Quantifying Risks to Data Assets Using Formal Metrics in Embedded System Design

Maria Vasilevskaya and Simin Nadjm-Tehrani

Dept. of Computer and Information Science
Linköping University, Sweden
maria.vasilevskaya@liu.se, simin.nadjm-tehrani@liu.se

Abstract. This paper addresses quantifying security risks associated with data assets within design models of embedded systems. Attack and system behaviours are modelled as time-dependent stochastic processes. The presence of the time dimension allows accounting for dynamic aspects of potential attacks and a considered system: the probability of a successful attack may change as time progresses; and a system may possess different data assets as its execution unfolds. For system modelling, we employ semi-Markov chains that are a powerful tool to capture system dynamics. For attack modelling, we adapt existing formalisms of attack trees and attack graphs. These models are used to analyse and quantify two important attributes of security: confidentiality and integrity. In particular, likelihood/consequence-based measures of confidentiality and integrity losses are proposed to characterise security risks to data assets. Identifying these risks in embedded systems is especially relevant in order to be able to trade them off against other constraints, e.g. limited resources. In our method, we consider attack and system behaviours as two separate models that are later elegantly combined for security analysis. This promotes knowledge reuse and avoids adding extra complexity in the system design process. We demonstrate the effectiveness of the proposed method and metrics on real smart metering devices.

1 Introduction

Concern for security issues in embedded systems is growing due to mass deployment of devices in a wide range of applications – from pace makers to critical infrastructures. Due to criticality of many such deployments and prevalent attacks, security should be an integral part of system development. Moreover, it should start already at an early design phase, because costs associated with security fixes grow steeply when progressing in the product life cycle. A serious obstacle to designing secure systems is that system engineers do not have suitable means to investigate how the decisions made at the design phase affect security. As a result, systems often remain unprotected. Our contribution to this research area is a method that enables analysing what security risks are inherent in a certain system model and how certain design decisions affect these risks. In particular, we address the risks related to data assets.

The output provided by our method can be used by system engineers to reason about the protection needed for data assets in the context of a given design and to evaluate these needs with respect to other factors such as economical costs. In particular, the contributions of this paper are as follows:

- Formal definition of two probabilistic risk-based metrics to quantify confidentiality and integrity of data assets in the context of a given design model.
- Development of a formal method to calculate these metrics using three types of models: functional and platform models of a system, and attack models.
- Illustration of how these metrics can be used to show different risks to the assets of a metering device which is part of a smart grid.

We suggest that security analysis should treat attack behaviour and system design as two separate, though interrelated, elements. Both elements are usually highly complex constituents of security. Treating them separately provides their more accurate elaboration, while both of them are clearly interdependent. One is addressed in the research on methodologies for attack modelling, e.g. an attack tree technique introduced[1] by Weiss [24]. Another is supported by evaluating a system design and identifying relevant assets [21]. This paper combines the two ideas as components in a bigger picture: a generic engineering process that exploits domain-specific security knowledge captured from security experts [22].

Attack and system behaviours are modelled as time-dependent probabilistic processes. The presence of the time dimension allows accounting for dynamic aspects of potential attacks and a considered system: the probability of a successful attack may change as time progresses, and a system may possess different valuable data assets as its execution unfolds. The use of probabilistic modelling, in turn, enables dealing with uncertainties that are present at the design phase.

One can potentially argue about difficulties of obtaining realistic data about the timing aspects of an attack and system at the design phase, and therefore, question reliability of results of the proposed security analysis. Nonetheless we propose that easier and more effective exploration of security threats and impacts is already a valuable input to design decisions, even when subject to some uncertainties. This enables asking 'what if' questions which help understanding the sensitivity of the system to potential attacks. Furthermore, the research that enables quantitative estimations of timing aspects of attacks and system at early design stages constantly progresses.

In Section 2, we discuss related works. Section 3 introduces two risk-based metrics by formalising the basic concepts and providing a method for their derivation. We illustrate the use of the proposed metrics in Section 4 on a real metering device in a smart grid, and conclude the paper in Section 5.

2 Related Work

There are general methods that cover main steps of risk analysis, e.g. CRAMM (CCTA Risk Analysis and Management Method) [1]. These methods prescribe basic steps of risk analysis while leaving out details of their implementation. Security risk management is also addressed by a set of standards, e.g. NIST SP800-30 [20] and ISO 31010 [3]. NIST creates foundations of risk management and contains basic guidance that broadly covers conducting risk assessment on federal IT systems and organisations. As pointed out by Lund et al. [15], this

[1] The term, however, was coined by Schneier.

guidance should be complemented with a risk analysis process and can not be considered as a fully-fledged method. Similarly, ISO 31010 provides guidance for risk assessment, but does not specify methods or techniques to be used in a specific application domain. Our work can be considered as complementary to these guidelines and is specific for design phases of embedded system development.

