In partial fulfilment of the requirements for the award of the degree of
Doctor of Philosophy in Computer Science

Models, Algorithms and Methodology for Design of Microcontroller-Based Physical Security Systems Protected from Cyber-Physical Attacks

Thesis submitted by

Dmitry Levshun

under the guidance of

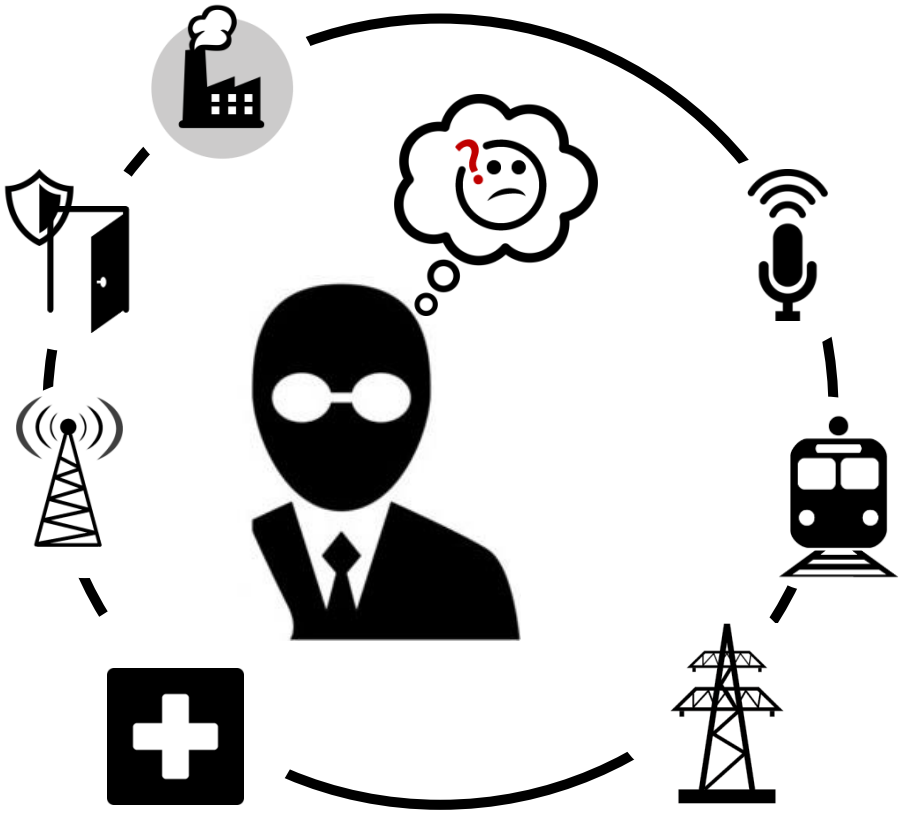
Prof. **Igor Kotenko**, ITMO University, Russia
Yannick Chevalier, University Paul Sabatier, France

Federal State Autonomous Educational Institution of Higher Education "National Research University ITMO" (**ITMO University**)
University of Toulouse III – Paul Sabatier (**University Paul Sabatier**)

December 15, 2021

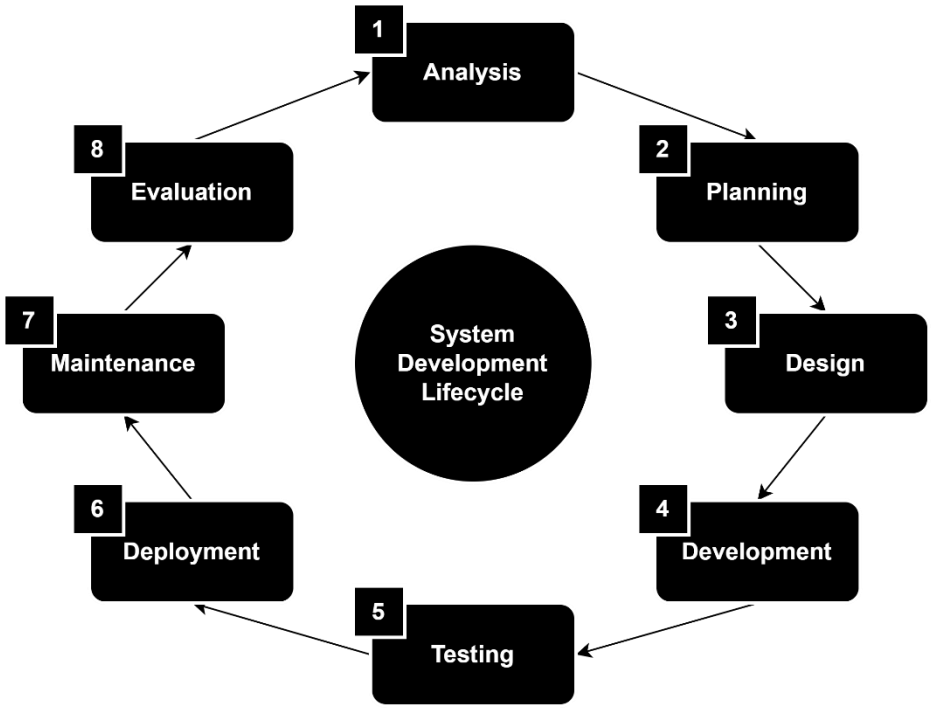
Synopsis

- **Microcontroller-based systems** are an integral part of any sphere of our vital activity. The importance of ensuring their **security is critical**.
- The **consequences** of **failure** of such systems include both **financial** and **reputational** damage as well as a threat to human **life** and **health**.
- According to the **SonicWall** report, microcontroller-based devices malware attacks jumped **215.7%** to **32.7 million** in 2018 up from 10.3 million in 2017.
- According to the **Palo Alto Networks** 2020 Unit 42 Threat Report "**98%** of all device traffic is **unencrypted**, exposing **personal** and **confidential** data on the network".



Synopsis

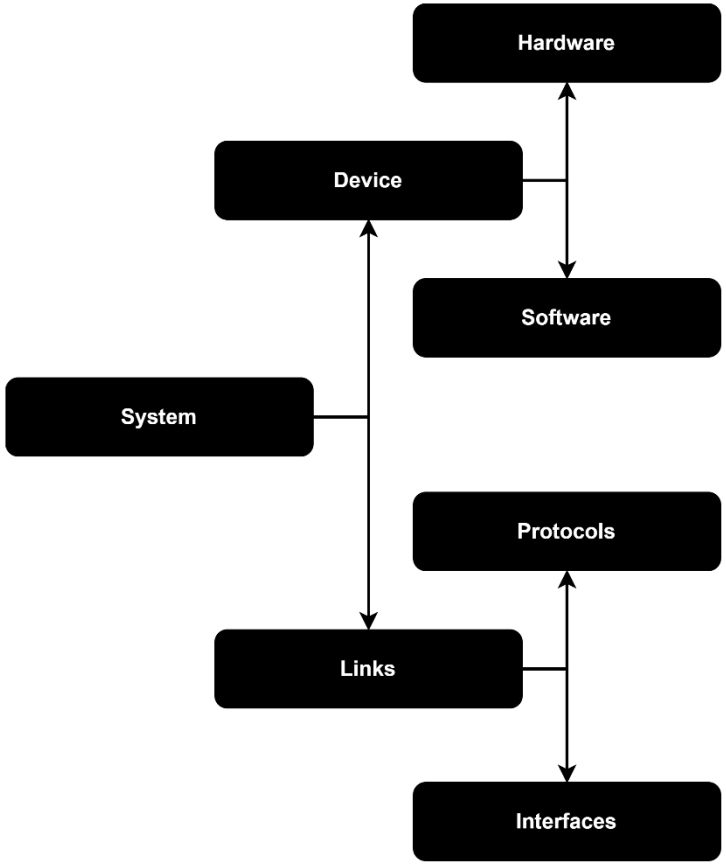
- **Approaches** to ensure the **information security** of microcontroller-based systems are associated with individual **stages** of development **lifecycle**.
- The **approaches** that are discussed in this work are used at the design stage of lifecycle and associated with **Security by Design**.
- **Security by Design** is an approach to software and hardware development that aims to reduce the number of **vulnerabilities** and enhance the **system's protection** against attacks.
- The main idea of the approach is in taking into account **security features** as a **design criterion** of products. Such features are represented **together** with non-security ones as **requirements**.



Synopsis

- The main **disadvantages** of **scientific** solutions:
  - focus on only **one aspect** of the security;
  - strong **relations** between software and hardware of devices are not considered;
  - devices are designed without taking into account the **system** they will work in;
  - secure communication is provided only between devices, while their communications with **external systems** is not considered.

- The main **disadvantages** of **commercial** solutions:
  - strong **restrictions** on the possible platforms and architectures of devices to be used;
  - security is ensured only on the level of **cloud services** and communications between the gateway device and the **cloud**;
  - there is **no trade-off** between the security level and resources expended on it.



Synopsis

- **Input data:**
  - general tasks of the designed system (TK);
  - parameters of the attacker (ac, kn, rs);
  - work mode (wm, manual/automatic selection).

$$D_I = (TK, ac, kn, rs, wm)$$

- **The objective function** of the methodology is aimed at maximization of the number of **parameters** that are analyzed during the design process.

number of levels of the system, the security of which can be ensured

$$O_F: |LEVELS \times ATTACKS| \rightarrow max$$

number of classes of attack actions against which the system can be protected

- **Output data:**
  - methodology work log (wl);
  - abstract model of the system (am);
  - list of components and controllers that are possible for selection (sl);
  - detailed model of the system (dm);
  - methodology output log (ol).

$$D_O = (wl, am, sl, dm, ol)$$

Synopsis

- **Time consumption:**
  - time required for the design process of the **abstract** model should be less than **1 sec**[1];
  - time required for the design process of the **detailed** model should be less than **4 sec**[1].
  - $P_T^{ACC} = 0.99$.

$$P_T(TIME_N \leq TIME^{ACC}) \geq P_T^{ACC}$$

- **Resource consumption:**
  - the number of resources required for the **design process** should be less than **25%**[2] of the computer resources;
  - $P_R^{ACC} = 0.99$.

$$P_R(RES_N \leq RES^{ACC}) \geq P_R^{ACC}$$

[1] Such time frames were selected for the design of a secure system with **3 types** of devices, while **each type** of devices contains **not less** than **5**, **10** and **15** elements with sub-elements accordingly.

[2] For the experimental evaluation, the computer with Windows 10 x64 operating system, Intel Core i7-4790 CPU 3.60GHz (8 cores) processor, 2 TB HDD and 32 GB RAM was used.

Synopsis

- Thus, in this work it is **required** to **develop** the design **methodology** for microcontroller-based **physical security systems** that based on D_I provides D_O, while $P_T \geq 0.99$, $P_R \geq 0.99$ and value of O_F exceeds values of analogs.

- **Functional requirements** to the design methodology:
  - building an **abstract** representation of the designed system;
  - finding a **trade-off** between the resources spent and the security;
  - **no restrictions** on platforms and architectures of the devices;
  - the **extensibility** of the design process;
  - taking into account the **physical layer** of the designed systems.

- **Limitations** of the design methodology:
  - works only with **ready-made** components and controllers, without taking into account elements of **electronic circuit**;
  - not generating **source code** of the system software and firmware;
  - parameters of the device **case**, its **cooling** and resistance to various **weather conditions** are not taken into account.

Synopsis

- **Scientific task:**
  - Development of the **model-methodological apparatus** for the design of microcontroller-based physical security systems protected from cyber-physical attacks.

- **Goal of the study:**
  - Enhancing the **protection** of microcontroller-based physical security systems from cyber-physical attacks by increasing the number of **analyzed parameters** during their design process.

