

Article

# Formalizing Attack Tree on Security Object for MySANi in Legal Metrology

Muhammad Azwan Ibrahim <sup>1</sup>, Faizan Qamar <sup>2,\*</sup>, Zarina Shukur <sup>2</sup>, Nasharuddin Zainal <sup>3</sup>, Nazri Marzuki <sup>1</sup> and Maria Ulfah Siregar <sup>4</sup>

<sup>1</sup> Electrical Group, National Metrology Institute of Malaysia, Sepang 43900, Malaysia

<sup>2</sup> Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia

<sup>3</sup> Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia

<sup>4</sup> Department of Informatics, Faculty of Science and Technology, UIN Sunan Kalijaga, Yogyakarta 55281, Indonesia

\* Correspondence: faizanqamar@ukm.edu.my

**Abstract:** Illegal software manipulation is one of the biggest issues in software security. This includes the legally relevant software which are now crucial modules in weight and measuring instruments such as weighbridges. Despite the advancement and complexity of weight and measuring instruments, the inspection methodology is weak and lacks of innovation. The conventional inspection method is merely based on the observation printed certificate of the software. This paper introduces Malaysia Software-Assisted Non-Automatic Weighing Instrument (NAWI) Inspection (MySANi), a method used to enhance the software inspection scheme in legal metrology. MySANi introduces security objects in order to assist and enhance the inspection process. The security evaluation is based on the best practices in IT in metrology, where the attack model on relevant assets of the security objects is simulated for the Attack Probability Tree. The attack tree is verified by integrating formal notation and comparison with finite state transition system domain to verify the correctness properties of the tree design before the model can be further used in a risk analysis procedure within the Attack Probability Tree framework. Results show that the designed attack tree is consistent with the designed simulation.

**Keywords:** formal model; specification techniques; infrastructure protection; security; integrity; law



**Citation:** Ibrahim, M.A.; Qamar, F.; Shukur, Z.; Zainal, N.; Marzuki, N.; Siregar, M.U. Formalizing Attack Tree on Security Object for MySANi in Legal Metrology. *Systems* **2023**, *11*, 49. <https://doi.org/10.3390/systems11010049>

Academic Editor: Vladimír Bureš

Received: 16 December 2022

Revised: 10 January 2023

Accepted: 11 January 2023

Published: 16 January 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Pattern approval (PA) is an examination and evaluation of regulated measuring instruments conducted by an impartial body to design an instrument prototype against national or international standards or statutory requirements [1]. This exercise is part of the legal metrology (LM) ecosystem to determine whether the measuring instrument is suitable for the intended use and can retain its accuracy and function under various environmental and operating conditions [2].

During the PA process, software evaluation, verification, and assessment are crucial for guaranteeing a credible and seamless operation of weighing and measuring equipment and systems [3]. The penalty for using forged measuring instruments is outlined in Section 17 of the Weights and Measures Act 1972 [4].

Despite the sophistication and complexity of measuring instruments, software enforcement framework within LM is poor. Furthermore, based on our literature studies, we found that there lack of best practices in tree structure and comprehensive tree design.

This paper introduces Malaysia Software-Assisted Non-Automatic Weighing Instrument (NAWI) Inspection (MySANi) which is a designed method in inspecting the compliance of universal computer-based software for NAWI within the LM framework in

Malaysia [5]. Security objects within MySANI are introduced, which is novel in the LM ecosystem. Thus, this proposed method enhances the overall inspection process within the LM framework in Malaysia. The paper demonstrates the use of SAND operator in the attack tree design and subsequently proving it using formal method against finite state transition which was impossible in the previous studies.

## 2. Related Works

In software engineering, risk analysis is one of the processes in risk assessment. It is a measuring stick for evaluating the effectiveness of security design of a system [6]. Around 50% of security problems are due to design weaknesses. Thus, the risk analysis performed at the design construction stage is very important for software security programs.

At the design stage, any risk analysis should be tailored to the software design [7]. Risks associated with threats can be modelled as risk = impact × likelihood.

Understanding the harm that an asset expose is necessary for effective asset protection. Risk assessment is used as a tool for information as an asset [8]. In information security, risk involves three main concepts: threat, vulnerability, and impact.

According to ISO/IEC 27005, the term risk is “combination of the consequences that would follow from the occurrence of an unwanted event and the likelihood of the occurrence of the event”. Figure 1 shows the components in risk analysis.

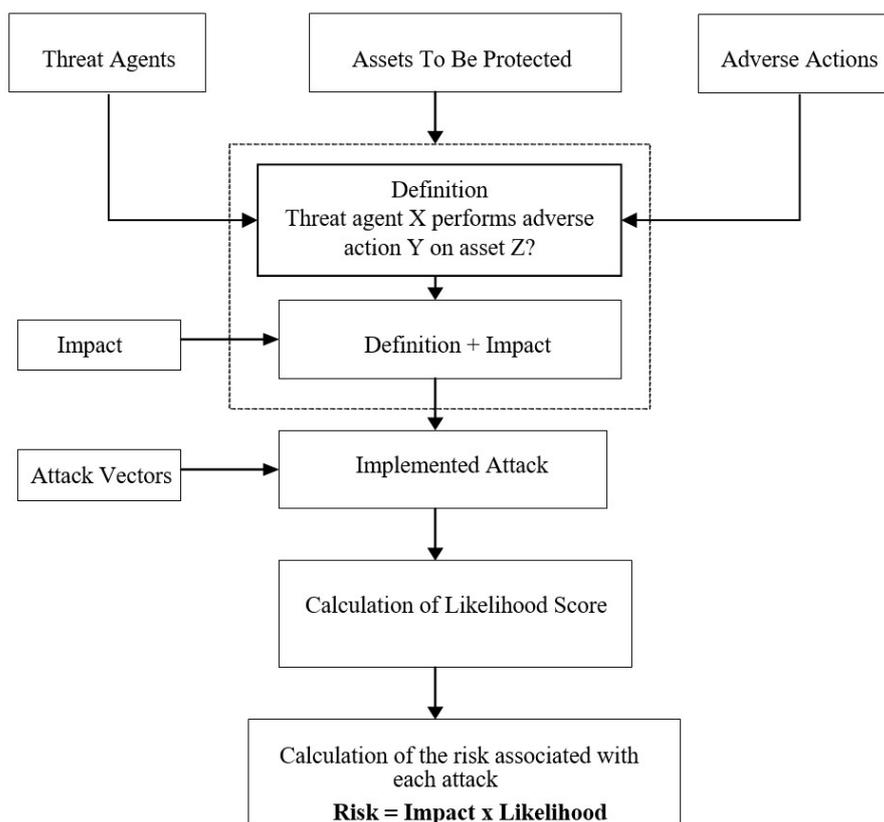


Figure 1. Component of risk analysis.

### 2.1. Security Evaluation in Legal Metrology

In LM, risk analysis is used as a tool to evaluate the security of a designed system or instrument. In a recent study, a method based on ISO/IEC 27005, 15408, and 18045 was developed by Esche and Thiel [9] to provide the foundation of security analysis in LM domain. However, this technique justifies the security of measuring instruments based merely on the device’s technical features alone.

This method later was improved by incorporating the attacker's motivation. It was performed by integrating an attack tree modeling as part of the evaluation framework which is known as Attack Probability Tree (AtPT) [10].

## 2.2. Attack Tree

An attack tree is a model for visualizing and analyzing potential security threats [11]. This method has been adopted by the North Atlantic Treaty Organization (NATO) as well as the Open Web Application Security Project (OWASP) [12]. Several studies have been conducted by integrating attack trees with risk management methodologies such as supervisory control and data acquisition (SCADA), radio-frequency identification (RFID), and automated teller machine (ATM).

ATSyRA [13] made an approach by extracting a defined filter structure from an expert. This allows the semantic domain and structure of the filter to be found without intervention from experts. However, this method has imperfect (informal) label structure.

TREsPASS [14] applies the security knowledge base to the attack tree generated from the model system. However, the generic IT security knowledge base is incompatible for the LM situation.

Another formal design methodology is ADTool [15]. This method adopts the ADTree-based attack tree concept [16]. However, ADTree has a different approach with two types of players: strikers and defenders. This methodology, however, is not compatible with the AtPT scoring semantics used in the LM based on ISO/IEC 18045.

Attack tree design is a subjective procedure according to [17]. Esche in [10] emphasizes that in the field of LM, all measuring instruments share the same basic characteristics. Based on the fact that all instruments are based on the same technical solution it should be that the diversity of attack tree designs produce the same results.

Although the attack tree design is considered simple and intuitive, there is room for conceptual errors and a diversity of interpretations regarding the semantic refinement of the tree. Thus, there lack of best practices in tree structure and comprehensive tree design.