Surveys of risk analysis methods and corresponding metrics are presented by Verendel [23], among others. Verendel [23] analyses more than 100 approaches and metrics to quantify security. The conclusion of the author, that is highly relevant for our work, is that CIA (Confidentiality, Integrity, Availability) quantification methods are under-represented.

Several models for risk evaluation exist that have slightly different ingredients. For example, risk for physical security is often modelled as $R = P \times V \times D$ [4, 9] where P is the probability of threat occurrence, V is vulnerabilities of the system, and D is consequences. Another model represents security risks as $R = U \times C$ [15, 20] where U is the likelihood of unwanted incidents and C is consequences. These models resemble each other, i.e. the unwanted incident component in the latter model combines the first two components in the former. We base our work on the second model and focus on evaluating the U component.

Model-based risk analysis encompasses methods such as CORAS [15] and CySeMoL [19]. While following the same basic structure of risk analysis, these methods provide richer support. CORAS [15] is a general approach for risk analysis (already applied in numerous domains) specified as 8 steps. It uses graphical notation to structure elicited assets and threat scenarios and is supported by formal calculus that enables calculation of risks. Sommestad et al. [19] propose the Cyber Security Modeling Language (CySeMoL) and framework for risk assessment of IT systems. CySeMoL captures dependencies among different elements of risk analysis and system architecture, so that a system engineer is aided in deriving risks associated with an architecture. Similar to such works we use formal models for deriving risks with a focus on the design phase.

Vasilevskaya et al. [21] provide a model-based method for security analysis (referred to as SEED [22]). SEED is described at two levels of abstraction: foundation and realisation. At the foundation level the method prescribes system modelling and domain-specific security modelling. At the realisation level, the method is refined to provide support for embedded systems by employing modelling languages specific for the domain. Risk estimation to support quantitative security analysis has yet to be elaborated in SEED [22]. This paper fills this gap applying and extending the ideas of the model-based risk analysis.

Another group of methods for security quantification is based on attack modelling. A comprehensive survey of these approaches based on direct acyclic graphs is presented by Kordy et al. [14]. We briefly mention some approaches that are not in the scope of this survey, but still relevant to our work. Almasizadeh and Azgomi [5] define a state-based stochastic model to quantify different aspects of security. A system is abstracted away as a set of defender transitions, i.e. as backward transitions in an attack model. Madan et al. [16] propose a method to quantify security attributes of intrusion-tolerant systems. The analysis is done

based on an interwoven model of the intrusion-tolerant system, encompassing both attack and system behaviours, as well as countermeasure reactions.

The novelty of our approach for quantification of risks is that it explicitly accounts for both elements (system and attacks), but avoids mixing them in a single model specifically created for security analysis. Such an approach has several benefits. A system engineer and a security expert can work separately on the artefacts belonging to their own fields of responsibility and professional expertise. In addition, the approach enables the reuse of self-contained attack models across several systems in one application domain (e.g. different designs of metering devices from the smart grid domain), because these systems can face the same attacks. To our knowledge this is the first method that aims to support system engineers in quantifying risks to data assets in the context of embedded system design models from multiple stakeholder perspectives.

In line with current attack-based analysis frameworks, our methodology deals with known attacks as opposed to unknown zero-day attacks. Preparing a system for zero-day attacks is also an important concern, but most embedded systems fail to resist even known attacks due to a lack of security considerations at the design phase. Bilge and Dumitras [7] demonstrate that the number of occurrences of an attack when it is transformed from a zero-day to a known attack only grows.

Acquiring quantitative estimations (e.g. execution time) already at the design phase in embedded systems is a recognised challenge. Model-based engineering provides methods for design-based estimations to support early design-space exploration, e.g. COMPLEX [11], among others. We capitalise on such methods in order to decorate our models with probability distributions for execution time.

Our approach rests on input data that may be uncertain to some degree. For example, this is the case due to incomplete knowledge about attacks or uncertainty connected to estimates given by experts. Today, there are methods and tools to deal with limited certainty of data used for decision making. A comprehensive classification of different kinds of imperfect information and tools to deal with it are discussed by Parsons [18]. In our work, we adopt probabilities and probability distributions to capture inherent uncertainty of used data.

3 Quantifying Risks to Data Assets

We start this section by introducing two probabilistic risk-based metrics to quantify confidentiality and integrity of data assets. Thereafter, we present a method to calculate these metrics using formal system and attack models as inputs.

3.1 Proposed Metrics and Risks

Risk is frequently modelled by the likelihood of a successful attack and the severity of its consequences. Different ways of handling assets within a particular system imply different exposure of these assets to confidentiality and integrity breaches associated with a certain attack vector. The likelihood of a successful disclosure and alteration of assets is, therefore, conditioned by the ways assets are handled (similar to [4]). Since assets are objects of value, their violation will

imply different costs for different stakeholders [22]. As a result, confidentiality and integrity losses can be naturally defined as risk-based metrics parameterised by different stakeholders. Additionally, since both an attack and a system are dynamic entities, whose behaviour includes different exposure degrees at different states over time, the proposed metrics should be time-dependent.