- **Objectives:**
  - Analysis of the **main security issues** of microcontroller-based systems, place and role of the design approaches in ensuring the security of such systems as well as the main features of such systems.
  - Development of **models of the elements** of microcontroller-based physical security systems, including security ones.
  - Development of the **hierarchical model of the attacker** that allows distinguishing between attackers based on their types of access, knowledge and resources.

Synopsis

Objectives:

- Development of the **model of attack actions** that allows checking the possibility of implementation of different classes of attacks based on the attacker's parameters (subject) and system elements (object).
- Development of the **model of microcontroller-based physical security system**, which is an extendable set-based hierarchical relational unification of models of system elements, attacker and attack actions.
- Development of the **set of algorithms** for the design of the extendable set-based hierarchical relational model of microcontroller-based physical security systems.
- Development of the **methodology for design** of microcontroller-based physical security systems, combining the set of algorithms and extendable set-based hierarchical relational model into a single automated approach with minimal operator involvement.
- Development of the **software implementation** of the design methodology for microcontroller-based physical security systems, its **experimental evaluation**.

Synopsis

- In this work based it was decided to divide possible elements of microcontroller-based **systems** into **components**, **controllers** and **devices** that are **communicating** with each other.

- **A component** is something that can be connected to a **controller** and either send signals to it or react to signals from it. **Components** can communicate only with **controllers** they are connected to. **Components** can be represented as sensors, transmitters, motors, batteries, etc.

- **A controller** is something that can be programmed to work with **components** and other **controllers**. **Controllers** can communicate with **components** and other **controllers** that are connected to them. **Controllers** can be represented as microcontrollers and single-board computers.

- **A device** is something that represents a system of **controllers** and **components** that are communicating with each other inside it. **Devices** can communicate only with other **devices**. **Devices** can be represented as system servers, hubs, robots, stations, drones, terminals, etc.

$$controller \leftrightarrow component$$
$$controller \leftrightarrow controller$$
$$device \leftrightarrow device$$
$$system \leftrightarrow system$$

Synopsis

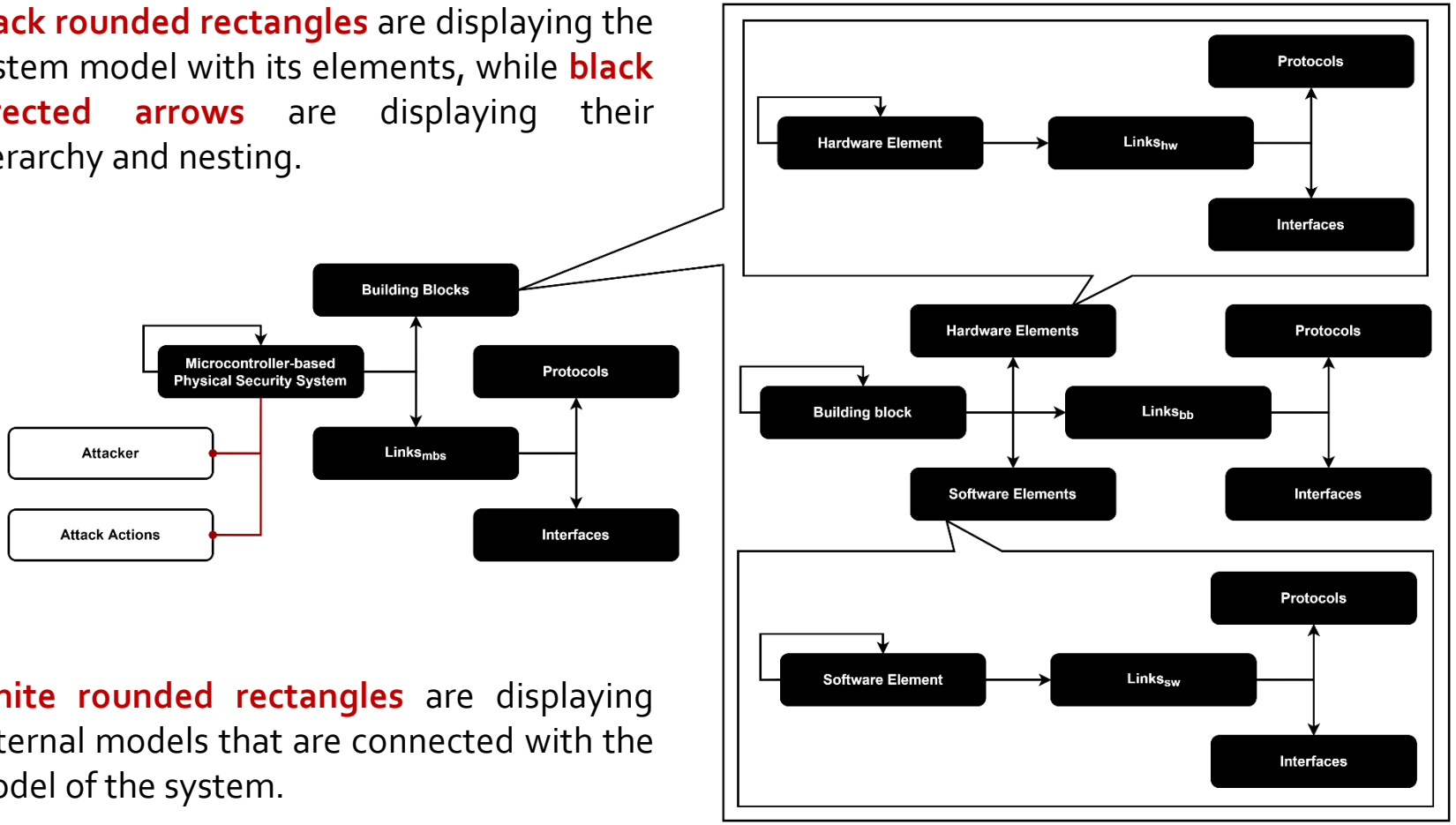
The main findings:

1. The **extendable set-based hierarchical relational model** of microcontroller-based physical security systems protected from cyber-physical attacks and its elements.
2. The **set of algorithms** for the design of extendable set-based hierarchical relational models of microcontroller-based physical security systems protected from cyber-physical attacks.
3. The **methodology** for the design of microcontroller-based physical security systems protected from cyber-physical attacks, that combines the set of algorithms and the extendable set-based hierarchical relational model into a **single automated approach** with minimal operator involvement.
4. The **software implementation** of the methodology for the design of microcontroller-based physical security systems protected from cyber-physical attacks, that validates its correctness based on the design of a **system of mobile robots** for perimeter monitoring.

1. Extendable set-based hierarchical relational model of microcontroller-based physical security systems

Extendable set-based hierarchical relational model

- **Black rounded rectangles** are displaying the system model with its elements, while **black directed arrows** are displaying their hierarchy and nesting.



- **White rounded rectangles** are displaying external models that are connected with the model of the system.

Extendable set-based hierarchical relational model

Microcontroller-based system

$$mbs = (MBS', BB, L_{mbs}, a, AA, p_{mbs}),$$

where MBS' — set of microcontroller-based sub-systems of mbs; BB — set of building blocks of mbs; L_{mbs} — set of links between BB and MBS' of mbs; a — attacker against mbs; AA — set of attack actions on mbs; p_{mbs} — properties of mbs.

Examples of mbs: access control, fire alarm, security alarm, perimeter monitoring, etc. The situation when mbs contains subsystems related to integrated physical security systems.

Building block

$$bb = (BB', HW, SW, L_{bb}, p_{bb}),$$

where BB' — set of building sub-blocks of bb; HW — set of hardware elements of bb; SW — set of software elements of bb; L_{bb} — links between elements of bb; p_{bb} — properties of bb.

Examples of bb: any device, controller or its combination with components — Raspberry Pi single-board computer, micro-SD card with pre-installed operating system, ESP8266 or Iskra JS microcontroller as well as even server, hub, robot, station, drone, etc.

Extendable set-based hierarchical relational model

Hardware element

$$hw = (HW', L_{hw}, p_{hw}),$$

where HW' — set of hardware sub-elements hw' of hw; L_{hw} — links between elements of hw; p_{hw} — properties of hw.

Examples of hw: any component — sensors, receivers, transmitters, readers, motors, batteries, etc. The situation when hw consists of multiple hw' — a motor shield with two collector motors that can be used for two-wheel mobile robots.

Software element

$$sw = (SW', L_{sw}, p_{sw}),$$

where SW' — set of software sub-elements sw' of sw; L_{sw} — links between elements of sw; p_{sw} — properties of sw.

Examples of sw: any algorithm, library, firmware, database, application or configuration can be used. The situation when sw consists of multiple sw' — a firmware of the controller that can be used as the brain of two-wheel robots. Such a firmware often contains library imports for work with components as well as algorithms for the navigation, communication, data processing, etc.

Extendable set-based hierarchical relational model

Links

$$L = (R, I, E, p_L),$$

where R — set of protocols that are used in L; I — set of interfaces that are used in L; E — set of communication parties of L; p_L — properties of L.

Examples of L: L_{mbs} — links between devices of the system; L_{bb} — links between controllers of devices; L_{hw} — links between controllers and components; L_{sw} — links between software elements.

Properties

$$p = (FR, NL, PF, PR),$$

where FR — set of functional requirements (functionality that satisfaction is necessary for the element to work); NL — set of non-functional limitations (limitation that satisfaction is necessary for the element to work); PF — set of provided functionalities; PR — set of provided resources.

Examples of FR: power source, protocol, interface, bootloader, library, compiler, driver, etc.
NL: space for placement, suitable environment, voltage, current, size, etc.
PF: access control, perimeter monitoring, navigation, obstacles detection, etc.
PR: data storage, computing resources, environment for launching applications, etc.

Extendable set-based hierarchical relational model

Attacker

$$a = (ac, kn, rs),$$

where ac — type of access a has to mbs; kn — type of knowledge a has about mbs; rs — type of resources available to a to compromise mbs.

	Attacker's types of access
1	No access to the system
2	Access to the system through global networks
3	Access to the system through local networks
4	Physical access to the system
5	Full access to the system

	Attacker's types of knowledge
1	General knowledge about the system from publicly available sources
2	Knowledge about parameters of the system
3	Knowledge about means of protection of the system
4	Knowledge about software and hardware of the system

	Attacker's types of resources
1	Widely-spread software tools and known vulnerabilities
2	Specialized software tools and previously non-used vulnerabilities
3	Possibility to investigate the system

The **structure** of attacker's access, knowledge and resources types is **hierarchical**:

- a_1 with $ac_{a_1} = 3$ is **able** to perform any attack action which is possible for a_2 with $ac_{a_2} = 2$, if $kn_{a_1} \geq kn_{a_2}$ and $rs_{a_1} \geq rs_{a_2}$.
- a_3 with $ac_{a_3} = 3$ is **able** to perform any attack action which is possible for a_1, if $kn_{a_3} \geq kn_{a_1}$ and $rs_{a_3} \geq rs_{a_1}$.
- a_4 with $ac_{a_4} = 3$, $kn_{a_4} = 2$, $rs_{a_4} = 2$ is **not able** to perform all attack actions that are possible for a_5 with $ac_{a_5} = 2$, $kn_{a_5} = 3$, $rs_{a_5} = 3$.

Extendable set-based hierarchical relational model

Attack actions

$$aa = (cl, oj, sj) \mid aa \in AA,$$

where cl — class of aa; oj — object of aa, helps to link aa with the target element(s) of mbs; sj — subject of aa, helps to link aa with a that is capable enough for its successful realization.