## 3. Research Methodology

Attack tree design in an AtPT framework is enhanced in this study where formal notation replaces the original version used only text to the name for each node [18]. We also improved the attack tree by adapting the SAND operator where the original model did not have it.

The attack tree decoration phases are formalized in mathematical notation in both the attack tree domain and finite state transition domain. This is to validate the correctness properties by using a method introduced by [19].

Table 1 shows the important notations that are used in the study.

**Table 1.** Notation with description.

Notation	Description
$Prop$	Set of propositions
$\iota$	Initial configuration
$\gamma$	Final configuration
$\mathcal{S}$	State transition system
$S$	Finite transition set
$s$	Element/state
$\mathbb{N}$	A set of natural numbers
$\lambda: Prop \rightarrow 2S$	Labelling function
$\Pi, \rho$	Path
$v$	Attack vector
$T$	Attack tree
$\alpha$	Set of attributes
$\mathcal{R}$	Risk scoring
$\langle \iota, \gamma \rangle$	Leaf/goal
$OP$	Operator

### 3.1. Finite State Transition

$Prop$  is a set of propositions containing all the variables representing a situation at one time. This  $Prop$  is in the form of  $\iota, \gamma$  to denote the state before ( $\iota$ ) and after an event ( $\gamma$ ). In this context the event is an attack vector  $v$ .

A state transition system is a tuple  $\mathcal{S} = (S, \rightarrow, \lambda)$ , where  $S$  is the finite transition set (elements of  $s, s_i$ ) for  $i \in \mathbb{N}$  while  $\rightarrow \subseteq S \times S$  is the transition relation to the system and assumed to be the left total. The  $\lambda: Prop \rightarrow 2S$  is the labeling function. A state  $s$  is labelled as  $p$  when  $s \in \lambda(p)$  and the size of  $S$  is  $|S| = |S| + |\rightarrow|$ .

A path in system  $S$  is the state condition where it has non-empty sequences of states. The symbols such as  $\pi$  and  $\rho$  are used to depict the paths. A cycle in paths  $\pi \in \Pi(S)$  is a factor of  $\pi'$  of  $\pi$  where  $\pi'(0) = \pi'(|\pi'|)$ .

An elementary path is a path which has no cycle. A path  $\pi$  which is elementary, does not have any state that more than once. Therefore,  $|\pi| \leq |S|$ , removing all the cycles from  $\pi$  iteratively until the resulting path is non-elementary. Table 2 summarizes the attack tree critical analysis.

**Table 2.** Attack tree critical analysis.

Description	Esche	Audinot
Tree Model	generic	formal
Refinement	OR, AND	OR, AND, SAND
Node Name	informal, ordinary text-based	formal notation $\langle \iota, \gamma \rangle$
Node Description	action-based	state-based
Design validation	unavailable	finite state transition

### 3.2. Formalizing Attack Tree

In conjunction with the analysis of AtPT methodology [20], attack vector  $v$  for attack tree  $T$  over a set of attack vector attributes (Table 3) is the refinement for the path  $\pi$  from with the direction from  $\iota$  to  $\gamma$ . Assuming  $v \in \rightarrow$ , where  $v_1, v_2, \dots, v_n$  is the attack vectors for each  $T_1, T_2, \dots, T_n$ , for  $1 \leq i \leq n$ . The size of attack vector is  $|v| = |T| \ni \forall T_n \exists v_n$ . The attack vector evaluation consists of conditions  $OP \in \{OR, AND, SAND\}$  and has the number of arity  $n \geq 2$ . The main goal of a an attack vector  $v$  is  $\gamma$  where the original state is  $\iota$  and where the operator is  $OP$ . The attack tree can be represented in the form of  $T = (\langle \iota, \gamma \rangle, OP)(T_1, T_2, \dots, T_n)$ . Where the size of an attack tree  $|T|$  is the number of attack vectors in  $T$  such that  $|\langle \iota, \gamma \rangle| = 1$  and  $|\langle \iota, \gamma \rangle, OP)(T_1, T_2, \dots, T_n)| = 1 + \sum_{i=1}^n |T_i|$ .

**Table 3.** Attack vector attributes.

Symbol	Attribute
$Du$	time required
$Ex$	expertise
$Kn$	knowledge required
$Wn$	windows of opportunity
$Eq$	equipment required

Let the set of attributes  $\alpha = \{Du, Ex, Kn, Wn, Eq\}$  for all  $v_1, v_2, \dots, v_n$  be the attack vectors for  $T_1, T_2, \dots, T_n$ . While the  $R$  be the AtPT risk scoring calculation based on the  $OP$  over the set  $\alpha$ . Therefore,  $v_i = Z(OP, \alpha)$  where,  $OP \in \{OR, AND, SAND \mid 1 \leq i \leq n\}$ .

### 3.3. Attack Tree Correctness Property

According to [21], an attack tree is considered to have the meet property when:

$$\llbracket OP(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \dots, \langle t_n, \gamma_n \rangle) \rrbracket^S \cap \llbracket \langle t, \gamma \rangle \rrbracket^S \neq \emptyset$$

While undermatch property has:

$$\llbracket OP(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \dots, \langle t_n, \gamma_n \rangle) \rrbracket^S \subseteq \llbracket \langle t, \gamma \rangle \rrbracket^S$$

And overmatch reflects:

$$\llbracket OP(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \dots, \langle t_n, \gamma_n \rangle) \rrbracket^S \supseteq \llbracket \langle t, \gamma \rangle \rrbracket^S$$

Finally, the match property is:

$$\llbracket OP(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \dots, \langle t_n, \gamma_n \rangle) \rrbracket^S = \llbracket \langle t, \gamma \rangle \rrbracket^S$$

In other words, the meet property is a minimum property for an attack tree before further analysis can be performed. This proves that at least one path can achieve the main goal and its refinements. This feature indicates that this tree model is essentially considered to be correct, and researchers can already begin to estimate the security of a system.

Undermatch shows that all paths that meet the refinement of a node also meet the goal itself. It can be considered as an underestimate for a set of scenarios and is enough to find the weaknesses of a system. While overmatch, on the other hand indicates that all paths that achieve the root node goal also met all decomposition to sub-goals. This is an assumption. If the tree characteristics are found to be a match, then it is considered that a tree has one hundred percent similarity with the system analyzed.

A tree should have a minimum of admissible and meet features for a tree design to be considered correct with respect to the analyzed systems.

## 4. Malaysian NAWI Software Inspection for Computer

Malaysia Software-Assisted Non-Automatic Weighing Instrument Inspection (MySANI) is a proposed method designed to enhance the inspection process within the framework of the LM ecosystem in Malaysia as depicted in Figure 2 [22]. It consists of two parts, which are PA and market surveillance.

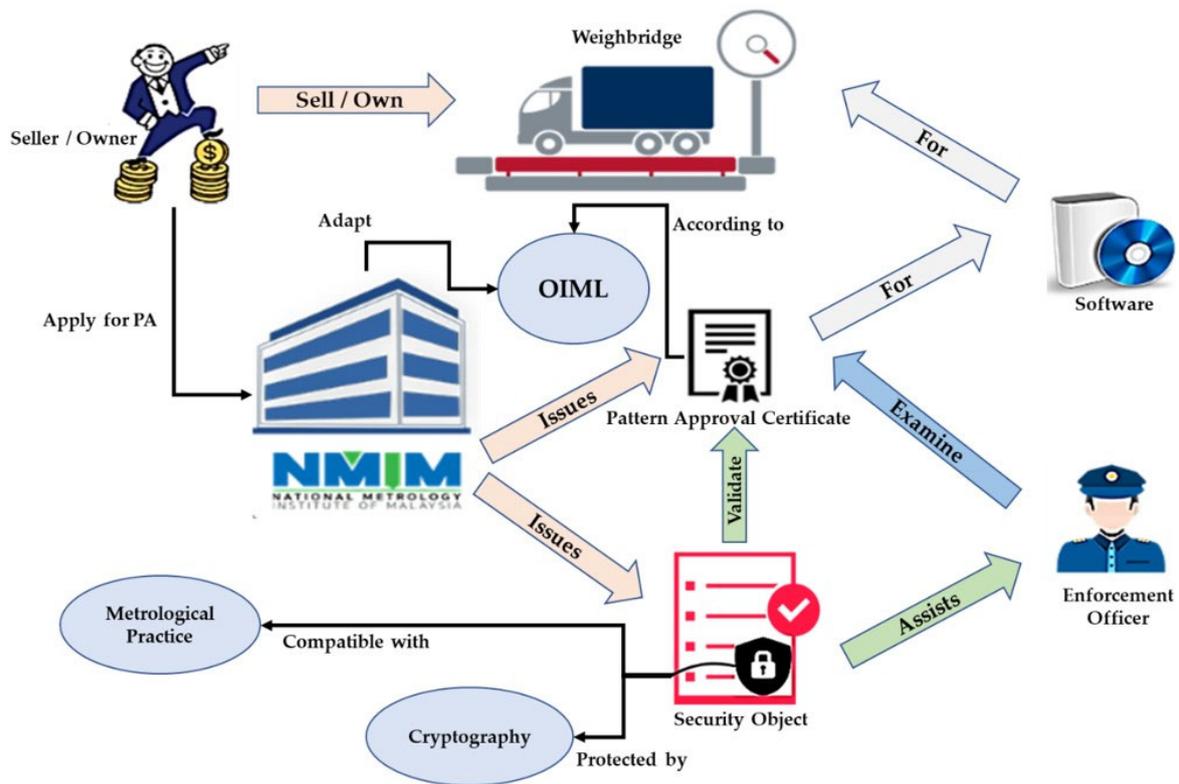


Figure 2. Contextual framework.