We define *confidentiality loss* (CL) of a valuable data asset given an attack by time t as a risk metric that characterises the damage potentially caused by this attack on the asset. It is calculated as a product of the likelihood that the attack would disclose the asset by t and the cost of this breach for a stakeholder R. In turn, confidentiality loss of a system Y is a function (denoted by the symbol $\otimes$) of confidentiality losses for each data asset o_i that is subject to an attack A. The actual function will depend on the properties of the data assets in question and stakeholder's take on them.

$$CL(Y, A, R, t) = \otimes_i CL(o_i, A, R, t) \tag{1}$$

Similarly, *integrity loss* (IL) of a data asset from an attack by time t is a risk metric that characterises the effect from the potential alteration of the affected data asset. The notion is analogously extended to the system level. In the rest of this paper, we focus on confidentiality and integrity losses (CL and IL) for a single asset. Note that the definitions of confidentiality and integrity losses can be extended to the case when several attacks are present. However, for the sake of simplicity, we consider the case of one attack in this paper.

3.2 Basic Terms: Domain, Attack, and System

A domain is a notion that creates a common ground for system engineers and security experts. It is a broad concept in domain-specific modelling that includes many aspects. However, in this work, we say that a security expert and a system engineer work in the same application domain when they refer to a common set of components and objects while modelling respectively a system and attacks.

Def. 1 *A* domain *M is a tuple*[2] *(C, O) where C is a set of components and O is a set of data objects accessible in an application area. A set of assets is a subset of O, i.e. $Assets \subseteq O$.*

Attack modelling is a technique to capture behaviour of attacks. It can be formalised with attack trees or graphs. The basic elements of attack trees and graphs are attack steps and relations on them. Trees, additionally, have special elements such as gates, which are logical operations applied on attack steps (e.g. AND and OR), and root, which represents the goal of an attack. Thus, attack trees resemble fault and event trees in other dependability analyses. An alternative language for modelling attacks is (classes of) Petri Nets. Quantitative time aspects of attack models can be captured by assigning a probability distribution of execution time to attack steps. In our work, we use the term *basic attack*

[2] We use the term "tuple" as a finite ordered list (a sequence) of elements. Each element is addressed by its name in this paper.

model (which is neutral w.r.t. above languages) that captures the basic elements of an attack model needed for our analysis.

Def. 2 *A* basic attack model *is a tuple* (AS, AR, l_{AS}) *where: AS is a finite set of attack steps;* $AR \subseteq AS \times AS$ *is a relation between attack steps; and* $l_{AS} : AS \to \mathcal{F}$ *is a labelling function that associates execution time distributions from the set* $\mathcal{F}$ *to attack steps (AS).*

We extend this basic definition of an attack model with the *attack step annotation* concept. It enriches the definition of a basic attack model with *what*, *where*, and *how* information: *what* assets are targeted; *where* in a system (i.e. on which parts of a system platform); and *how* these assets are compromised, meaning which security attributes are violated.

Def. 3 *An* attack step annotation *is a tuple (TA, TC, AV) where:* $TA \in 2^O$ *is a set of* ***t****argeted* ***a****ssets;* $TC \in 2^C$ *is a set of* ***t****argeted* ***c****omponents;* $AV \in 2^{Attr}$ *is a set of security* ***a****ttributes* ***v****iolated by the attack step where* $Attr = \{Cnf, Int, Avl\}$*. We denote a set of such annotations by N and we refer to each element x of the attack step annotation as by as.x (e.g. as.TA).*

For example, if an attack step reads message $m \in O$ from a network component $link \in C$ then an annotation for this step is $(\{m\}, \{link\}, \{Cnf\})$; if an attack step only connects to some *link* then its annotation is $(\emptyset, \{link\}, \emptyset)$. These annotations allow relating an attack model to relevant elements of a system model. This enables combining attack models with system models. A basic attack model enriched with annotations is called an *annotated attack model*.

Def. 4 *An* annotated attack model *A is a tuple* (AS, AR, l_{AS}, l_N) *where* (AS, AR, l_{AS}) *is a basic attack model and* $l_N : AS \to N$ *is a labelling function that assigns an annotation to each attack step.*

For our analysis, we need to capture three aspects of a system: its functionality, execution platform and allocation information, and data object dependencies. These aspects are represented by *state model* and *data model*.