Classes of attack actions

$$cl = \{cn, cr, dv, st\},$$

where cn — aa on the level of components and their communications with controllers; cr — aa on the level of controllers and their communications with other controllers; dv — aa on the level of devices and their communications with other devices; st — aa on the level of the system and its communications with other systems.

Examples of attack actions on cn level

$$cn = \{gie, bcd, rpt, rmt\},$$

where gie — generation of incorrect component events; bcd — bypassing component detection algorithms; rpt — replacement of the component; rmt — removement of the component.

Extendable set-based hierarchical relational model

Examples of attack actions on cr level

$$cr = \{rfw, rbl, mup, imv\},$$

where rfw — replacement of the controller's firmware; rbl — reinstallation of the controller's bootloader; mup — malfunction of the controller's update system; imw — interception, modification or termination of wired communications.

Examples of attack actions on dv level

$$dv = \{vau, cad, iec, iws\},$$

where vau — violation of the authentication system; cad — cryptographic analysis of transmitted data; iec — increased energy consumption; iws — interception, modification or termination of wireless communications.

Examples of attack actions on st level

$$st = \{soc, pwr, web, dbd\},$$

where soc — social engineering; pwr — power failure; web — disruption of web services; dbd — disruption of database services.

Extendable set-based hierarchical relational model

			cl													
			ec				*mc*				*dv*				*st*			
			gie	*bsd*	*rpt*	*rmt*	*rfw*	*rbl*	*mup*	*imv*	*vau*	*cad*	*iec*	*iws*	*soc*	*pwr*	*web*	*dbd*
a	*ac*	1																
		2																
		3																
		4																
		5																
	kn	1																
		2																
		3																
		4																
	rs	1																
		2																
		3																

Extendable set-based hierarchical relational model

			cl													
			ec				*mc*				*dv*				*st*			
			gie	*bsd*	*rpt*	*rmt*	*rfw*	*rbl*	*mup*	*imv*	*vau*	*cad*	*iec*	*iws*	*soc*	*pwr*	*web*	*dbd*
a	*ac*	1													+			
		2													+		+	+
		3	+	+					+		+	+	+	+	+		+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	kn	1				+				+				+	+	+		
		2	+		+	+				+			+	+	+	+	+	+
		3	+	+	+	+			+	+	+	+	+	+	+	+	+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	rs	1			+	+				+			+		+	+	+	+
		2		+	+	+			+	+	+	+	+	+	+	+	+	+
		3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Extendable set-based hierarchical relational model

			cl														
			ec				*mc*				*dv*				*st*			
			gie	*bsd*	*rpt*	*rmt*	*rfw*	*rbl*	*mup*	*imv*	*vau*	*cad*	*iec*	*iws*	*soc*	*pwr*	*web*	*dbd*
a	*ac*	1													+			
		2													+		+	+
		3	+	+					+		+	+	+	+	+		+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	kn	1				+				+				+	+	+		
		2	+		+	+				+			+	+	+	+	+	+
		3	+	+	+	+			+	+	+	+	+	+	+	+	+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	rs	1			+	+				+			+		+	+	+	+
		2		+	+	+			+	+	+	+	+	+	+	+	+	+
		3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Extendable set-based hierarchical relational model

			cl													
			ec				mc				dv				st			
			gie	bsd	rpt	rmt	rfw	rbl	mup	imv	vau	cad	iec	iws	soc	pwr	web	dbd
a	ac	1													+			
		2													+		+	+
		3	+	+					+		+	+	+	+	+		+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	kn	1				+				+				+	+	+		
		2	+		+	+				+			+	+	+	+	+	+
		3	+	+	+	+			+	+	+	+	+	+	+	+	+	+
		4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	rs	1			+	+				+			+		+	+	+	+
		2		+	+	+			+	+	+	+	+	+	+	+	+	+
		3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Extendable set-based hierarchical relational model

			Security elements
cl	cn	gie	anomaly detection algorithm, hidden placement of sensors
		bcd	events correlation algorithm, hidden placement of sensors
		rpt	vandal-proof device case, hardware authentication
		rmt	vandal-proof device case
	cr	rfw	vandal-proof device case, firmware encryption
		rbl	vandal-proof device case, bootloader encryption
		mup	vandal-proof device case, removement of physical update interface
		imw	vandal-proof device case, encryption, authentication
	dv	vau	strong login credentials, password policy, brute-force protection
		cad	strong encryption algorithms, secure key distribution mechanism
		iec	behavior-based anomaly detection, devices isolation/limitation
		iws	strong encryption algorithm on access point, strong login credentials, public key pair-based authentication
	st	soc	training of operators and users, security policy
		pwr	uninterruptible power supplies, backup power supply
		web	firewall, update mechanism, backup mechanism, logging mechanism
		dbd	input validation, strict access policy, strong login credentials, separate database users for different operations

			Non-security elements
cl	cn	gie	sensors and receivers that react on the environment
		bcd	sensors that monitor environment
		rpt	any component
		rmt	any component
	cr	rfw	any controller with rewritable firmware
		rbl	any controller with rewritable bootloader
		mup	any controller with update system
		imw	controller ↔ controller, controller ↔ component
	dv	vau	device ↔ device, where authentication is used
		cad	device ↔ device, where encryption is used
		iec	devices with sleep mode/wireless interfaces
		iws	device ↔ device
	st	soc	any system with operators or/and users
		pwr	any system that relies on power grid
		web	any system with web-services
		dbd	any system with database

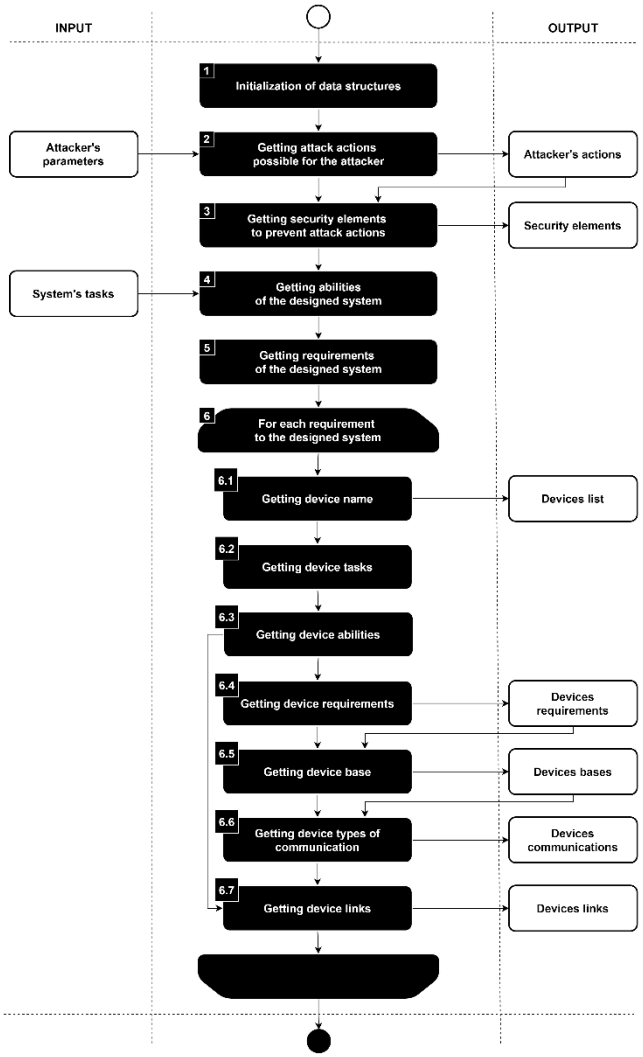
Extendable set-based hierarchical relational model

- Unlike existing solutions, the extendable set-based hierarchical relational **model represents** a microcontroller-based physical security **systems instead of** representing individual microcontroller-based **devices**.
- Developed model is **modular**, **extensible** and **hierarchical**, has a strong **focus on security** of the resulting solution as well as considers security elements as an **integral part** of the designed system.
- The **extension** is possible by introduction of the **new levels of abstraction**. The **modularity** of the solution provides the **possibility to change** only its **individual parts** without the need to change the model completely.
- The **hierarchical nature** of the model allows **decomposition** from the whole **system** into individual **elements**, and **composition** from individual **elements** to the **system** as a whole.

2. Set of algorithms for the design of microcontroller-based physical security systems

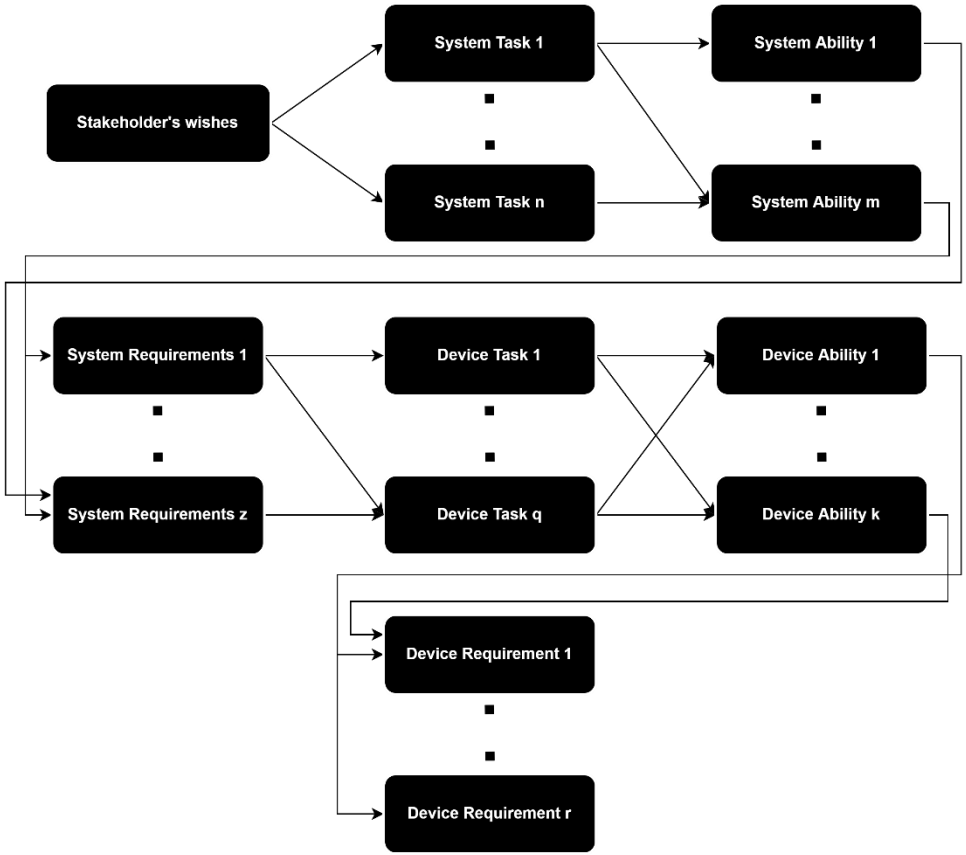
Formation of requirements for the system

- **Input data**:
  - attacker's parameters (ac, kn, rs);
  - system's tasks (id, description).
- **Output data** :
  - attacker's actions (id, name, description);
  - security elements (id);
  - devices list (name);
  - devices requirements (device key, id, description);
  - devices communications (device key, id, name);
  - devices links (device key , id, type, description);
  - devices bases (device key , id, description).
- The algorithm is **automated**, the operator is required for the **translation of wishes** of the stakeholder into the attacker's **parameters** and general **tasks** of the system.
- The work process of the algorithm contains **6 main stages**, while the last stage is divided into **7 sub-stages**.