In Malaysia, PA is stated in the Weights and Measures Act 1972 (Act 71) and the National Measurement System Act 2007 (Act 675). The PA is provided for the National Measurement Standard Laboratory (NMSL) to carry out inspection and evaluation activities of measuring equipment [23,24]. Each measuring equipment has its own set of rules and laws enforced by specialized government enforcement agencies. For example, smoke meters, axel scales, and speed traps are enforced by the Road and Transport Department. Meanwhile measuring instruments for trade purposes are under the jurisdiction of Ministry of Trade and Consumer Affairs. PA covers all trade and law enforcement activities to ensure that the instruments meet the requirements of legal and international standards. It also ensures that measurement accuracy, fair trade, and law enforcement function well in all environmental conditions and situations [25]. The instrument was tested and evaluated from various physical aspects in a variety of conditions, including software functionalities [26].

Illegal usage of software for NAWI is detected in the market during market surveillance once the pattern is approved [27]. This activity is carried out by regular inspections such as by a government-appointed third-party organization (as in the current situation). There are three elements to be checked on NAWI software during market surveillance:

1. Printed certificate/hard copy certificate (HC) information;
2. Certificate–software pairing is correct;
3. Legally relevant (LR) part of the software is intact.

If any of these stated elements are in doubt, the surveillance check is considered failed. The enforcement officer can take further steps such as sealing the whole system and seizing the whole measuring instrument system for further investigation.

The details of MySANI workflow during pattern approval and for market surveillance are depicted in Figures 3 and 4, respectively.

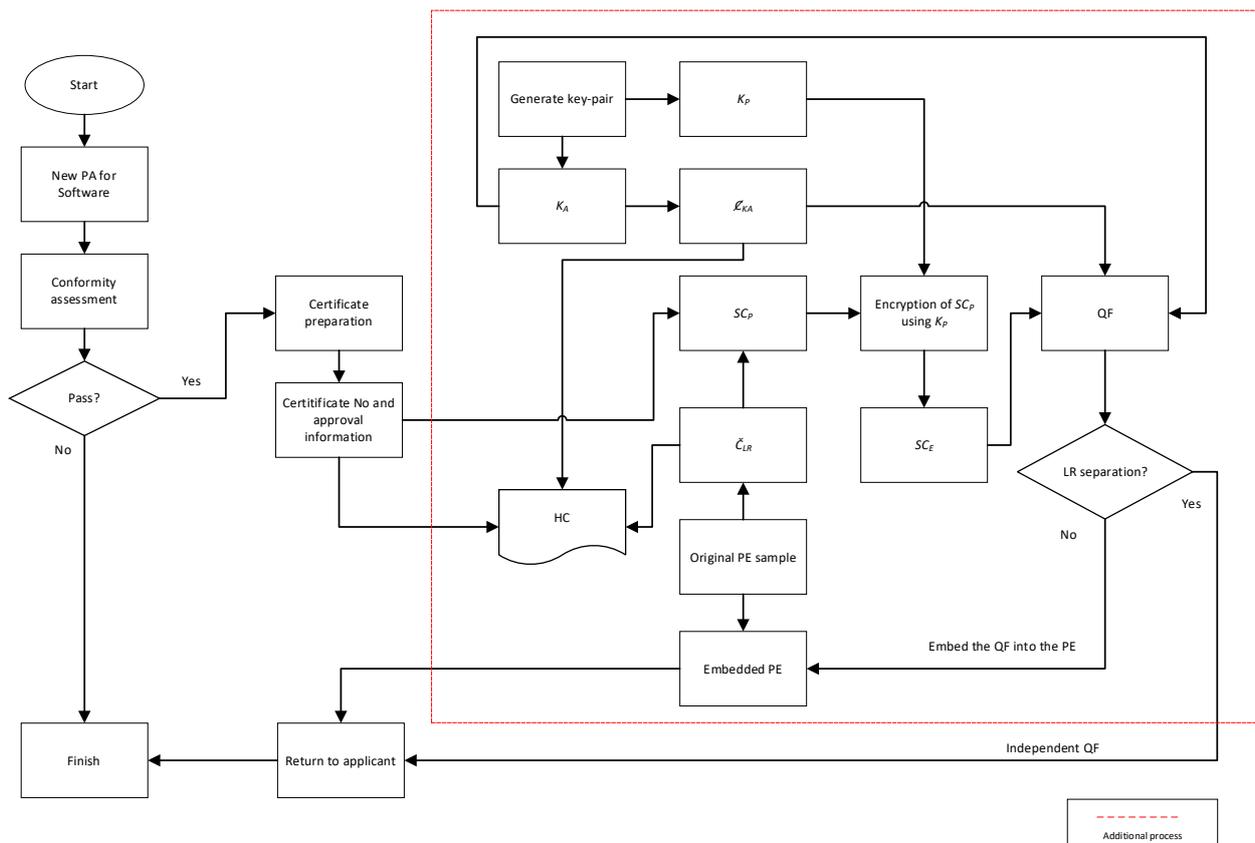


Figure 3. Proposed workflow during pattern approval.

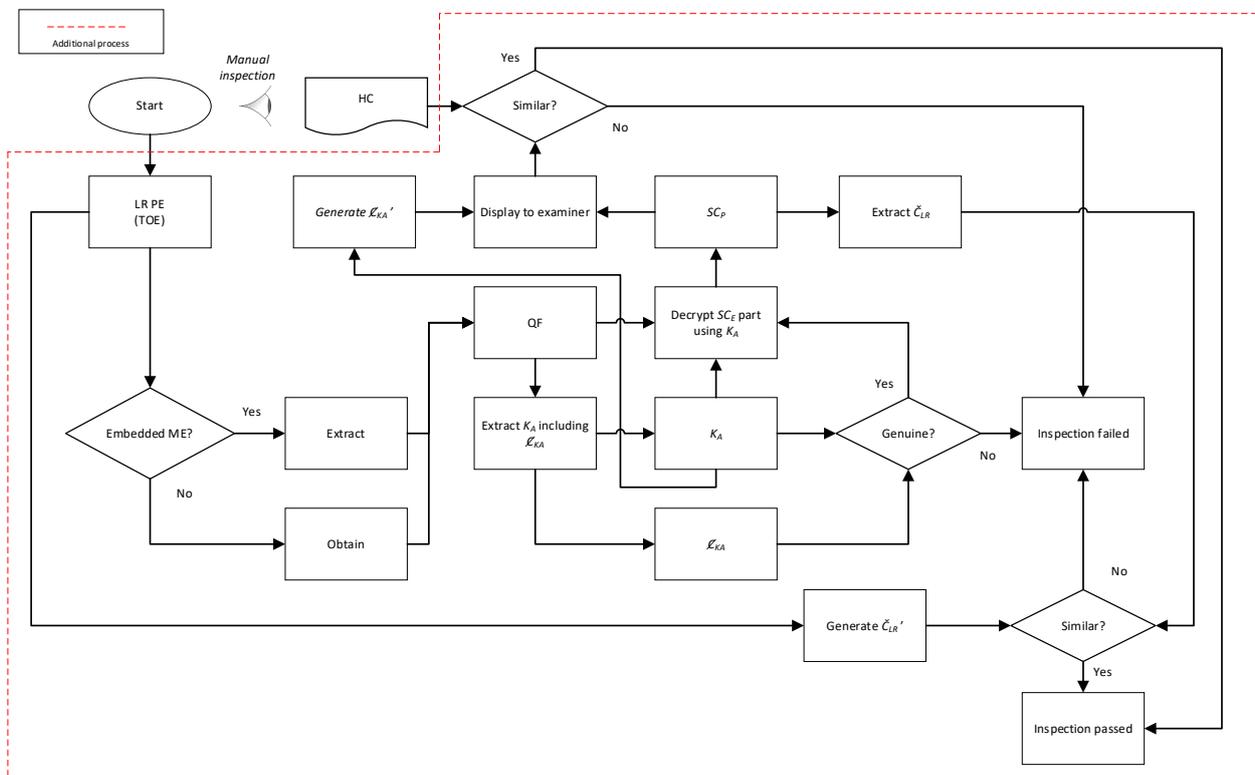


Figure 4. Proposed workflow during market surveillance.