Def. 5 *A* state model *SM of a system is a tuple* (S, s_0, P, H, l_O, l_S) *where:*
- S *is the set of system states related to each other by a set of transitions;*
- s_0 *is the initial state;*
- $P : S \times S \to [0, 1]$ *associates a probability with a transition;*
- $H : S \to \mathcal{F}$ *associates a probability distribution with a state;*
- $l_O : S \to 2^O$ *is a labelling function that associates a set of objects from domain M with each state;*
- $l_C : S \to 2^C$ *is a labelling function that associates a set of components from domain M with each state.*

A state in S can be seen as a basic element of behaviour (e.g. an action) of a system. P and H formalise the probability of moving from one state to another and the probabilistic measure of execution time of each system state respectively. Thus, the first four elements of our state model form a semi-Markov chain [12]

(SMC). The latter two elements extend a SMC with additional information that is utilised to automatically combine system and attack models. Function l_O allows capturing the information about data objects which exist at a certain state. Function l_C associates the states with components of an execution platform.

Def. 6 *A* data model *DM of a system is a tuple* (D, l_D) *where:* $D \subseteq O \times O$ *is a relation that captures immediate object dependency;* $l_D : D \to 2^S \setminus \emptyset$ *is a labelling function that associates a set of states from* S *with each tuple in* D*.*

The relation D represents dependencies between data objects in a system. In particular, $(o_i, o_j) \in D$ means that an asset o_j depends on an asset o_i, e.g. $o_j = f(o_i)$ where f is some function. In this paper, we omit the nature and strength of such dependencies, but this information can also be used. Function l_D captures at which system state the dependencies in D occur. Thus, if $l_D(o_i, o_j)$ returns state $\{s\}$ then it means that o_j is derived from o_i in s. Implicitly, a well-formed data model associates a non-empty state set to every element in D.

Finally, a system model is a combination of state and data models.

Def. 7 *A* system model Y *is a tuple (SM, DM) where SM is a state model and DM is a data model.*

Table 1 summarises the notation and terms used in this paper.

3.3 Metrics and Their Derivation

Confidentiality loss (CL) caused by an attack A to each valuable data asset o by time t is a product of the likelihood that A would disclose o by t, and the cost of this disclosure to stakeholder R. In our context, the likelihood is a probability.

$$CL(o, A, Y, R, t) = p(o, A, Y, t)\ cost(o, R) \tag{2}$$

In equation (2), the cost of an asset, $cost(o, R)$, is a subjective estimate expressed by a stakeholder. In general case, the cost can also be time-dependent, but in this work we assume that it is time-agnostic. In turn, probability of a disclosure, $p(o, A, Y, t)$, can be broken down into a product of two events: (E_1) an attack A is in a step that can disclose o by time t; and (E_2) an asset o actually exists in system Y when it is attacked by time t.

$$p(o, A, Y, t) = p(E_1(o, A), t)\ p(E_2(o, Y), t) \tag{3}$$

In other words, the E_1 event accounts for a subset of attack steps $AS_{\langle Cnf,o\rangle} \subseteq AS$ that compromise an asset o and violate its confidentiality; and the E_2 event accounts for a subset of system states $S_{\langle o\rangle} \subseteq S$ that are associated with asset o. Additionally, the attack steps from $AS_{\langle Cnf,o\rangle}$ should target a set of components that are used for allocation of system states from $S_{\langle o\rangle}$. This simply means that, for the attack to be successful, a system should have components with certain targeted vulnerabilities exploited by the attack steps from $AS_{\langle Cnf,o\rangle}$. We refer to this subset of targeted components as C_{target}.

Table 1. Summary of the used notation

Sets and subsets	
C – components	AV – security **a**ttributes **v**iolated
O – objects	N – attack step annotations
$Assets$ – assets, $Assets \subseteq O$	S – system states
$\mathcal{F}$ – probability distributions	$\Omega_{o,o'}$ – all state sequences between o' and o
AS – attack steps	$S_{\langle o \rangle}$ – system states where object o exists
TA – **t**argeted **a**ssets	C_{target} – system components targeted by an attack
TC – **t**argeted **c**omponents	$AS_{\langle Cnf,o \rangle}$ – attack steps violating confidentiality of an object o
Functions, dependencies and relations	
l_{AS} – assigns execution time probability distributions to attack steps, $l_{AS} : AS \to \mathcal{F}$	
l_N – assigns an annotation to an attack step, $l_{ASN} : AS \to \text{ASN}$	
P – associates a probability with a transition, $P : S \times S \to [0, 1]$	
D – a dependency relation between data objects, $D \subseteq O \times O$	
l_D – associates a set of system states with a data dependency, $l_D : D \to 2^S$	
H – associates a probability distribution of execution time with a state, $H : S \to \mathcal{F}$	
l_O – associates a set of existing objects with a state, $l_O : S \to 2^O$	
l_C – associates a set of components with a system state, $l_C : S \to 2^C$	
AR – a relation between attack steps, $AR \subseteq AS \times AS$	
$cost$ – a cost of asset disclosure or alternation expressed by a stakeholder	
κ – a function that checks whether there is a transitive dependency between two objects	
Tuples	
$SM = (S, s_0, P, H, l_O, l_C)$ – a state model	$M = (C, O)$ – an application domain
$DM = (D, l_D)$ – a data model	$Y = (SM, DM)$ – a system model
$A = (AS, AR, l_{AS}, l_{ASN})$ – an annotated attack model	
Other	
R – a stakeholder	PE – propagation effect
CL – confidentiality loss	DE – direct effect
IL – integrity loss	ϕ – interval transition probability
	ω – sequence of states
	γ – sequence of data objects