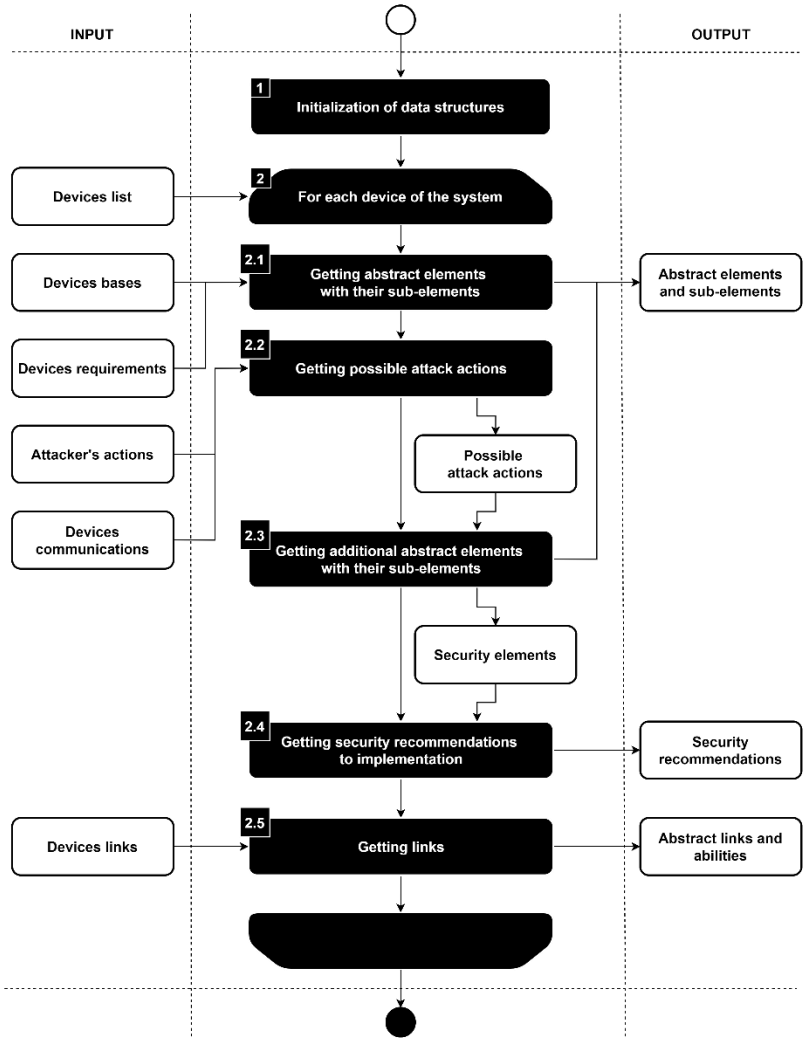
Formation of requirements for the system

- The **novelty** of the algorithm:
  - List of **devices**, their **communications** and **requirements** are retrieved based on system **tasks**.
  - List of **attack actions** that are **possible** for the **attacker** is retrieved based on the type of access, knowledge and resources the attacker has.
  - The algorithm takes into account **dependencies** between stakeholder's **wishes** and system **tasks**, system **tasks** and system **abilities**, system **abilities** and system **requirements**, system **requirements** and devices **tasks**, devices **tasks** and devices **abilities**, devices **abilities** and devices **requirements**.



Formation of the system component composition

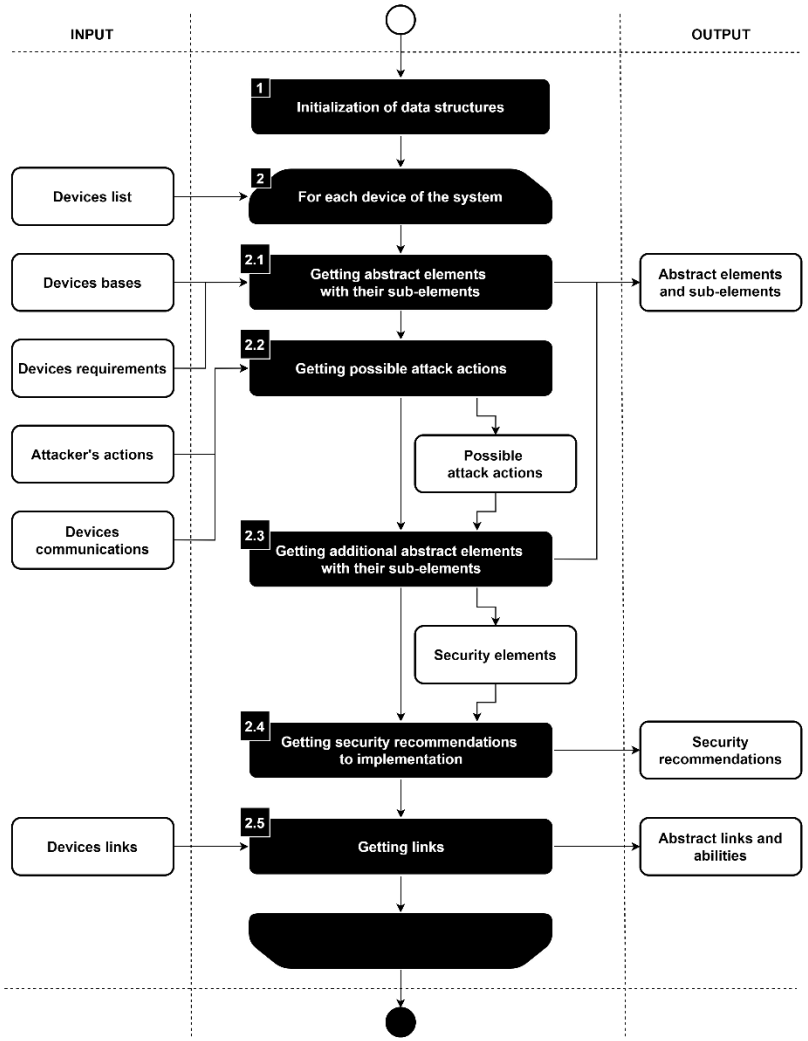
- **Input data**:
  - devices list (name);
  - devices bases (device key, id, description);
  - devices requirements (device key, id, description);
  - attacker's actions (id, name, description);
  - devices communications (device key, id, name);
  - devices links (device key, id, type, description);
- **Output data** :
  - abstract elements and sub-elements (device key, element key, id, description);
  - security recommendations (system key, device key, id, description);
  - abstract links and abilities (device key, id, type, description, related abilities).
- Algorithm is **automatic**, the operator is **not required**.
- Algorithm contains **2 main stages**, while the last stage is divided into **5 sub-stages**.



Formation of the system component composition

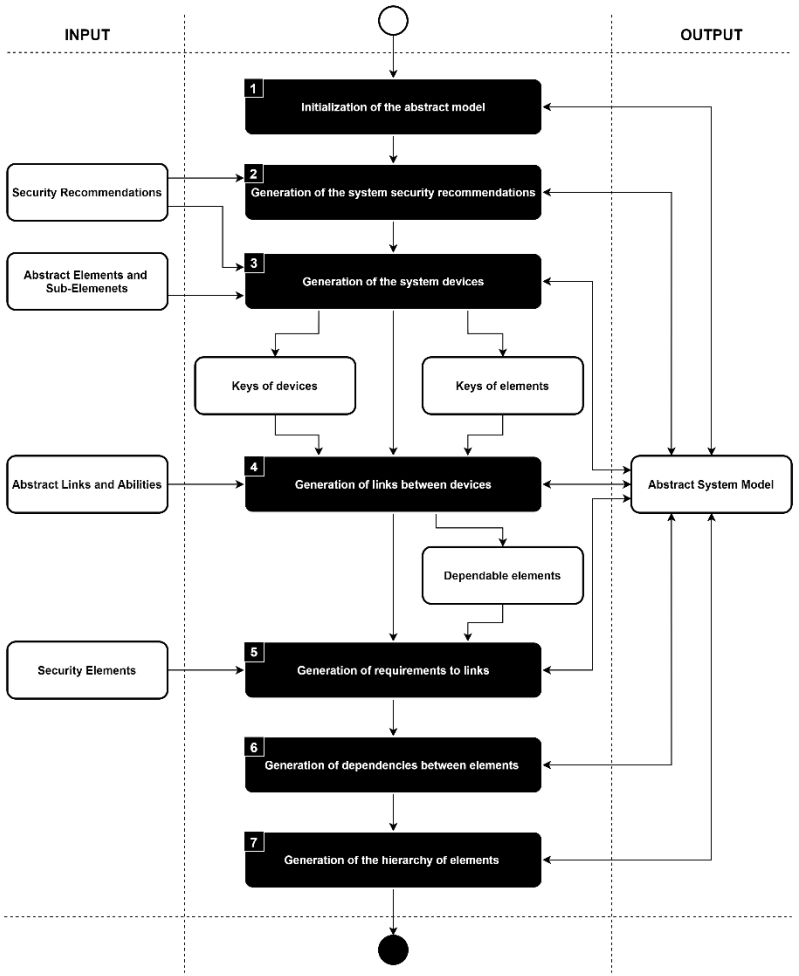
- The **novelty** of the algorithm:
  - **Elements** are extracted **recursively** for each device in accordance with the provided device **base**, its **requirements** and already extracted **elements**.
  - Required **security** elements are represented as abstract **elements**, **sub-elements**, and **recommendations** for the system and its devices implementation.

- This algorithm can be **useful** to an **expert** in the design of secure systems, but its full **potential** is revealed only when interacting with other algorithms within the **framework** of the design **methodology**.



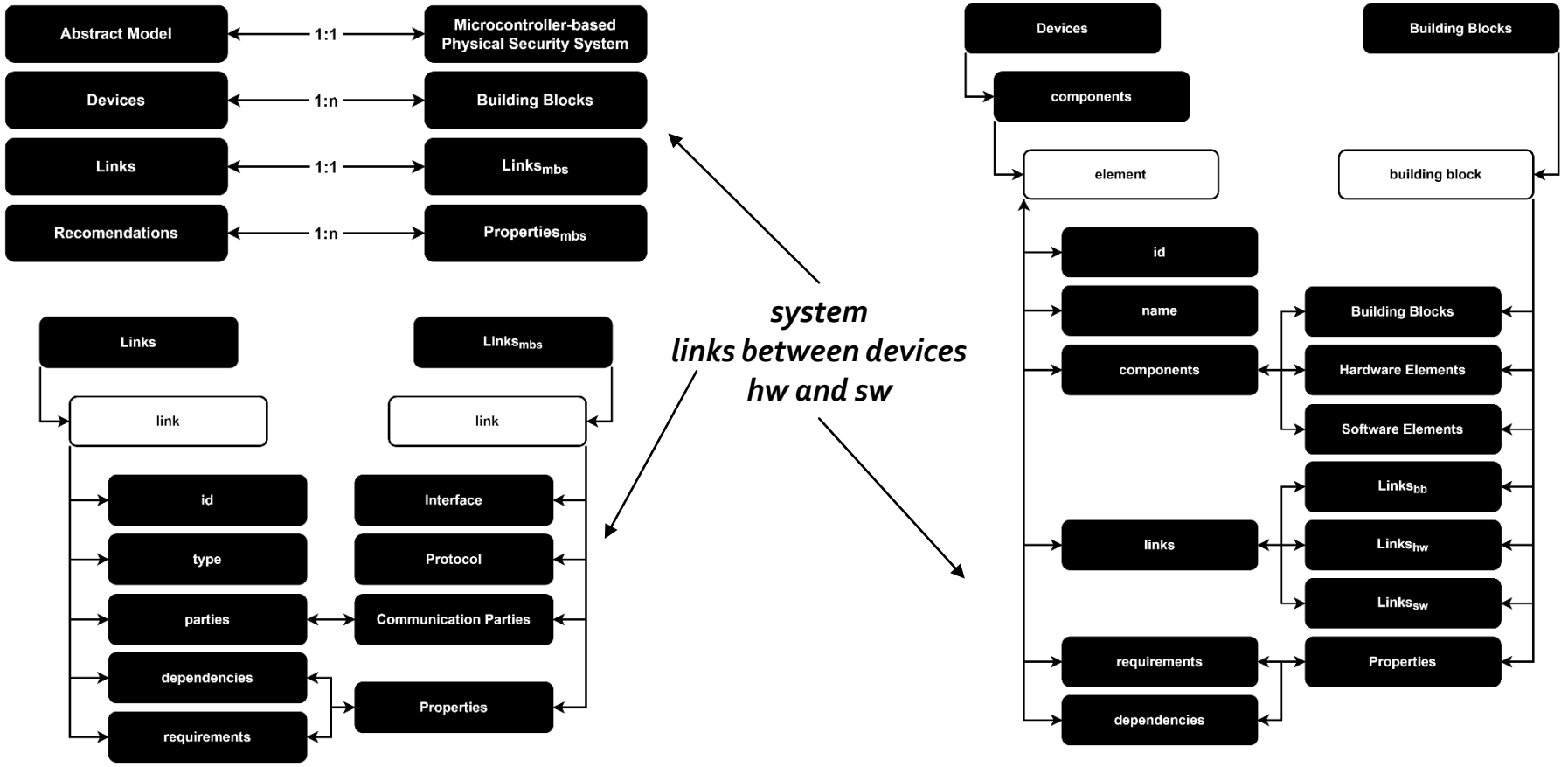
Design of the abstract model of the system