### 4.1. Security Object

To achieve the aforementioned objectives, security objects (SOs) are implemented within the MySANI method. SOs consist of two parts: (1) soft-certificate plaintext ( $SC_P$ ); and (2) quasi file (QF) [5]. Figure 5 illustrates the components of the security object which are used in MySANI.

1. Transformed polynomial checksum for  $\mathcal{C}_{KA}$ ;
2. Plain checksum  $\check{C}_{LR}$  for LR module which are recorded inside the  $SC_P$ .

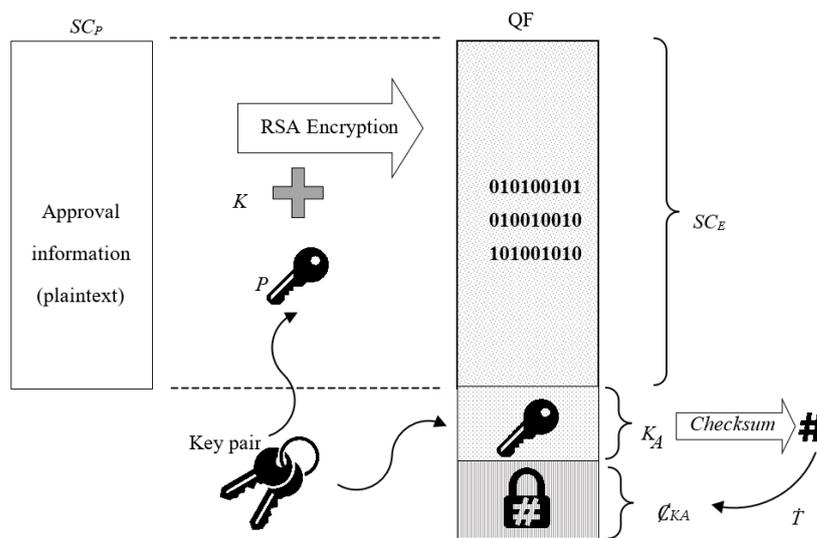


Figure 5. Security objects.

QF is designed as the security carrier to the  $SC_P$ . It protects the information contained from modification. Attacks on the  $SC_P$  merely be focus on the QF security features. QF gathers several sections of data into single file. It consists of three main parts: ciphered soft certificate  $SC_E$ , public key  $K_A$ , and the checksum of  $K_A$  with a hidden polynomial  $\mathcal{C}_{KA}$ . The  $SC_P$  is encrypted using the RSA private key  $K_P$  to become  $SC_E$ .  $K_A$  and  $K_P$  are generated by National Metrology Institute of Malaysia (NMIM) during certification process.  $K_P$  is not disclosed and is only known by NMIM. The  $QF = \{SC_E, K_A, \mathcal{C}_{KA}\}$  consists of three sub-components as described in Table 4.

Table 4. QF components.

Component	Description
$SC_E$	Ciphertext of $SC_P$
$K_A$	Public Key (RSA)
$\mathcal{C}_{KA}$	Checksum of $K_A$ which undergone polynomial transformation $\check{T}$

Let  $decr$  decryption and  $encl$  be the encryption function,  $K_A$  as public key and  $K_P$  as private key.  $SC_E$  is the ciphered text of  $SC_P$ . Therefore,  $SC_P = decr(SC_E, K_A) = decr(encl(SC_P, K_P), K_A)$  is the plaintext certificates which contain approval information that is important for inspection. It contains some of the plaintext information of static HC without security features  $\forall SC_P : SC_P \subseteq HC$  where it uses an XML format based on the proposed DCC format [28].

### 4.2. Process Workflow

During the PA certificate preparation stage, the certificate number and approval of related information along with the important information are used in the preparation of the  $SC_P$ . The information in this is important and sufficient for field inspection purposes.

This  $SC_P$  is then encrypted for the QF preparation process at the later stage. The checksum for the LR module  $\check{C}$  is generated for the purpose of printed certificate HC preparation. Some important information is used in the preparation of the  $SC_P$ . The information in this is important and sufficient for field inspection purposes. This  $SC_P$  is then encrypted for the QF preparation process at a later stage. Table 5 shows the information inside the plaintext certificate.

**Table 5.** Information inside plaintext certificate.

Field	Description
Certificate owner	Name of approval
Approval number	Approval number
Approval mode	Mode of approval (full/conditional)
Date of approval	Effective date of approval
Validity period	The duration of approval validity
Software name	The official software name
LR module	List of LR module

The MySANI workflow during pattern approval is shown in Figure 3. After a software passes the conformity assessment, while the certificate and approval information are being prepared for printed certificate HC, a pair of keys consisting of  $K_A$  and private key  $K_P$  are generated. The polynomial  $\mathcal{L}$  for  $K_A$  is also calculated and placed inside HC. The soft certificate plaintext  $SC_P$  contains approval information as well as the plain checksum for legally relevant module  $\check{C}$ . The  $SC_P$  is then encrypted by using  $K_P$  and becomes the ciphered certificate  $SC_E$ . The three elements ( $SC_E$ ,  $K_A$  and  $\mathcal{L}_{K_A}$ ) are then combined to form the QF. The QF is passed to the applicant or undergoes an embedding process to its main executable (ME) depending on the approval mode for distribution.

A special tool, MySANI inspection software (MySANI-IS), is developed to assist the enforcement officer to verify the security objects. The MySANI first tries to locate the QF; whether it is embedded within the main executable ME or as a standalone QF file. If it is embedded, then extraction procedures are performed to pull out the QF out of the ME. The QF is then extracted into three parts:  $SC_E$ ,  $K_A$ , and  $\mathcal{L}_{K_A}$ . The  $\mathcal{L}_{K_A}$  is used to determine the key's integrity and authenticity before it is used to extract the  $SC_E$  to form the  $SC_P$ . The approval information is then extracted from the  $SC_P$  and is displayed to the examiner for inspection. The plain  $\check{C}$  is used to determine whether the legally relevant part of the software is intact or not.

#### 4.3. Polynomial Transformation

The polynomial transformation  $\check{T} = \{st_1, st_2, \ell\}$  is an algorithm function to generate a checksum with hidden polynomial  $\mathcal{L}$  instead of a standard plain checksum  $\check{C}$ , for alignment with the LM requirements.  $\check{T}$  is realized by first calculating the checksum of two concatenated strings ( $st_1$  and  $st_2$ ) to become the  $\mathcal{L}'$  and then performing bitwise XOR off the resulting checksum with secret bytes  $\ell$ .

$\check{C}_{K_A}$  is the unknown plain checksum of  $K_A$  where  $\check{T}(K_A, S_S, \ell): \check{C}_{K_A} \rightarrow \mathcal{L}_{K_A}$ . The  $\mathcal{L}_{K_A}$  is in different form of  $\check{C}_{K_A}$  with equivalent strength of the checksum algorithm such that  $\check{C}_{K_A} \Leftrightarrow \mathcal{L}_{K_A}$ . Any modification that changes  $\check{C}_{K_A}$  also changes the  $\mathcal{L}_{K_A}$ .

$\mathcal{L}_{K_A'}$  checksum is calculated over the concatenated string of  $j = K_A \# S_S$ , where  $S_S$  is secret string and  $j$  contains both strings in the form of  $st_1$  and  $st_2$  where the  $st_1$  is the string of  $K_A$  while  $st_2$  is the string  $S_S$  such that  $K_A \# S_S = \{st_1 \ st_2: st_1 \in K_A, st_2 \in S_S\}$ . Then,  $\mathcal{L}_{K_A'} = crcf(j)$  and  $\mathcal{L}_{K_A} = \check{C}_{K_A'} \oplus \ell$ . The  $\mathcal{L}_{K_A'}$  can be retrieved back for integrity comparison purposes by  $\mathcal{L}_{K_A'} = \check{C}_{K_A} \oplus \ell$ .

Information within the  $SC_P$  is retrieved back and displayed to the inspector during market surveillance to confirm the correctness of the printed certificate information. Inspection is considered failed if the information displayed within  $SC_P$  is not the same as the information on the printed certificate.