Given a set of states S and a set of attack steps AS of an attack A, we define a set of targeted components (C_{target}), a set of attack steps disclosing an object o ($AS_{\langle Cnf,o \rangle}$), and a set of states where an object o is potentially compromised ($S_{\langle o \rangle}$):

$$C_{target} = \{c \mid s \in S, as \in AS, c \in l_C(s) \cap as.TC\} \tag{4}$$

$$AS_{\langle Cnf,o \rangle} = \{as \mid as \in AS, as.TC \cap C_{target} \neq \emptyset, o \in as.TA, Cnf \in as.AV\} \tag{5}$$

$$S_{\langle o \rangle} = \{s \mid s \in S, l_C(s) \cap C_{target} \neq \emptyset, o \in l_O(s)\} \tag{6}$$

To put it another way, execution of any attack step in $AS_{\langle Cnf,o \rangle}$ leads to disclosure of a given object o, which is essentially the E_1 event. This corresponds to construction of an attack tree with attack steps from $AS_{\langle Cnf,o \rangle}$ which are all connected by the OR gate. This is expressed in equation (7).

$$p(E_1(o, A), t) = 1 - \prod_{as \in AS_{\langle Cnf,o \rangle}} \big(1 - p(as, t)\big) \tag{7}$$

Finally, the probability of E_2 is the sum of probabilities that the system Y will transit to each state from $S_{\langle o \rangle}$ where asset o exists. Thus,

$$p(E_2(o,Y),t) = \sum_{s \in S_{\langle o \rangle}} \phi(s_0, s, t) \tag{8}$$

In equation (7), $p(as,t)$ is a probability of success of an attack step as by time t. It is returned by l_{AS} given t that is, in turn, calculated from a selected modelling formalism for attack step relations (e.g. attack trees or graphs). In equation (8), $\phi(s_0, s, t)$ is a so called interval transition probability of the system Y transiting from a state s_0 to a state s in interval $(0,t)$ [12]. It is calculated from the system equation that describes the dynamics of an underlying SMC.

Integrity loss (IL) is a metric that defines the risk of alterations to an asset o, and should account for two aspects: the *direct effect* (DE) – the loss caused by the direct influence of an attack A on asset o; and the *propagation effect* (PE) – the loss caused by spreading corrupted data and further contaminating the computations dependent on asset o.

$$IL(o,A,Y,R,t) = DE(o,A,Y,R,t) + PE(o,A,Y,R,t) \tag{9}$$

The reason to include the propagation effect in the IL metric, but not in the CL metric can be explained with the following rationale. Whether a breach of confidentiality will propagate depends on specific attack capabilities, i.e. if an attack is capable of learning additional data when it has observed a part of it. This learning is a self-contained attack step and should be explicitly included in an attack model. For example, the sole fact that an attack compromises an encryption key does not directly imply that all data encrypted by this key is compromised; it is compromised if an attack actually reads this data. In contrast, a breach of integrity propagates independently from an attack, but it depends on the system behaviour. For example, if a public key is modified, then data signed by this key can not be decrypted if the decryption state is part of a system.

The direct effect DE is calculated in analogy to CL, where $AS_{\langle Int,o \rangle}$ is defined by equation (5) and $Int \in as.AV$ replaces $Cnf \in as.AV$.

The intuition for the propagation effect is as follows: if an object $o' \in O$ is computed in a state $s' \in S$ based on an object o that has already been corrupted in a state $s \in S_{\langle o \rangle}$ then o' is also considered corrupted. To derive this effect PE with respect to each such object o' we consider the four aspects described below.

First, we need to check whether o' immediately or transitively depends on o. The immediate dependency is already captured in the data model DM by $(o,o') \in D$. We say that transitive dependency exists, if it is possible to construct such a sequence of objects γ of length n that $\gamma = (\gamma_k \mid \gamma_1 = o, \gamma_n = o', 1 \leq k < n : (\gamma_k, \gamma_{k+1}) \in D)$. We formalise this test as a function $\kappa : O \times O \to \{0,1\}$ that returns 1 if such a sequence γ exists, otherwise it returns 0.

The next two elements are the cost of o' as expressed by a stakeholder R and the probability that o will be actually attacked by time $\tau \leq t$.