- **Input data**:
  - security recommendations to the system and devices implementations, abstract elements of devices and their sub-elements, abstract links between devices, devices abilities, security elements of devices.
- **Output data**:
  - abstract system model in JSON-based format.
- **Abstract model**:
  - devices: key, id, name, components and recommendations;
  - recommendations: key, id, name (description);
  - links: key, id, type, parties, dependencies, requirements.
- Each **element** of "components" field has its unique key, id, components (**sub-elements**), links, requirements and dependencies.



Design of the abstract model of the system

- The **abstract system model** is a **mapping** of the extendable set-based hierarchical relational model

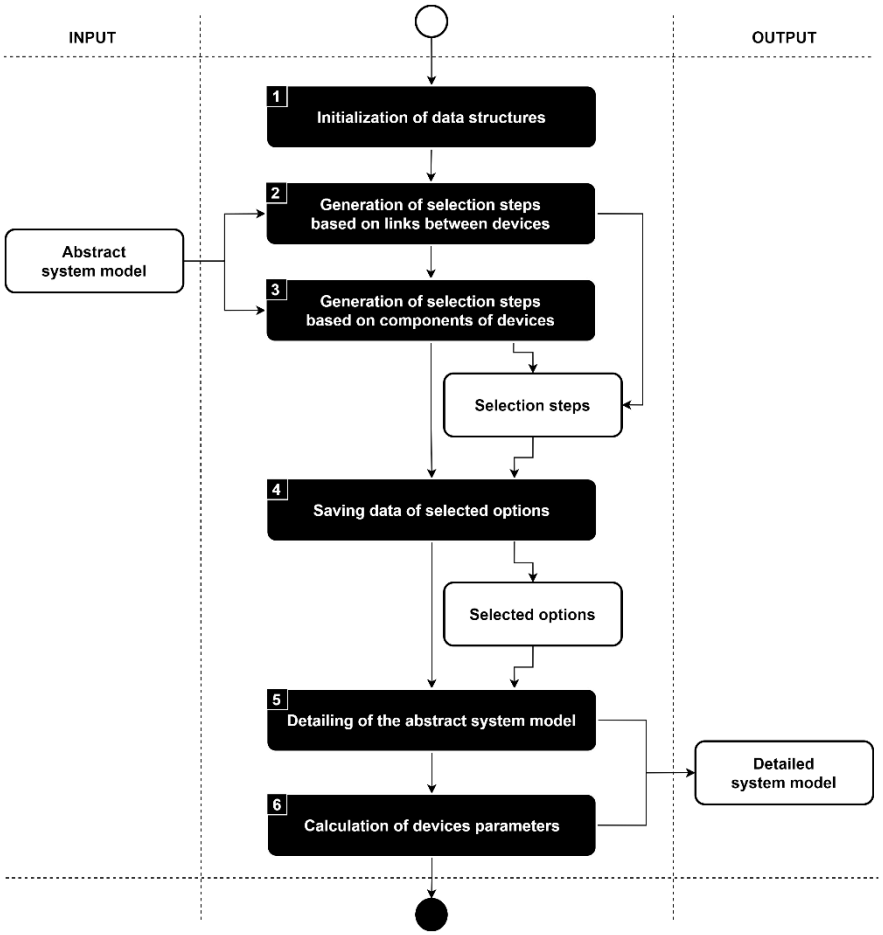


Design of the abstract model of the system

- The **novelty** of the algorithm:
  - The **complex dependencies** between the elements of microcontroller-based systems are taken into account, namely, their **hierarchy**, **nesting**, **communications**, **conflicts** and **requirements**.
  - There are **no limitations** to specific platforms and architectures.
  - The **abstract nature** of the algorithm allows one to reduce the number of parameters to be searched, thereby increasing its **work speed**.
- The algorithm is **automatic**, the operator is **not required**.
- This algorithm can be **useful** to an **expert** in the design of secure systems, but its full **potential** is revealed only when interacting with other algorithms within the **framework** of the design **methodology**.
- In such a situation the **abstract model** would be **detailed** by replacing abstract elements with their **concrete implementations** (taking into account **requirements**, mutual **dependencies** and possible **conflicts**).

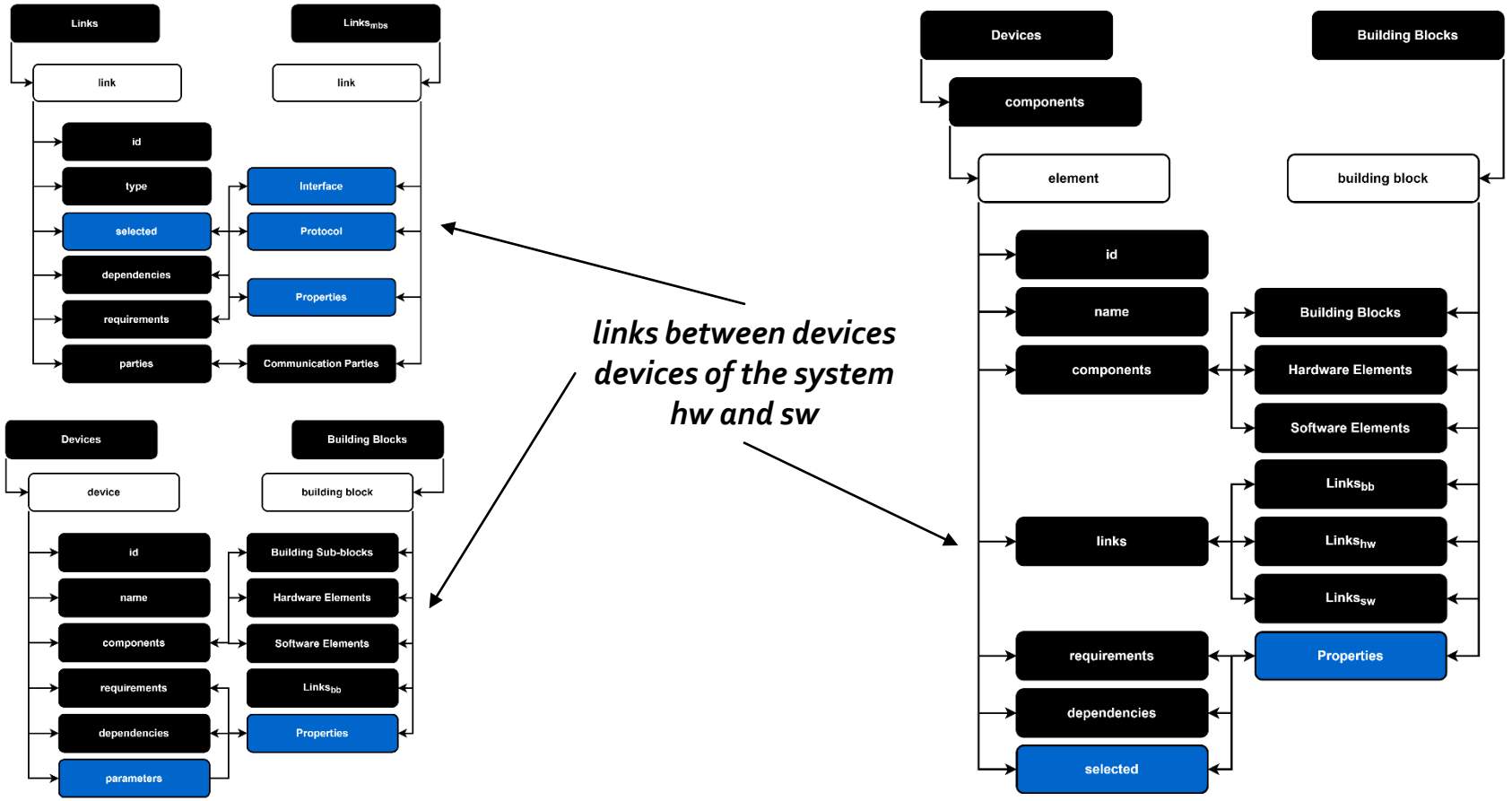
Design of the detailed model of the system

- **Input data**:
  - abstract system model.
- **Output data**:
  - detailed system model in JSON-based format.
- **Detailed model** = **abstract**, but with additions:
  - each detailed **element** is extended with the **selected field**: data about selected element, including id, name and parameters of its implementation;
  - each **device** is extended with the **parameters field**: data about parameters of the designed device, including price, energy consumption, voltage, current, length, width, height, free memory and battery life.
  - each **link between devices** is extended with the **selected field**: data about the selected links between devices, including id, name, interface, protocol and parameters.



Design of the detailed model of the system

- The **detailed system model** is a **mapping** of the extendable set-based hierarchical relational model



Design of the detailed model of the system

- The **novelty** of the algorithm:
  - A **step-by-step detailing process** of the abstract system representation is formed in accordance with the **hierarchy** and **mutual dependencies** of the system elements.
  - The **parameters** of **devices** are calculated based on the parameters of their **elements**, while the parameters of the **system** are calculated based on the parameters of its **devices**.
  - The **abstract system model** is not **replaced**, but **expanded** and **complemented**.
- The work process of the algorithm is **automated**, involvement of the operator is **possible** at the stage of **selection of** the concrete **implementations** of elements. Alternatively, the algorithm can select concrete implementations on its own.
- This algorithm can be **useful** to an **expert** in the design of secure systems, but its full **potential** is revealed only when interacting with other algorithms within the **framework** of the design **methodology**.