The checksum of the legally relevant software is generated ( $\check{C}_{LR}'$ ) in real time for the purpose of comparison with the stored  $\check{C}_{LR}$  to check the integrity of the legally relevant module. Let the *extr* be the QF extraction function such that  $((\check{C}_{LR}' = \check{C}_{LR} \rightarrow \text{extr}(\text{QF})) \wedge (\neg \check{C}_{LR}' = \check{C}_{LR})) \rightarrow \text{FAIL!}$ .

### 5. Modelling Attack Scenario for MySANI

The attack tree is used to visualize the attack vectors that an attacker might perform. In order to model an attack tree, an attack scenario is required to be established in the first place. The attack scenario describes how the attacker might access the protected information and try to fake the information within the SO.

In this simulation, the attacker's main target is to modify the content of QF. The attacker himself is the owner of the software and indirectly also owns the QF (worst case scenario). The attacker has unrestricted access to the target artifact. The attacker is assumed to already possess MySANI-IS used by the inspector as the second target of attack.

The attacker realizes that the  $K_P$  is required to encrypt back the amended  $SC_P'$ . Since the key pair uses the RSA algorithm and the original  $K_P$  is not placed in the inspection's software, the only way is by generating the fake private key pair of  $K_P'$  and the fake public key of  $K_A'$ .  $K_P'$  is used to re-encrypt  $SC_P'$  while  $K_A'$  is placed back into QF replacing the original  $K_A$ .

In order for the status of the QF to remain valid during the field inspection, the attacker must find a way to perform reverse engineering on the software used by the inspector to obtain the  $\check{T}$  algorithm which protects the  $K_A$  as part of QF. Suppose the attacker succeeds in breaking the  $\check{T}$  algorithm: in that case, the attacker can recalculate the fake hidden polynomial  $\mathcal{L}'$  and once again, the attacker can reconstruct the fake QF ( $\text{QF}'$ ) by reassembling the above three parts, namely  $SC_P'$ ,  $K_A'$  and  $\mathcal{L}'$ . If this happens, the inspection is in valid status even if the QF's contents are changed. The inspector is not able to detect the irregularities of the QF.

The analysis is started by studying the attack vectors on QF. An attacker will not carry out an attack motivated by damaging or destroying the availability of the QF or altering any QF structure as this causes the status check to fail and the enforcement officer may immediately seize the entire system for further investigation.

#### 5.1. Attack Simulation

The root of the attack tree is  $v_t$ , when it is the ultimate goal for the attacker with intention to fake the QF with fabricated information and leave no trace of evidence during the market surveillance inspection. The tree consists of three main branches ( $v_1$ ,  $v_2$ , and  $v_3$ ). Attack vector  $v_1$  is about obtaining the QF, attack vector  $v_2$  is about modifying the content, while  $v_3$  is about hiding the trace. Figure 6 shows the attack tree designed for the simulation. The details of each of attack vector is described in Table 6.

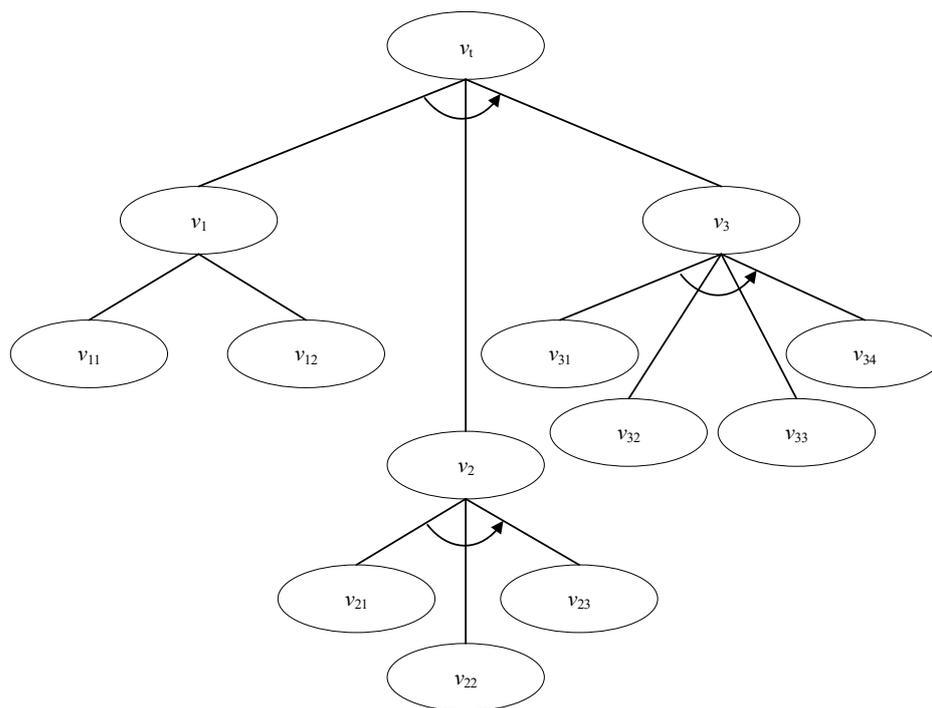


Figure 6. Attack tree analysis.

Table 6. Summary of attack vectors.

Field	Description
$v_t$	Attacker wants to fake the QF without any trace.
$v_1$	Attacker wants to obtain the QF.
$v_{11}$	The attacker extracts QF which is embedded inside the ME and tries to rebuild it into a single file.
$v_{12}$	Attacker tries to obtain the QF which is already in a single file form.
$v_2$	Attacker tries to amend the $SC_P$ information inside the QF.
$v_{21}$	Attacker tries to obtain the $K_A$ which is part of elements in QF.
$v_{22}$	Attacker decrypts the $SC_P$ using $K_A$ .
$v_{23}$	Attacker amends the information inside $SC_P$ into $SC_P'$ .
$v_3$	Attacker reorganize, reform the QF into fake QF'.
$v_{31}$	Attacker generates fake $K_P'$ .
$v_{32}$	Attacker encrypts the $SC_P'$ using $K_P'$ .
$v_{33}$	Attacker tries to obtain the $\tilde{T}$ algorithm and recalculate the transformed polynomial based on $SC_P'$ and $K_A'$ .
$v_{34}$	The attacker reforms and rebuilds the QF' again using fake components.

### 5.2. Security Object State Modelling

The formal validation method is performed by using a finite state transition system to validate the attack tree  $T$  design model. Before analysis can be performed, the attack tree must also be presented in the form of formal notation.

$Q = \{Q^-, Q^+, Qg, Qt\}$ : This variable specifies the state of QF. Where,  $Q^-$  indicates the state where the attacker does not have QF and it is embedded in the ME target of evaluation (TOE),  $Q^+$  indicates the state of the attacker does not have QF and is in the form of an individual file,  $Qg$  is the state where the attacker already has QF and is ready for attack, and  $Qt$  is the state where the attacker rearranges the QF with false elements.

$PKey = \{NaC, AcG, AcF\}$ : This variable specifies the state of public key.  $NaC$  indicates the attacker does not yet obtain the original public key  $K_A$ ,  $AcG$  indicates the original public key of the  $K_A$  is successfully obtained, and  $AcF$  the situation where the attacker generates and possesses a fake key pair.

$ScP = \{Up, Oe, Oc\}$ : This variable indicates the condition of plaintext certificate. It consists of three states: *Up* is the situation where the SCP plaintext soft certificate is not opened, *Oe* is the situation where the SCP plaintext soft certificate is successfully opened, and *Oc* is the situation where the  $ScP$  plaintext soft certificate is amended with fake information.

$TpRev = \{tt, ff\}$  is the variable which indicates the states of  $\tilde{T}$  algorithm cracks, whether the attacker successfully cracks the algorithm (*tt*) or not (*ff*).

### 5.2.1. State Transition Modelling

Let the  $Z$  system consist of two parties, namely the attacker and the QF. The system is modeled using a state variable, i.e., where its value determines the configuration possibilities for the system. In the initial settings, the attacker is assumed to not yet have a QF and intends to attack the QF by modifying the  $ScP$  without being able to be detected by the inspecting officer. To model the dynamic properties for the system, let the system be  $(z_{i-1}, z_i) \in \rightarrow$ , for every  $1 \leq i \leq 9$ . For the initial condition for the QF,  $q = (Q = Q^-) \vee (Q = Q^+)$  while the  $q' = (Q = Qg) \vee (Q = Qt)$  and  $z_0, z_1 \in \lambda(q)$ . For each state  $z_i \in \lambda(q')$  for  $3 \leq i \leq 9$ .

The path  $\rho$ , shows the scenario for the attacker to execute an attack from the beginning of the situation to the end of the goal in the  $\rho$  system. Table 7 summarized the state transition.