Finally, the propagation effect occurs only when the system Y will transit from a state $s \in S_{\langle o \rangle}$ to a state s' where o' is computed from o immediately or transitively. Such a state s' can be returned by the labelling function l_D, if immediate dependency between o and o' exists. However, if an immediate

dependency does not exist, but a transitive dependency exists then we need to consider a sequence of states ω (of length $n-1$) along which the transitive dependency, captured by a sequence of objects γ, occurs. We construct ω in such a way that $\omega = (\omega_k \mid 1 \leq k < n-1 : \omega_k \in l_D((\gamma_k, \gamma_{k+1})))$. Since there can be several valid sequences of states relating o and o', we denote by $\Omega_{o,o'}$ the set of such state sequences. In other words, $\Omega_{o,o'}$ is the set of all state sequences along which o' can be compromised when o is attacked.

The propagation effect given the four elements above is calculated as follows:

$$PE(o, A, Y, R, t) = \sum_{o' \in O} \kappa(o, o')\ cost(o', R) \sum_{s \in S_{\langle o \rangle}} p(E_1(o, A), \tau) \sum_{\omega \in \Omega_{o,o'}} P(s_0, s, \omega, t) \tag{10}$$

In equation (10), $P(s_0, s, \omega, t)$ is the interval transition probability that the system that starts at s_0 will first pass the state s (where asset o is attacked by A), and then will go through each state from (any possible) sequence ω. This probability can be computed recursively as follows:

$$P(s_0, s, \omega, t) = \phi(s_0, s, \tau)\ P(s, \omega_1, \omega_{[2..]}, t - \tau) \tag{11}$$

We denote by ω_1 the first element of the sequence ω and by $\omega_{[2..]}$ a suffix of this sequence starting from the 2^{nd} element. The validity of equation (11) can be proven by induction (omitted due to space restrictions).

4 An Illustrative Example

In this section we apply our methodology on an open platform device developed based on the design from the European project SecFutur [2], where a Trusted Sensor Network (TSN) was a case study. We illustrate how confidentiality and integrity losses capture the security risks associated with data assets.

The TSN consists of a set of meters, servers, and a communication infrastructure. TSN collects measurements of energy consumption at households for billing purposes. This infrastructure has a range of diverse security considerations and data assets: measurements, user account data, a set of certificates, commands, etc. Here we study a *metering device* and, for simplicity, we focus on *measurements* as an asset under analysis.

Table 2. Stakeholder costs expressed for measurements

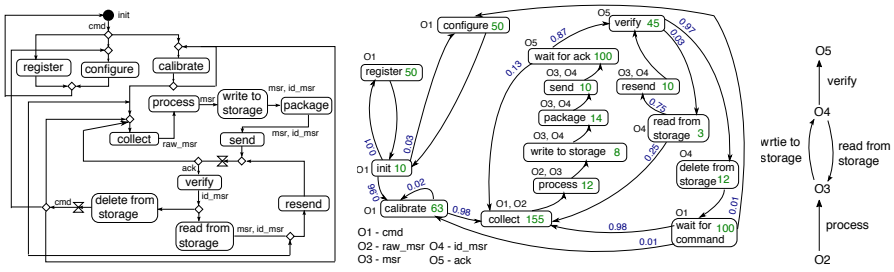
	User	**Utility provider**	**National regulatory agency**
Confidentiality	major	minor	insignificant
Integrity	moderate	major	minor

We consider three stakeholders: user, utility provider, and national regulatory agency. The stakeholder costs of losing confidentiality or integrity for measurements are shown in Table 2. We adopt a common linguistic scale [15] {*insignificant, minor, moderate, major, catastrophic*} to encode these costs, which we further map to a numerical vector $\{0, 0.1, 0.5, 0.8, 1\}$. Note that we use this simplified numerical scale only for exemplification. In practice, one can use intervals or even continuous scales, as our methodology does not impose specific requirements on the form of the scale.

4.1 System and Attack Modelling

Figure 1(a) depicts a system model for a metering device from the TSN scenario. The system expects a command (*cmd*) from an administrator of a utility company. On receipt, it proceeds to registration, configuration, or calibration procedures. When the device is calibrated it starts collecting data (*raw_msr*), processing it into ready measurements (*msr*), writing them into the memory and creating a unique id (*id_msr*), sending them to the server, and waiting for an acknowledgement (*ack*). If *ack* has not arrived a meter continues to collect measurements; otherwise, it proceeds to the verification procedure. If verification succeeds the device searches for the measurement by id (*id_msr*), deletes this measurement from storage, and waits for the next command from the server. If verification fails, the device reads the measurement from storage and resends it.

Construction of state and data models (introduced in Section 3.2) can be accomplished by traversing and transforming control and data flows of the UML activity model. UML activity diagrams can be directly transformed into a state model [17]. Alternatively, UML activities can be first transformed into some variant of Stochastic Petri Nets (SPNs) [8]. They offer easier translation from UML activity diagrams and also provide great capabilities such as dealing with concurrency and various forms of dependency. These variants of SPNs can be further used to generate Markov and none-Markov models, i.e. SMC in our case. For the purpose of this illustration we directly move on to the state model and omit intermediate steps. A data model can be obtained by traversing the UML activity and utilising its built-in data flows. Note that we use the UML activity as an example. Our approach is not limited to this choice.



