_1).\ \mathsf{C} \\
& & \mathsf{else\ case}\ x_2\ \mathsf{of}\ \mathsf{some}(y_2) : \mathsf{decrypt}\ y_2\ \mathsf{into}\ z_2 :\ \mathsf{tell}!(\mathsf{f}\ z_2).\ \mathsf{C} \\
& & \quad \mathsf{else\ case}\ x_3\ \mathsf{of}\ \mathsf{some}(y_3) : \mathsf{decrypt}\ y_3\ \mathsf{into}\ z_3 :\ \mathsf{tell}!(\mathsf{f}\ z_3).\ \mathsf{C} \\
& & \quad\quad \mathsf{else\ tell}!\mathsf{alert}.\ \mathsf{C}
\end{array}
$$

The risk of having no measurement from any of the sensors now amounts to $p_c = 2.5 \cdot 10^{-13}$ and the failure rate per hour now becomes $p_h = 4.4 \cdot 10^{-6}$ meaning that we can achieve the Safety Integrity Level 1.

8 Conclusion

The ideas presented in this paper were largely motivated by the considerations of the SESAMO project [11] which studies the interplay between safety and security in the construction of embedded systems. Both safety and security are desirable or even necessary properties of such systems but sometimes they offer conflicting demands on the embedded system. In the present paper our focus was on timeliness of response – a property that is often required for safety and that may be jeopardised by the time taken to follow cryptographic communication protocols.

Using the Quality Calculus [9, 10, 12] as a means to explaining our design we showed how to analyse a tiny system to achieve security guarantees in compliance with the Safety Instrumented Standard [4]. We next showed how to model the "strain" on the system presented by needing to use part of the time for cryptographic protection. The resulting safety analysis showed that we could not maintain the required security guarantee. In our case this could be rectified by redundancy – simply having two sensors instead of one. This study was scalable in the sense that it could be adapted to different instantiations of the cryptographic scheme presenting different demands on the amount of computation time needed.

This suggests that some of the methods and techniques for fault tolerance employed in embedded system can also be used for overcoming the challenges posed by the need to simultaneously offering safety and security in embedded systems.

Acknowledgement. This work was supported by the MT-LAB research centre (*www.MT-LAB.dk*) funded by the VILLUM foundation and studying the modelling of IT systems; the research question addressed was largely motivated by the European Artemis project SESAMO (*www.SESAMO-PROJECT.eu*).

References

[1] *Advanced Encryption Standard*, csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[2] Ed Brinksma and Holger Hermanns. Process Algebra and Markov Chains. In *Euro Summer School on Trends in Computer Science*, volume 2090 of *Lecture Notes in Computer Science*, pages 183–231. Springer, 2001.

[3] *Common Criteria for Information Technology Security Evaluation — IEC 15408*, www.commoncriteriaportal.org.

[4] *Functional Safety of electrical, electronic and programmable electronic safety related systems — IEC 61508*, www.61508.org, www.iec.ch/functionalsafety/.

[5] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In Marco Bernardo and Jane Hillston, editors, *Formal Methods for Performance Evaluation*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, 2007.

[6] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[7] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200–209, 1962.

[8] Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.

[9] Hanne Riis Nielson and Flemming Nielson. Probabilistic analysis of the quality calculus. In Dirk Beyer and Michele Boreale, editors, *Formal Techniques for Distributed Systems (FMOODS/FORTE)*, volume 7892 of *Lecture Notes in Computer Science*, pages 258–272. Springer, 2013.

[10] Hanne Riis Nielson, Flemming Nielson, and Roberto Vigo. A calculus for quality. In Corina S. Pasareanu and Gwen Salaün, editors, *Formal Aspects of Component Software (FACS)*, volume 7684 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2013.

[11] *Security and Safety Modelling — an EU Artemis project*, www.sesamo-project.eu.

[12] Roberto Vigo, Flemming Nielson, and Hanne Riis Nielson. Broadcast, denial-of-service, and secure communication. In Einar Broch Johnsen and Luigia Petre, editors, *Integrated Formal Methods (IFM)*, volume 7940 of *Lecture Notes in Computer Science*, pages 412–427. Springer, 2013.

[13] Kebin Zeng, Flemming Nielson, and Hanne Riis Nielson. *The Stochastic Quality Calculus (Work in Progress)*, 2013.