**Table 7.** State transition summary.

Variable	$z_0$	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$	$z_6$	$z_7$	$z_8$	$z_9$
<i>Q</i>	$Q^-$	$Q^+$	<b>Qg</b>	<i>Qg</i>	<i>Qg</i>	<i>Qg</i>	<i>Qg</i>	<i>Qg</i>	<i>Qg</i>	<b>Qt</b>
<i>PKKey</i>	<i>NaC</i>	<i>NaC</i>	<i>NaC</i>	<b>AcG</b>	<i>AcG</i>	<i>AcG</i>	<b>AcF</b>	<i>AcF</i>	<i>AcF</i>	<i>AcF</i>
<i>ScP</i>	<i>Uo</i>	<i>Uo</i>	<i>Uo</i>	<i>Uo</i>	<b>Op</b>	<b>Oe</b>	<i>Oe</i>	<b>Oc</b>	<i>Oc</i>	<i>Oc</i>
<i>TpRev</i>	<i>ff</i>	<b>tt</b>	<i>tt</i>							

For the transition system, only one variable changes at a time.  $z_0$  indicates a situation where the QF is not yet owned, is embedded, and needs to be extracted into an individual file, the attacker does not have any cryptographic keys, while the content of the soft certificate are in its original form. The state can be represented as  $Q = Q^-$  while the  $PKKey = NaC$ ,  $z_0, z_1 \in \lambda(q)$  and  $TpRev = ff$ .

The transition is as follows:

- $z_1$  is the same as  $z_0$  but QF is in the form of a separate file that needs to be identified by the attacker. The  $Q = Q^+$  state transition occurs;
- $z_2$  is the same as  $z_0$  but QF is successfully owned by the attacker. The  $Q = Qg$  state transition occurs;
- $z_3$  is the same as  $z_2$  but the attacker managed to obtain the  $K_A$  public key from the QF element. The  $PKKey = AcG$  state transition occurs;
- $z_4$  is the same as  $z_3$  but this time the attacker managed to open the  $ScP$  by decrypting the  $ScE$  using  $K_A$ . The  $ScP = Op$  state transition occurs;
- the  $z_5$  is the same as the  $z_4$  but the attacker generates a fake key pair that matches the original public key configuration. The following  $PKKey = AcF$  variable change occurs;
- $z_6$  is the same as  $z_5$  but the attacker amended  $ScP$  to  $ScP'$ . The  $ScP = Oe$  variable values are changed;
- $z_7$  is the same as  $z_6$  but the attacker encrypted  $ScP'$  using fake private key. The  $ScP = Oc$  values are changed;
- $z_8$  is the same as  $z_7$  except at this point the attacker successfully cracked the  $\tilde{T}$  algorithm,  $TpRev = tt$ ;
- And finally,  $z_9$  is the same last state as the  $z_8$  but the attacker reformed the QF with fake elements  $Q = Qt$ .

### 5.2.2. Attack Tree Formal Modelling

A similar attack is being modeled as attack tree  $T$  for the system  $Z$  as discussed, which has a goal in the form  $T = (\langle t, \gamma \rangle, \text{SAND})(T_1, T_2, T_3)$ . Where the formal representation of each sub-tree:

$$T_1 = (\langle t_1, \gamma_1 \rangle, \text{OR})(\langle t_{11}, \gamma_{11} \rangle, \langle t_{12}, \gamma_{12} \rangle);$$

$$T_2 = (\langle t_2, \gamma_2 \rangle, \text{SAND})(\langle t_{21}, \gamma_{21} \rangle, \langle t_{22}, \gamma_{22} \rangle, \langle t_{23}, \gamma_{23} \rangle);$$

$$T_3 = (\langle t_3, \gamma_3 \rangle, \text{SAND})(\langle t_{31}, \gamma_{31} \rangle, \langle t_{32}, \gamma_{32} \rangle, \langle t_{33}, \gamma_{33} \rangle, \langle t_{34}, \gamma_{34} \rangle).$$

Each sub-tree  $T_i$  has the atomic goal in the form of  $\langle t_i, \gamma_i \rangle$ . The same variables are applied to the state transition notation. Figure 7 shows the attack tree with formal notations.

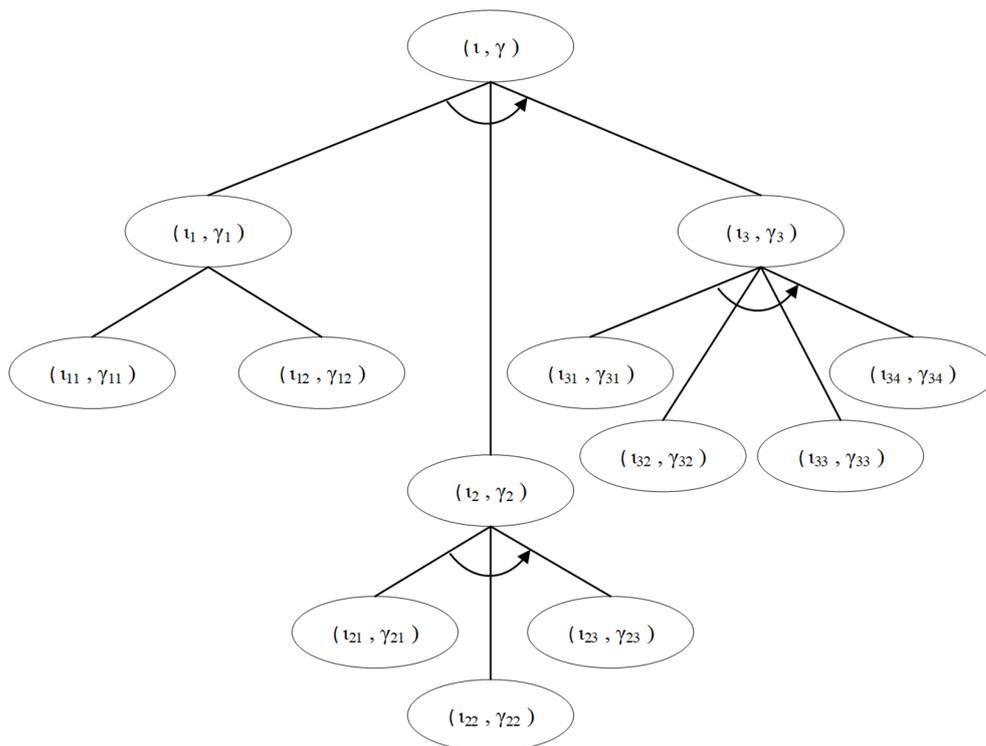


Figure 7. Attack tree with formal notations.

For the formal model the attack tree is as follows:

- The original situation is where the attacker does not yet have a QF, does not have a public key, is not able to access plaintext soft certificates, and did not successfully crack the  $\tilde{T}$  algorithm.

$$t := (Q = Q^-) \wedge (PKey = NaC) \wedge (ScP = Uo) \wedge (TpRev = ff);$$

- The ultimate goal for the attacker is to be able to amend the plaintext soft certificate, possess a fake key pair, re-encrypt using a fake private key, and put a fake public key back in the amended QF.

$$\gamma := (Q = Qt) \wedge (PKey = AcF) \wedge (ScP = Oe);$$

- Atomic target  $\langle t_1, \gamma_1 \rangle$ : In the original state, the attacker does not yet have QF and needs to identify a QF as the target first. The goal is to have the QF in the form of individual file. Assuming  $\top$  represents an empty configuration, therefore:

$$t_1 := \top \text{ and } \gamma_1 := (Q = Qg);$$

- Atomic goals  $\langle t_{11}, \gamma_{11} \rangle$ : QF is embedded in the ME and the attacker must first extract it into a single file. Therefore:

$$t_{11} := (Q = Q^-) \text{ and } \gamma_{11} = \gamma_1;$$

- Atomic goals  $\langle t_{12}, \gamma_{12} \rangle$ : QF is available as an individual file. The attacker needs to identify the location of the QF file. Therefore:

$$t_{12} := (Q = Q^+) \text{ and } \gamma_{12} := (Q = Qg);$$

- Atomic goal  $\langle t_2, \gamma_2 \rangle$ : The attacker aims to modify the  $SC_P$  in the QF. Therefore:

$$t_2 := (Q = Qg) \wedge (PKey = AcG) \wedge (ScP = Uo) \wedge (TpRev = ff) \text{ and } \gamma_2 := \gamma;$$

- Atomic goal  $\langle t_{21}, \gamma_{21} \rangle$ : Attacker obtains  $K_A$ . Therefore:

$$t_{21} := (Q = Qg) \wedge (PKey = NaC) \text{ and } \gamma_{21} := (PKey = AcG);$$