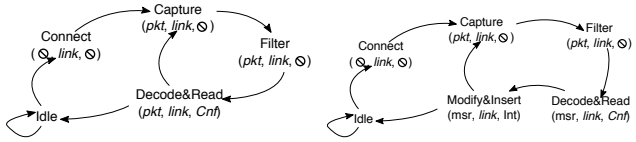
(a) UML Activity (b) *SM* (c) *DM*

Fig. 1. Illustration of the formalised system model for the metering device

Figure 1(b) depicts the state model (*SM*). The numbers on arcs are probabilities of the transitions (P) and the numbers on states are mean state execution times from normal distributions (H). To obtain this data, design phase estimation methods outlined in Section 2 exist, but we employ prototyping as a viable alternative. The object references next to each state correspond to the labelling function l_O. Our metering device model includes a simplified execution platform built of two components: a *link* and a *device*. The *link* is a communication network, and the *device* is the basic meter. The *init*, *package*, *send*, *wait for acknowledgement*, *resend*, and *wait for command* states are allocated on the

link, and the rest of the states are executed on the *device*. This corresponds to the labelling function l_C. Figure 1(c) depicts the data model (DM), where the labels on dependency arcs are names of corresponding states from the state model (SM). This corresponds to the labelling function l_D.

Privacy and integrity of measurements are two serious concerns for smart meters [13]. Privacy can be violated by eavesdropping energy consumption measurements passed over the communication network. Integrity of measurements can be broken by spoofing the original data sent by a meter. Here we consider these two types of attack. Respective annotated attack models in the form of attack graphs are depicted in Figures 2(a) and 2(b). To eavesdrop an attacker should intercept the data packets (*pkt*) sent over the network (*link*), i.e. to sniff, filter and decode them. To perform a spoof attack the data packets should also be intercepted and then inserted back into the network in a modified form.



(a) Eavesdropping (b) Spoofing

Fig. 2. Two attacks against measurements

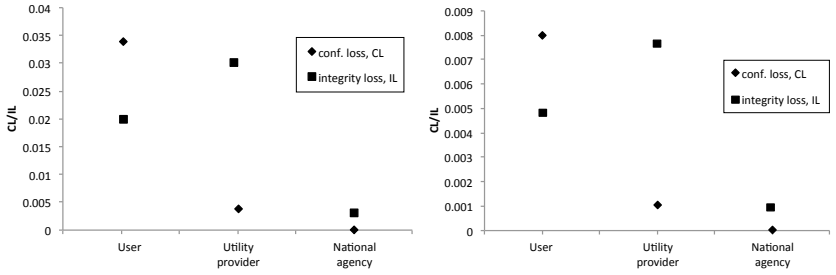
In addition to annotations, each attack step is associated with a probability distribution (the labelling function l_{AS}). Arnold et al. [6] discuss two alternative approaches to obtain such distributions: (1) based on historical or empirical data; and (2) based on expert opinions. For our illustrative example, we employ the former approach by running experiments with the constructed prototype. We use kernel density estimations [10] to obtain the needed distribution functions.

4.2 Calculating Metrics

A combination of the attack and system models gives the following sets of compromising attack steps and system states (from equations (4)-(6)): $AS_{\langle Cnf,msr\rangle} = \{Decode\&Read\}$, $AS_{\langle Int,msr\rangle} = \{Modify\&Insert\}$, and $S_{\langle msr\rangle} = \{package,\ send,\ resend\}$.

In this section we show only the values observed when confidentiality (CL) and integrity losses (IL) stabilise. In reality the risks can still change over time, which is also the case in our illustrative example, but for brevity we do not show the whole trend since the risks stabilise relatively fast.

Figure 3(a) shows that confidentiality loss for the user is about 9 times higher than for the utility provider (0.035 against 0.004), integrity loss is highest for the utility provider (0.031), and the national agency bears the lowest risk both in terms of confidentiality and integrity losses. It should be mentioned that due to a narrowed down set of considered assets (i.e. measurements only) in our example, stakeholder-wise comparison of the confidentiality and integrity loss metrics is not as informative as it could be in a general case with multiple assets. However, what our example demonstrates distinctly is how the proposed metrics reflect reduction of risks when mitigation measures are applied. That, in



(a) Initial design (b) Modified design

Fig. 3. *CL* and *IL* in the initial design and after mitigation by modification

turn, indicates how each stakeholder benefits when the initial design is modified for strengthening its security aspects [22].

Now, we illustrate how a modification of a design can act as a mitigation against the two attacks. We modify the state "collect", so that the system in Figure 1 sends measurements in chunks of 10 readings. By this, the mean execution time of this state is changed from 155 to 1550 ms. Figure 3(b) shows results for an improved system, and we observe a significant drop in original risks. In particular, confidentiality loss for the user and utility provider drops down by a factor 4 in comparison with the risks derived from the initial design. Similarly, a significant drop is observed for integrity loss, i.e. *IL* for all stakeholders is 3-4 times lower than in the original design.