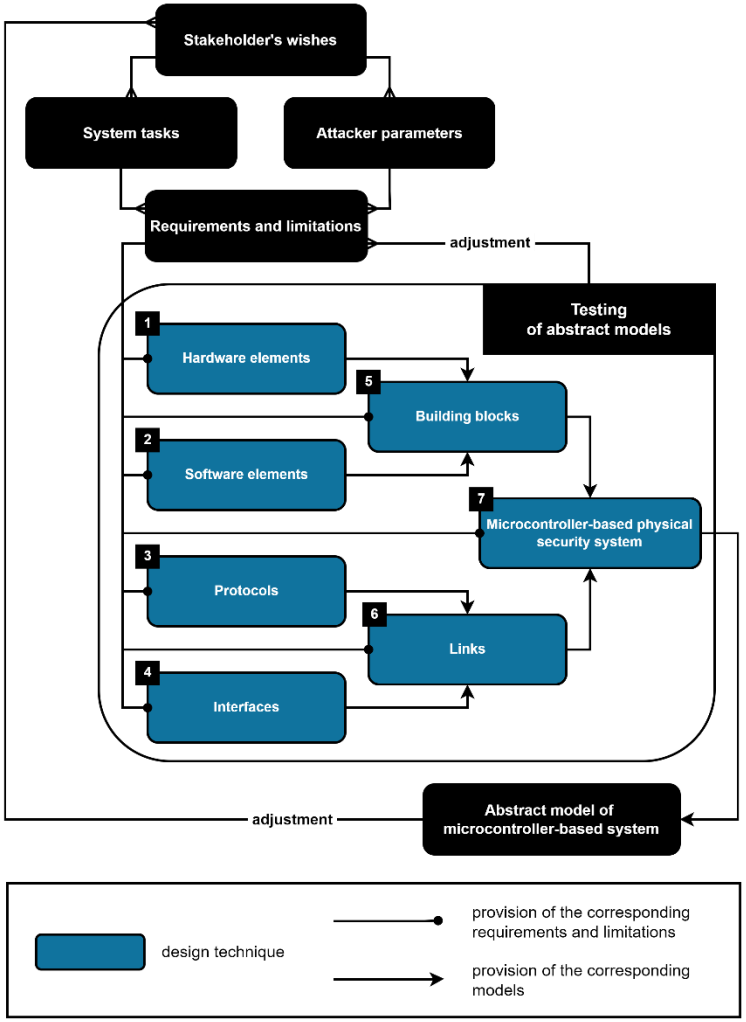
3. Methodology for the design of microcontroller-based physical security systems

Abstract system model design cycle

- This cycle is aimed at designing the **abstract model** of the microcontroller-based system, taking into account the **hierarchy** of its components, their **compatibility**, **requirements** and **dependencies**.

- The cycle is **automated**, the operator is required for the **translation of wishes** of the stakeholder.

	Design technique	Input	Output
1	hardware element	requirements and limitations, hardware sub-elements models (NULL is possible)	model of hardware element
2	software element	requirements and limitations, software sub-elements models (NULL is possible)	model of software element
3	protocol	requirements and limitations	model of protocol
4	interface	requirements and limitations	model of interface
5	building block	requirements and limitations, hardware sub-elements models (NULL is possible), software sub-elements models (NULL is possible), building sub-blocks models (NULL is possible)	model of building block
6	link	requirements and limitations, model of protocol, model of interface	model of link
7	system	requirements and limitations, building blocks, links models, sub-systems models (NULL is possible)	model of system

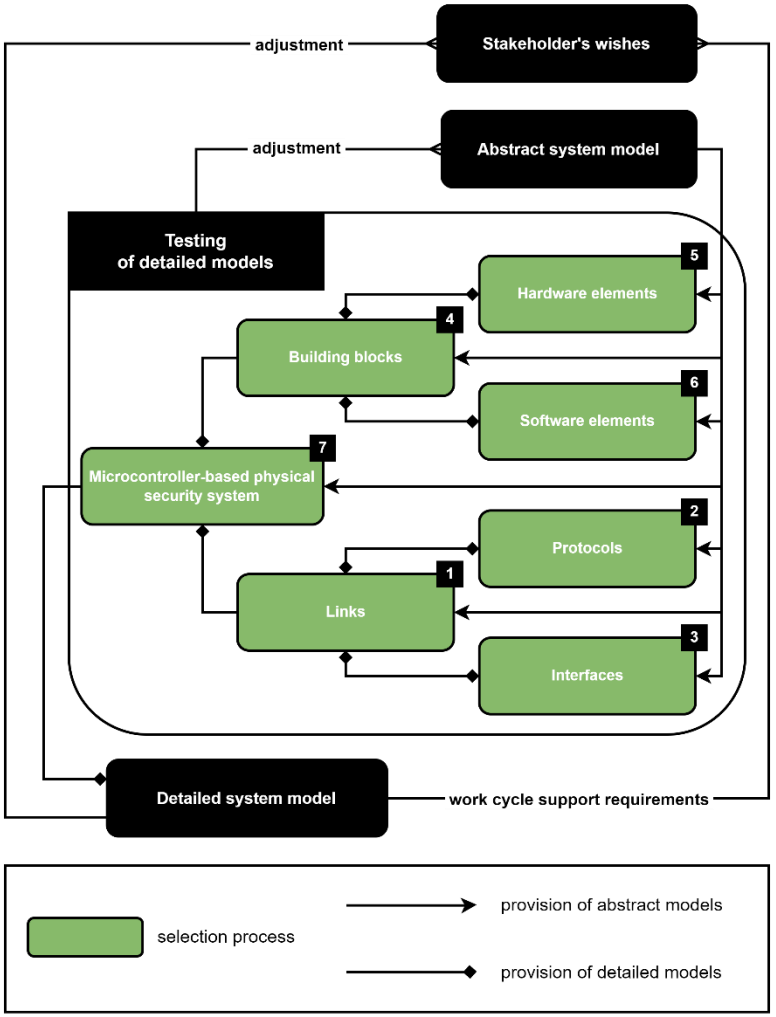


Detailed system model design cycle

- This cycle is aimed at designing the **detailed model** of the microcontroller-based system, in accordance with the **hierarchy** and **mutual dependencies** of the system components as well as available **implementations**.

- The cycle is **automated**, involvement of the operator is **possible** during the **selection of** the **implementations**.

	Selection process	Input	Output
1	link	link model, already selected elements (NULL is possible)	selected link with parameters
2	protocol	protocol model, already selected elements (NULL is possible)	selected protocol with parameters
3	interface	interface model, already selected elements (NULL is possible)	selected interface with parameters
4	building block	building block model, already selected elements (NULL is possible)	selected building block with parameters
5	software element	software element model, already selected elements (NULL is possible)	selected software element with parameters
6	hardware element	hardware element model, already selected elements (NULL is possible)	selected hardware element with parameters
7	system	system model, already selected elements (NULL is possible)	system model with parameters



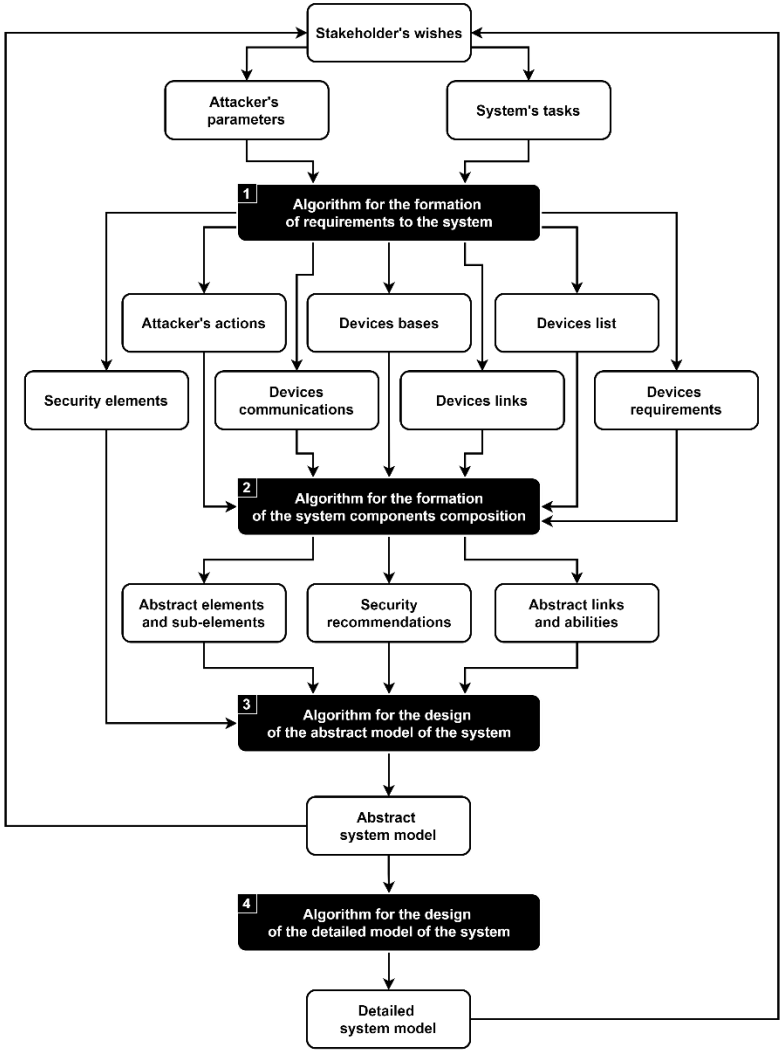
Methodology for the design

- The **workflow** of the **methodology** can be represented through its **connection** with the **algorithms**:
  1. formation of requirements for the system;
  2. formation of the system component composition;
  3. design of the abstract model of the system;
  4. design of the detailed model of the system.

  In such a situation, first 3 algorithms are representing the **abstract** model design **cycle**, while the last one — **detailed** model design **cycle**.

- The main idea of the developed **methodology** is to provide an **automated tool** for the design of microcontroller-based physical security systems that are protected from **cyber-physical attacks**.

- This **methodology** allows one to **reduce** the number of **weak places** and **architectural defects**, thereby significantly reducing the **attack surface** of the system.



Methodology for the design

- Developed methodology provides a **new approach** to the design, which allows one to **integrate** various **design techniques** on the basis of the extendable hierarchical relational model **composition** and **decomposition**.
- The suggested approach is **modular** and **extensible**, takes into account the security of the **physical layer** of the system, works with the **abstract** system representation and is looking for a **trade-off** between the security of the final solution and resources expended on it.
- The methodology is aimed at ensuring the **protection** of the system against **cyber-physical attacks** at the **design stage**, considers **security** elements as an **integral part** of the system and checks if the system can be designed in accordance with given **requirements** and **limitations**.
- Note: the methodology is **not aimed** to replace **security experts**. Moreover, the **quality** of the provided solution **directly depends** on the **correctness** and **completeness** of the **database**. But it can be useful for an expert to **automate routine tasks** and **provide alternative solutions**.

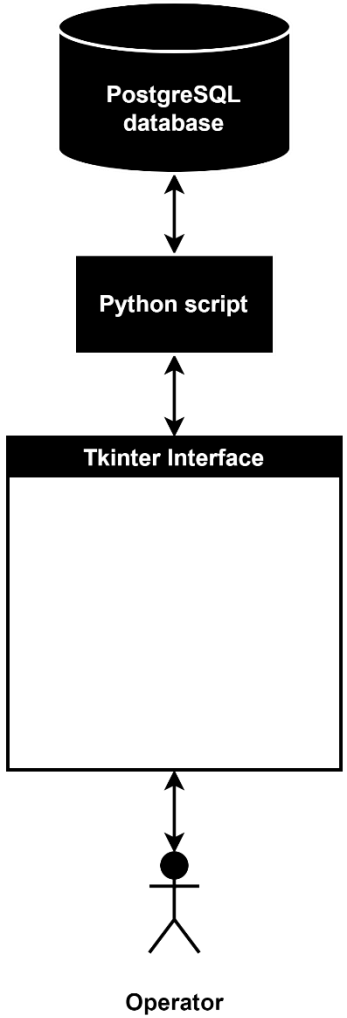
4. Software implementation of the methodology for the design of microcontroller-based physical security systems

Software implementation of the methodology

- Software implementation is an application that consists of **Python script**, **PostgreSQL** database and **Tkinter interface**.
- PostgreSQL **database** is required to store data about the extendable set-based hierarchical relational **model** as well as data for **algorithms** and **methodology** that is required for decision making.