- Atomic goal  $\langle t_{22}, \gamma_{22} \rangle$ : Attacker decrypts  $SC_E$  using  $K_A$ . Therefore:

$$t_{22} := \gamma_{21} \text{ and } \gamma_{22} := (ScP = Op);$$

- Atomic goal  $\langle t_{23}, \gamma_{23} \rangle$ : Attacker modifies the contents of the  $SC_P$ . Therefore:

$$t_{23} := \gamma_{22} \text{ and } \gamma_{23} := (ScP = Oe);$$

- Atomic goal  $\langle t_3, \gamma_3 \rangle$ : The attacker aims to reconstruct the QF using modified plaintext data and fake keys. Therefore:

$$t_3 := (PKey = AcF) \wedge (ScP = Oe) \wedge (TpRev = ff) \text{ and } \gamma_3 := \gamma;$$

- Atomic goal  $\langle t_{31}, \gamma_{31} \rangle$ : Attacker generates fake  $K_P'$ . Therefore:

$$t_{31} := \top \text{ and } \gamma_{31} := (PKey = AcF);$$

- Atomic goals  $\langle t_{32}, \gamma_{32} \rangle$ : The attacker encrypts the  $SC_P'$  by using fake key  $K_P'$ . Therefore:

$$t_{32} := (PKey = AcF) \wedge (ScP = Oe) \text{ and } \gamma_{32} := (ScP = Oc);$$

- Atomic goals  $\langle t_{33}, \gamma_{33} \rangle$ : Attack polynomial transformations based on  $K_P'$  and  $SC_P$ . Therefore:

$$t_{33} := (TpRev = ff) \text{ and } \gamma_{33} := (TpRev = tt);$$

- Finally, the atomic goal  $\langle t_{34}, \gamma_{34} \rangle$ : The attacker reconstructs all elements into a false QF. Therefore:  $t_{34} := (PKey = AcF) \wedge (ScP = Oe) \wedge (TpRev = tt)$  and  $\gamma_{34} := \gamma_3$ .

### 5.3. Formal Analysis and Discussion

$T_1$  is defined as  $\llbracket \langle t_1, \gamma_1 \rangle \rrbracket^S$  while for the sub-trees, are  $\llbracket \langle t_{11}, \gamma_{11} \rangle \rrbracket^S$  and  $\llbracket \langle t_{12}, \gamma_{12} \rangle \rrbracket^S$ , respectively. Therefore, because of  $\llbracket \text{OR}(\langle t_{11}, \gamma_{11} \rangle, \langle t_{12}, \gamma_{12} \rangle) \rrbracket^S = \llbracket \langle t_{11}, \gamma_{11} \rangle \rrbracket^S \cup \llbracket \langle t_{12}, \gamma_{12} \rangle \rrbracket^S$  while  $\llbracket \langle t_1, \gamma_1 \rangle \rrbracket^S \cap \llbracket \text{OR}(\langle t_{11}, \gamma_{11} \rangle, \langle t_{12}, \gamma_{12} \rangle) \rrbracket^S \neq \emptyset$ , then the tree is considered to have the meet property.

Since the  $\llbracket \text{OR}(\langle t_{11}, \gamma_{11} \rangle, \langle t_{12}, \gamma_{12} \rangle) \rrbracket^S \supseteq \llbracket \langle t_1, \gamma_1 \rangle \rrbracket^S$ , thus the  $T_1$  is also has the overmatch property.

For  $T_2$ , it is defined as  $\llbracket \langle t_2, \gamma_2 \rangle \rrbracket^S$  and all the sub-trees can be refined as  $\llbracket \text{SAND}(\langle t_{21}, \gamma_{21} \rangle, \langle t_{22}, \gamma_{22} \rangle, \langle t_{23}, \gamma_{23} \rangle) \rrbracket^S$ ; therefore, the sub-trees are equivalent to  $\llbracket \langle t_{21}, \gamma_{21} \rangle \rrbracket^S \cdot \llbracket \langle t_{22}, \gamma_{22} \rangle \rrbracket^S \cdot \llbracket \langle t_{23}, \gamma_{23} \rangle \rrbracket^S$ . From a quick analysis, the  $\llbracket \text{SAND}(\langle t_{21}, \gamma_{21} \rangle, \langle t_{22}, \gamma_{22} \rangle, \langle t_{23}, \gamma_{23} \rangle) \rrbracket^S \neq \emptyset$  and thus,  $T_2$  is considered to have the meet property. More-

over,  $\llbracket \text{SAND}(\langle t_{21}, \gamma_{21} \rangle, \langle t_{22}, \gamma_{22} \rangle, \langle t_{23}, \gamma_{23} \rangle) \rrbracket^S = \llbracket \langle t_2, \gamma_2 \rangle \rrbracket^S$ ; therefore,  $T_2$  also has the match property.

For the last branch  $T_3$ ,  $\llbracket \langle t_3, \gamma_3 \rangle \rrbracket^S$  where the sub-trees can be refined as  $\llbracket \text{SAND}(\langle t_{31}, \gamma_{31} \rangle, \langle t_{32}, \gamma_{32} \rangle, \langle t_{33}, \gamma_{33} \rangle, \langle t_{34}, \gamma_{34} \rangle) \rrbracket^S$  where it is equivalent to  $\llbracket \langle t_{31}, \gamma_{31} \rangle \rrbracket^S \cdot \llbracket \langle t_{32}, \gamma_{32} \rangle \rrbracket^S \cdot \llbracket \langle t_{33}, \gamma_{33} \rangle \rrbracket^S \cdot \llbracket \langle t_{34}, \gamma_{34} \rangle \rrbracket^S$ . By removing non-elementary paths  $\llbracket \langle t_3, \gamma_3 \rangle \rrbracket^S \cap \llbracket \text{SAND}(\langle t_{31}, \gamma_{31} \rangle, \langle t_{32}, \gamma_{32} \rangle, \langle t_{33}, \gamma_{33} \rangle, \langle t_{34}, \gamma_{34} \rangle) \rrbracket^S \neq \emptyset$ . Thus, the  $T_3$  has the meet property and since  $\llbracket \langle t_3, \gamma_3 \rangle \rrbracket^S = \llbracket \text{SAND}(\langle t_{31}, \gamma_{31} \rangle, \langle t_{32}, \gamma_{32} \rangle, \langle t_{33}, \gamma_{33} \rangle, \langle t_{34}, \gamma_{34} \rangle) \rrbracket^S$ ; therefore,  $T_3$  also has the match property.

For the final path semantics of goal expression  $T$ ,  $\llbracket \langle t, \gamma \rangle \rrbracket^S$ , can be refined as  $\llbracket \text{SAND}(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \langle t_3, \gamma_3 \rangle) \rrbracket^S$  and is equivalent to  $\llbracket \langle t_1, \gamma_1 \rangle \rrbracket^S \cdot \llbracket \langle t_2, \gamma_2 \rangle \rrbracket^S \cdot \llbracket \langle t_3, \gamma_3 \rangle \rrbracket^S$ . By removing non-elementary paths, similar to the previous sub-trees,  $\llbracket \langle t, \gamma \rangle \rrbracket^S \cap \llbracket \text{SAND}(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \langle t_3, \gamma_3 \rangle) \rrbracket^S \neq \emptyset$ , thus the  $T$  has the meet property and  $\llbracket \langle t, \gamma \rangle \rrbracket^S = \llbracket \text{SAND}(\langle t_1, \gamma_1 \rangle, \langle t_2, \gamma_2 \rangle, \langle t_3, \gamma_3 \rangle) \rrbracket^S$ ; therefore,  $T$  has the match property.

Considering the main tree  $T = (\langle t, \gamma \rangle, \text{SAND})(T_1, T_2, T_3)$ , and considering all the following sub-branches, the  $(\langle t_1, \gamma_1 \rangle, \text{OR})(T_{11}, T_{12}) \neq \emptyset$ ,  $(\langle t_2, \gamma_2 \rangle, \text{SAND})(T_{21}, T_{22}, T_{23}) \neq \emptyset$  and  $(\langle t_3, \gamma_3 \rangle, \text{SAND})(T_{31}, T_{32}, T_{33}, T_{34}) \neq \emptyset$  as well as the main tree  $(\langle t, \gamma \rangle, \text{SAND})(T_1, T_2, T_3) \neq \emptyset$ , the tree  $T$  is considered to have the Global Admissible property.

The design of attack tree  $T$  is found to be consistent and correct with respect to the system  $Z$  in its finite transition system domain. Each of the sub-trees have match property except the sub-tree  $T_1$ , which has the overmatch property. This however does not affect the overall security features of QF since all the sub-branches of  $T$  relate to  $OP = \text{SAND}$  thus complementing each other. Thus, the attack tree  $T$  modelling can be used to further analyze the security features of QF.