Once the risks for the stakeholders are calculated as confidentiality and integrity losses, they should be assessed. This is an extensive decision problem that typically involves other criteria (e.g. resource footprint of countermeasures, quality of service requirements, etc.). The following steps in risk assessment are out of the scope of this paper and is treated elsewhere.

5 Conclusion and Future Work

In this paper we formalised confidentiality and integrity losses as two probabilistic metrics that quantify risks associated with data assets within embedded systems. Our proposed metrics account for system design, attack scenarios, and different stakeholder preferences regarding data assets. We applied the metrics on a smart metering device and showed their use for visualising of and reasoning about security risks. In addition, we illustrated how our methodology allows analysing the impact of design decisions on the risks in question, demonstrating their potential to increase security awareness of engineers within early design stages. For future work, we aim to extend the tool set developed for the SEED process [22] by integrating this methodology, and enable trading off risks against other criteria, e.g. resources efficiency, when selecting suitable security measures.

References

1. CCTA Risk Analysis and Management Method. `www.cramm.com`, visited Oct. 2013.

2. The SecFutur project: Design of Secure and Energy-efficient Embedded Systems for Future Internet Application. `http://www.secfutur.eu`.
3. IEC/ISO 31010 – Risk Management – Risk Assessment Techniques. 2009.
4. DHS Risk Lexicon. Technical report, DHS Risk Steering Committee, 2010.
5. J. Almasizadeh and M. A. Azgomi. A Stochastic Model of Attack Process for the Evaluation of Security Metrics. *Journ. Comp. Networks. Elsevier*, 57(10), 2013.
6. F. Arnold, H. Hermanns, R. Pulungan, and M. Stoelinga. Time-dependent Analysis of Attacks. In *Conf. on Principles of Security and Trust*. Springer, 2014.
7. L. Bilge and T. Dumitras. Before We Knew It: An Empirical Study of Zero-Day Attacks In The Real World. In *ACM Conf. on Comp. and Comm. Security*, 2012.
8. G. Ciardo, R. German, and C. Lindemann. A Characterization of the Stochastic Process Underlying a Stochastic Petri Net. *IEEE Trans. on Soft. Eng.*, 20(7), 1994.
9. F. Flammini, S. Marrone, N. Mazzocca, and V. Vittorini. Petri net modelling of physical vulnerability. In *Workshop Critical Information Infrastructure Security*. Springer, 2013.
10. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
11. F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, R. Valencia, and G. Palermo. The COMPLEX Methodology for UML/MARTE Modeling and Sesign Space Exploration of Embedded Systems. *Journ. of Syst. Archit. Elsevier*, 60(1), 2014.
12. R. A. Howard. *Dynamic Probabilistic Systems*. John Wiley & Sons, 1971.
13. M. E. Jobst. Security and Privacy in the Smart Energy Grid. In *Smart grid security workshop at CSS*. ACM, 2014.
14. B. Kordy, L. Pièvre-Cambacédès, and P. Schweitzer. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *Computer Science Review. Elsevier*, 13-14, 2014.
15. M. S. Lund, B. Solhaug, and K. Stølen. *Model-Driven Risk Analysis: The CORAS Approach*. Springer-Verlag Berlin Heidelberg, 2010.
16. B. B. Madan, K. Goševa-Popstojanova, K. Vaidyanathan, and K. S. Trivedi. A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems. *Performance Evaluation. Elsevier*, 56(1-4), 2004.
17. S. Ouchani, O. Mohamed, and M. Debbabi. A Formal Verification Framework for SysML Activity Diagrams. *Journ. on Expert Systems with Applications. Elsevier*, 41(6), 2014.
18. S. Parsons. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Trans. on Knowledge and Data Engineering*, 8(3), 1996.
19. T. Sommestad, M. Ekstedt, and P. Johnson. A Probabilistic Relational Model for Security Risk Analysis. *Computers & Security. Elsevier*, 29(6), 2010.
20. G. Stoneburner, A. Y. Goguen, and A. Feringa. SP 800-30. Risk Management Guide for Information Technology Systems. In *NIST*, 2002.
21. M. Vasilevskaya, L. A. Gunawan, S. Nadjm-Tehrani, and P. Herrmann. Integrating Security Mechanisms into Embedded Systems by Domain-specific Modelling. *Journal of Security and Communication Networks, Wiley*, 7(12), 2013.
22. M. Vasilevskaya and S. Nadjm-Tehrani. Model-based Security Risk Analysis for Networked Embedded Systems. In *Conf. on Critical Information Infrastructures Security*. Springer, 2014.
23. V. Verendel. Quantified Security is a Weak Hypothesis: A Critical Survey of Results and Assumptions. In *New security paradigms workshop, ACM*, 2009.
24. J. Weiss. A System Security Engineering Process. In *National Computer Security Conference*, 1991.