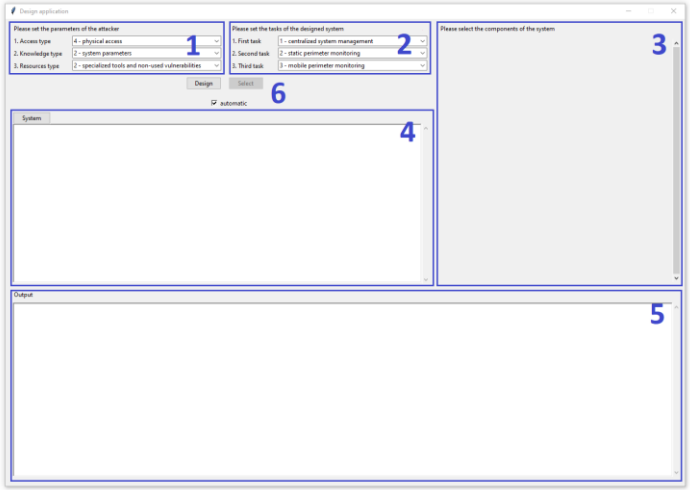
  The developed database contains more than **90 tables**, while the database initialization contains more than **2300 lines** of **PL/pgSQL** queries.
- Python **script** represents the **implementation** of the **algorithms** and **methodology**. Each algorithm is implemented as a number of **functions**, while all functions are **connected** with each other in a single methodology.

  The developed script contains more than **3000 lines** of code (including **comments**) and works with such imports as **psycopg2**, **tkinter**, **pygubu**, **networkx**, **json**, **functools** and **time**.
- Tkinter **interface** is required to receive **input data** from the **operator**, namely, **parameters** of the attacker and **tasks** of the designed system, as well as to provide the **output data** to the operator.



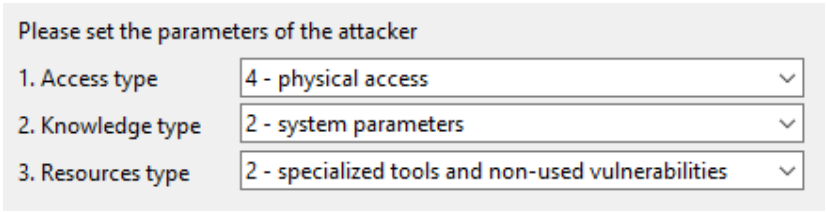
Software implementation of the methodology

- The **interface** consists of **6 main parts**:
  1. **Input of the parameters of the attacker** against which the designed system needs to be protected.
  2. **Input of the tasks** that need to be solved by the designed system.
  3. **Frame to display the process of selection** of components of the designed system.
  4. **Frame to display the work log of the design methodology** for microcontroller-based systems.
  5. **Frame to display the results of work of the design methodology** for microcontroller-based systems.
  6. **Control buttons** of the application.

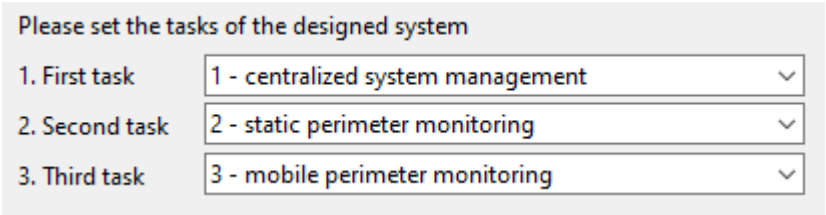


- The **interface** is an important part of the software **implementation**, because it provides a possibility for the **operator** to work with the design **methodology**.

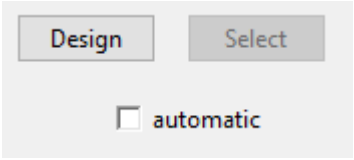
Software implementation of the methodology



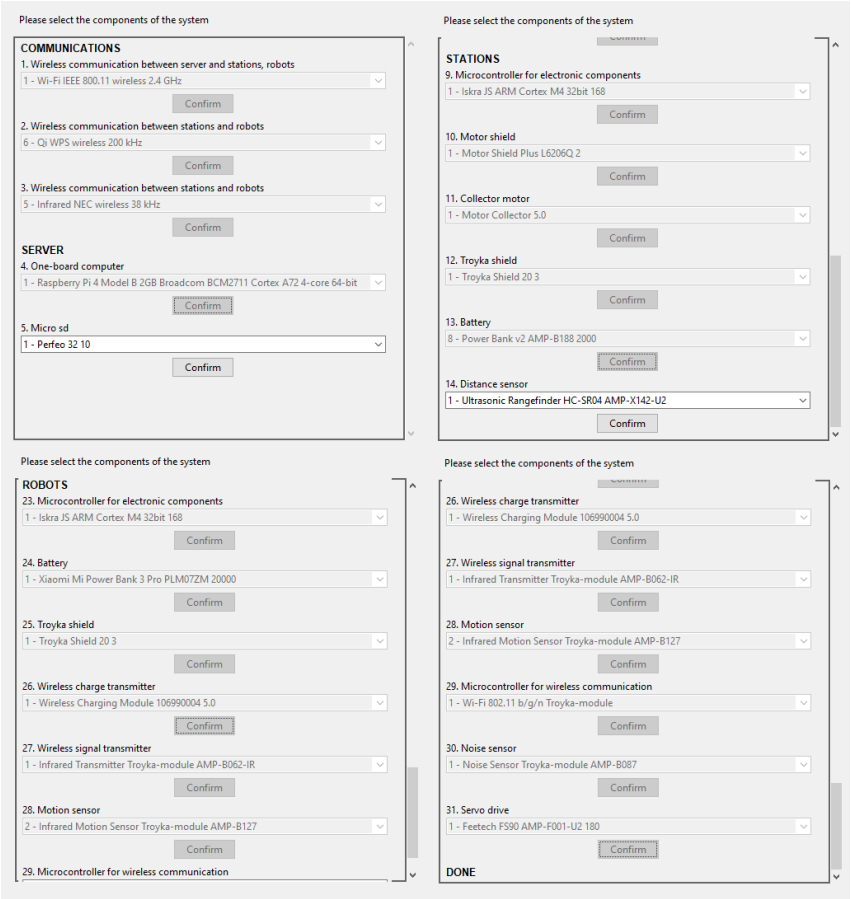
Part 1. Attacker's parameters



Part 2. System's tasks

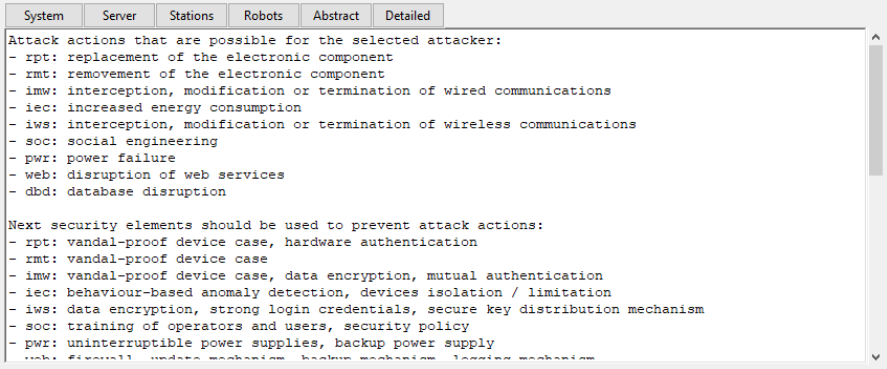


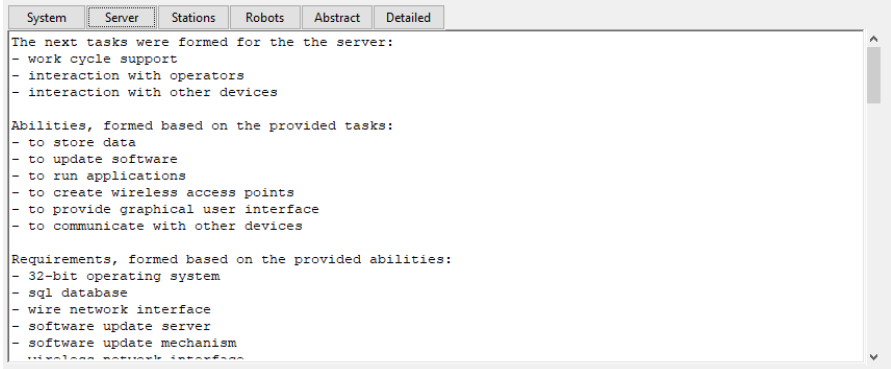
Part 6. Control buttons

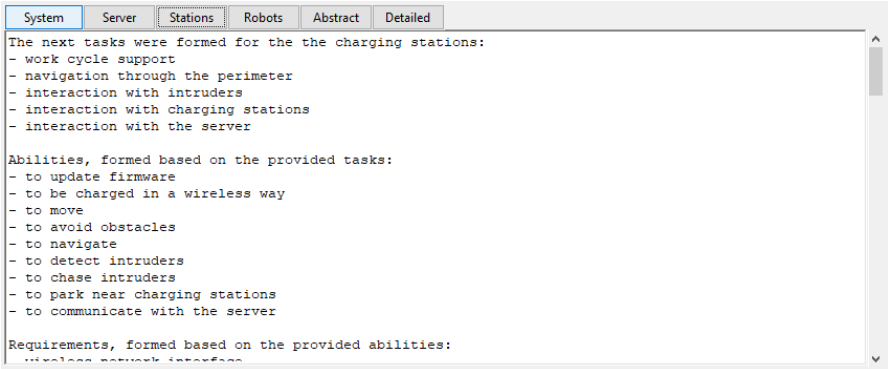


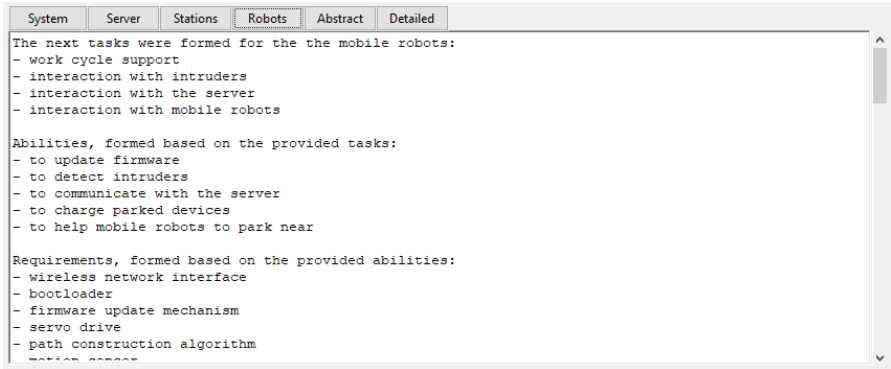
Part 3. Selection process

Software implementation of the methodology









Part 4. Work log

Software implementation of the methodology

Output

```
SERVER
========================================
Selected components:
- one-board computer | name: Raspberry Pi 4 Model B 2GB | soc: Broadcom BCM2711 | arm: Cortex A72 4-core 64-bit | soc_fq_ghz: 1.5 | ram_gb: 2.0 | voltage_v: 5.0 | cu
rrent_a: 3.0 | length_mm: 85.0 | width_mm: 56.0 | height_mm: 17.0 | price_rub: 5890.0 | energy_ma: 540.0
- micro SD | name: Perfeo | memory_gb: 32 | class: 10 | speed_mbs: 20 | price_rub: 560
- battery | name: Xiaomi Mi Power Bank 3 Pro | code: PLM07ZM | capacity_mah: 20000 | current_a: 3.0 | length_mm: 73.5 | width_mm: 153.5 | height_mm: 27.5 | weight_g:
440 | price_rub: 2554
- 32-bit operating system | name: Raspberry Pi OS | extra: with desktop and recommended software | version: 5.10 | size_mb: 2868
- sql database | name: PostgreSQL | os: Debian | bit: 64 | version: 12 | size_mb: 500

To be developed:
- application with graphical user interface: devices isolation / limitation algorithm, devices communication algorithm, devices behaviour-based anomaly detection alg
orithm, data presentation algorithm, data processing algorithm, application-database connection
- wireless access point: strong login credentials, secure key distribution mechanism, brute-force protection algorithm, communication data encryption and decryption
algorithm

After implementation it is recommended:
- to remove physical update interfaces from this device

Price: 9004.0 rub | Energy consumption: 540.0 mAh | Voltage: 5.0 V | Current: 3.0 A | Size: 85.0x153.5x44.5 mm | MCSD FM: 29400.0 MB | Charge: 37.04 hours
```