## 6. Conclusions

This paper discussed MySANI, the enhanced method in regulating the software used for NAWI within the LM framework in Malaysia. This method is realized by introducing QF, which is used as a secondary information source of LR information of the TOE and secured via public cryptography. The security design is validated by using attack tree modelling where the attack tree is described using formal notation and the correctness validation is achieved via comparison against the finite state transition system. As part of security analysis in LM, this paper also demonstrated the possibility of the formalization design of attack tree before it could be integrated with the AtPT framework analysis. This enables the attack tree to be verified prior to further security risk analysis evaluation. The issue of scalability is not studied in this paper due to the nature of the LR part of NAWI, which is usually very small. However, in the future, if the security objects in this paper are to be implemented for other types of measuring instruments, the scalability issues of the formal method approach might be considered.

**Author Contributions:** Conceptualization, M.A.I. and F.Q.; Funding acquisition, Z.S.; Methodology, N.Z.; Supervision, N.M.; Visualization, Z.S.; Writing—original draft, M.A.I. and N.Z.; Writing—review and editing, F.Q., N.M. and M.U.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper is supported by the Universiti Kebangsaan Malaysia Grant Number: KKP/2020/UKM-UKM/4/3.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would also like to thank the respected editor and reviewer for their support.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. O'Brien, M.T.; Schuh, J.C.L.; Wancket, L.M.; Cramer, S.D.; Funk, K.A.; Jackson, N.D.; Kannan, K.; Keane, K.; Nyska, A.; Rousselle, S.D.; et al. Scientific and Regulatory Policy Committee Points to Consider for Medical Device Implant Site Evaluation in Nonclinical Studies. *Toxicol. Pathol.* **2022**, *50*, 512–530. [[CrossRef](#)] [[PubMed](#)]
2. Doe, J.; Van de Wetering, R.; Honyenuga, B.; Versendaal, J. Eco-system oriented instrument for measuring firm technology adoption. In Proceedings of the 19th International Conference on Electronic Business, Newcastle Upon Tyne, UK, 8–12 December 2019.
3. Ghazvini, A.; Shukur, Z. Review of information security guidelines for awareness training program in healthcare industry. In Proceedings of the 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), Langkawi, Malaysia, 25–27 November 2017; pp. 1–6.
4. Schwemer, S.F. Article 17 at the Intersection of EU Copyright Law and Platform Regulation. *Nord. Intellect. Prop. Law Rev.* **2020**, *1*, 400–435.
5. Said, I.O.; Shukur, Z.; Bin Ibrahim, M.A. A certification criteria for software of measuring instruments based on Malaysian environment. In Proceedings of the 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), Langkawi, Malaysia, 25–27 November 2017; pp. 1–5.
6. Sahlabadi, M.; Muniyandi, R.C.; Shukur, Z.; Qamar, F. Lightweight Software Architecture Evaluation for Industry: A Comprehensive Review. *Sensors* **2022**, *22*, 1252. [[CrossRef](#)] [[PubMed](#)]
7. Verdon, D.; McGraw, G. Risk analysis in software design. *IEEE Secur. Priv.* **2004**, *2*, 79–84. [[CrossRef](#)]
8. Talabis, M.; Martin, J. *Information Security Risk Assessment Toolkit: Practical Assessments through Data Collection and Data Analysis*; Newnes: Boston, MA, USA, 2012.
9. Esche, M.; Thiel, F. P7.4—Incorporating a measure for attacker motivation into software risk assessment for measuring instruments in legal metrology. In Proceedings of the 18th GMA/ITG-Fachtagung Sensoren und Messsysteme 2016, Nuremberg, Germany, 10–11 May 2016; Volume 1, pp. 735–742. Available online: <https://www.ama-science.org/proceedings/details/2436> (accessed on 10 January 2023).
10. Esche, M.; Toro, F.; Thiel, F. Representation of attacker motivation in software risk assessment using attack probability trees. In Proceedings of the 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), Prague, Czech Republic, 3–6 September 2017; pp. 763–771.
11. Meng, B.; Larraz, D.; Siu, K.; Moitra, A.; Interrante, J.; Smith, W.; Paul, S.; Prince, D.; Herencia-Zapana, H.; Arif, M.; et al. VERDICT: A Language and Framework for Engineering Cyber Resilient and Safe System. *Systems* **2021**, *9*, 18. [[CrossRef](#)]
12. Audinot, M.; Pinchinat, S.; Kordy, B. Guided design of attack trees: A system-based approach. In Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF), Oxford, UK, 9–12 July 2018; pp. 61–75.
13. Pinchinat, S.; Acher, M.; Vojtisek, D. ATSyRa: An integrated environment for synthesizing attack trees. In Proceedings of the International Workshop on Graphical Models for Security, Verona, Italy, 13 July 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 97–101.
14. Pieters, W.; Hadziomanovic, D.; Lenin, A.; Montoya, L.; Willemson, J. TRESPASS: Plug-and-play attacker profiles for security risk analysis. *IEEE Secur. Priv. Poster Abstr.* **2014**, *1*, 1–2.
15. Kordy, B.; Kordy, P.; Mauw, S.; Schweitzer, P. ADTool: Security analysis with attack–defense trees. In Proceedings of the International conference on quantitative evaluation of systems, Buenos Aires, Argentina, 27–30 August 2013; pp. 173–176.
16. Kordy, B.; Mauw, S.; Melissen, M.; Schweitzer, P. Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In Proceedings of the International Conference on Decision and Game Theory for Security, Berlin, Germany, 22–23 November 2010; pp. 245–256.
17. Mauw, S.; Oostdijk, M. Foundations of attack trees. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Republic of Korea, 1–2 December 2005; pp. 186–198.
18. Scala, N.M.; Goethals, P.L.; Dehlinger, J.; Mezgebe, Y.; Jilcha, B.; Bloomquist, I. Evaluating mail-based security for electoral processes using attack trees. *Risk Anal.* **2022**, *42*, 2327–2343. [[CrossRef](#)]
19. Audinot, M.; Pinchinat, S.; Kordy, B. Is my attack tree correct? In Proceedings of the European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 83–102.
20. Schiele, N.D.; Gadyatskaya, O. A Novel Approach for Attack Tree to Attack Graph Transformation. In Proceedings of the International Conference on Risks and Security of Internet and Systems, Sousse, Tunisia, 7–9 December 2022; pp. 74–90.
21. Yu, L.; Chen, K.; Chang, Y.; Chen, A.; Yin, Q.; Zhang, H. A New Correlation Model of IoT Attack Based on Attack Tree. In Proceedings of the 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCOM/CyberSciTech), Calgary, AB, Canada, 25–28 October 2021; pp. 930–935.
22. Manaf, M.R.A.; Nawawi, A.M.; Tauhid, N.M.; Othman, H.; Rahman, M.R.A.; Yusoff, H.M.; Safian, N.; Ng, P.Y.; Manaf, Z.A.; Kadir, N.B.A.; et al. Prevalence of metabolic syndrome and its associated risk factors among staffs in a Malaysian public university. *Sci. Rep.* **2021**, *11*, 1–11. [[CrossRef](#)] [[PubMed](#)]
23. Ibrahim, M.A.; Shukur, Z.; Zainal, N.; Marzuki, N.; Zakaria, O.; Yusof, M.M. Legalizing Software For Measuring Instruments: A Proposed Plan For Malaysian Case Study. *Asia-Pac. J. Inf. Technol. Multimed.* **2018**, *9*, 99–109. [[CrossRef](#)]
24. Ibrahim, M.A.; Marzuki, N.; Shukur, Z.; Zainal, N. A Proposed Plan in Legalising Software for Measuring Instruments in Malaysia. In Proceedings of the 2018 Cyber Resilience Conference (CRC), Putrajaya, Malaysia, 13–15 November 2018; pp. 1–4.

25. Berk, R.A. Predictive Policing, and Risk Assessment for Law Enforcement. *Annu. Rev. Criminol.* **2021**, *4*, 37. [[CrossRef](#)]
26. Ahmed, F.; Straub, J. Initial Work on the Development of a Hardware-Based Gradient Descent Trained Expert System. *Systems* **2022**, *10*, 160. [[CrossRef](#)]
27. Wang, J.; Han, Z.; Peng, C.; Wu, D. Preliminary study of parameter optimizations toward a lab-designed acoustic-based volume measuring system for weights. *Measurement* **2022**, *197*, 111244. [[CrossRef](#)]
28. Brown, C.; Elo, T.; Hovhannisyanyan, K.; Hutzschenreuter, D.; Kuosmanen, P.; Maennel, O.; Mustapaa, T.; Nikander, P.; Wiedenhofer, T. Infrastructure for Digital Calibration Certificates. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 3–5 June 2020; pp. 485–489.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.