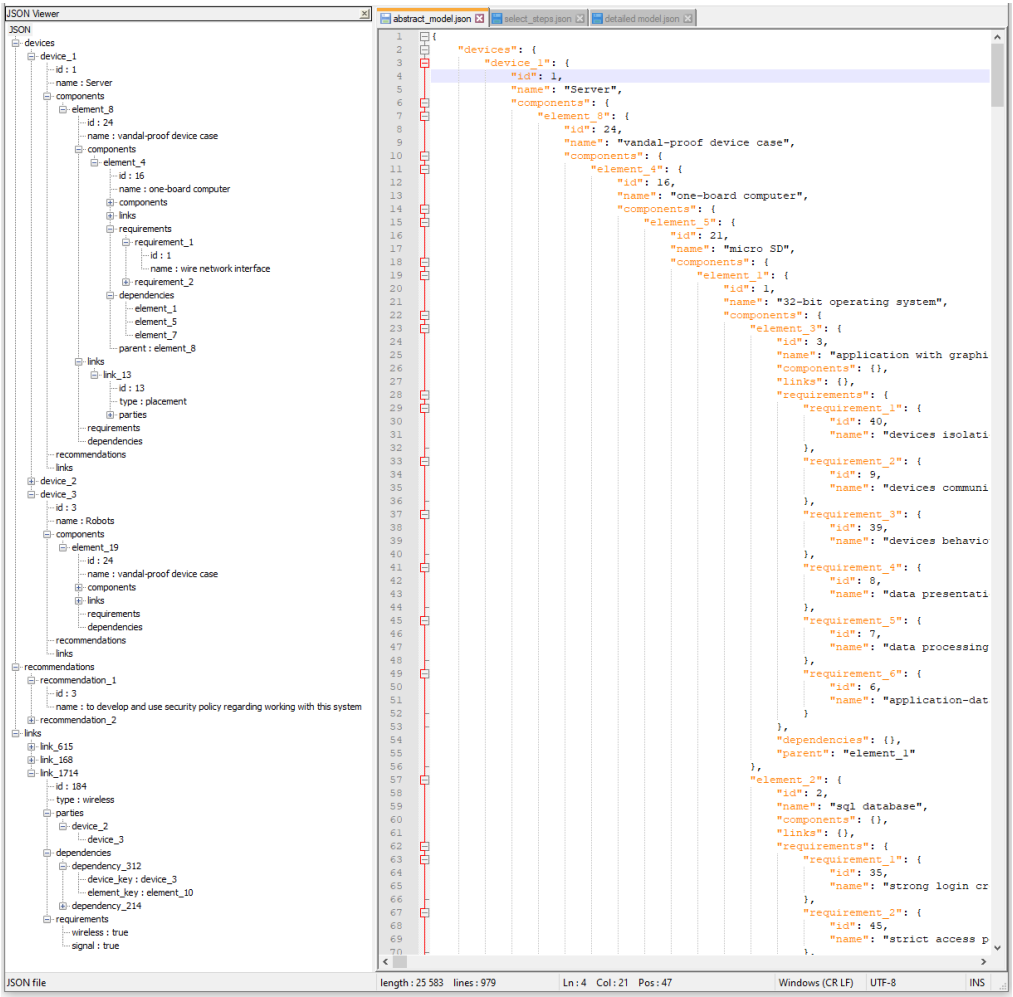
Output

```
========================================
STATIONS
========================================
Selected components:
- microcontroller for electronic components | name: Iskra JS | cpu: ARM Cortex M4 32bit | clock_fq_mhz: 168 | flash_kb: 1024 | voltage_v: 3.3 | pins: 26 | length_mm:
69.0 | width_mm: 53.0 | height_mm: 20.0 | price_rub: 1490 | current_a: 0.4
- troyka shield | name: Troyka Shield | svg: 20 | i2c: 3 | spi: 1 | length_mm: 69.0 | width_mm: 53.0 | height_mm: 19.0 | price_rub: 740
- battery | name: Xiaomi Mi Power Bank 3 Pro | code: PLM07ZM | capacity_mah: 20000 | current_a: 3.0 | length_mm: 73.5 | width_mm: 153.5 | height_mm: 27.5 | weight_g:
440 | price_rub: 2554
- microcontroller for wireless communication | name: Wi-Fi | interface: 802.11 b/g/n | type: Troyka-module | code: AMP-B081 | frequency_ghz: 2.4 | flash_kb: 512 | vo
ltage_v: 3.3 | energy_ma: 250 | length_mm: 50.8 | width_mm: 25.4 | height_mm: 5.0 | price_rub: 850
- servo drive | name: Feetech FS90 | code: AMP-F001-U2 | rotation_dg: 180 | rot_mom_kgsm: 1.3 | rot_speed_s: 0.275 | voltage_v: 5.0 | energy_ma: 150 | length_mm: 32.
0 | width_mm: 13.0 | height_mm: 32.0 | weight_g: 9 | price_rub: 390
- wireless signal transmitter | name: Infrared Transmitter | type: Troyka-module | code: AMP-B062-IR | frequency_khz: 38 | angle_dg: 30 | voltage_v: 3 | length_mm: 2
5.4 | width_mm: 25.4 | height_mm: 5.0 | price_rub: 150
- noise sensor | name: Noise Sensor | type: Troyka-module | code: AMP-B087 | voltage_v: 3.3 | energy_ma: 10 | length_mm: 25.4 | width_mm: 25.4 | height_mm: 5.0 | pri
ce_rub: 740
- motion sensor | name: Infrared Motion Sensor | type: Troyka-module | code: AMP-B127 | voltage_v: 3.3 | energy_ma: 0.5 | angle_g: 110 | range_m: 7 | length_mm: 25.4
| width_mm: 25.4 | height_mm: 15.0 | weight_g: 30 | price_rub: 530
- wireless charge transmitter | name: Wireless Charging Module | code: 106990004 | voltage_v: 5.0 | current_ma: 600 | distance_mm: 20 | length_mm: 17.0 | width_mm: 1
2.0 | height_mm: 4.0 | weight_g: 20 | price_rub: 550
```

Part 5.
Output log

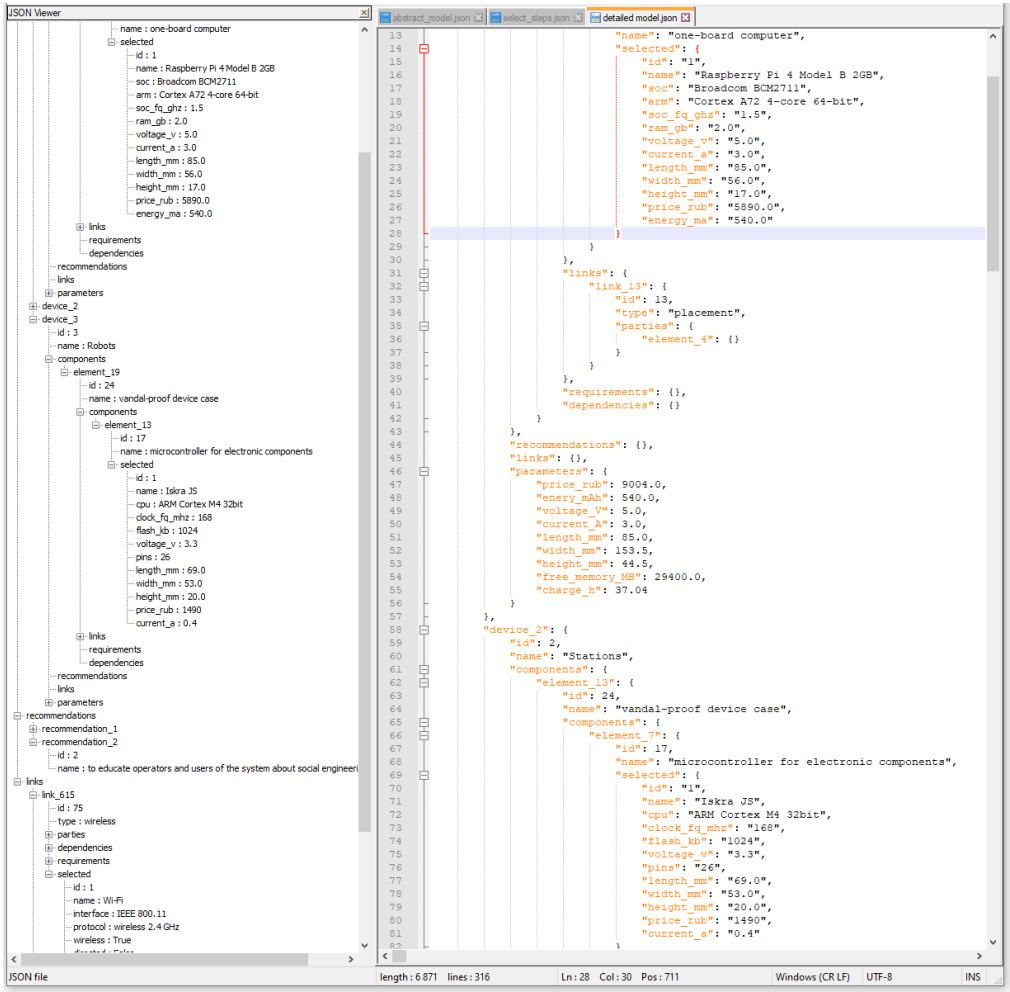
Software implementation of the methodology

- **Abstract log** contains the abstract system **model** in **JSON** format:
  - **system** contains **devices**, **recommendations** and **links**;
  - each **device** contains **id**, **name**, **components** and **recommendations**;
  - each **recommendation** (system/device) contains **id** and **name**;
  - **components** of each **device** contain **elements**, each of which has its own **id**, **name**, **components**, **links**, **requirements**, **dependencies** and **parent tag**;
  - element **link** contains **id**, **type** and **parties**;
  - element **requirement** contains **id** and **name**;
  - element **dependency** contains **keys of elements** that depend on them;
  - element **parent tag** contains the **key of the element** that contains this element as its sub-element.



Software implementation of the methodology

- **Detailed log** contains the detailed system **model** in **JSON** format (complements abstract one):
  - each **device** has **parameters** calculated based on the parameters of the selected **elements** — **price**, **energy consumption**, **voltage**, **current**, **length**, **width**, **height**, **free memory** and **battery life**;
  - for each **selected element**, data appears about the **controller** or **component** used, for example, the **name** of the single-board computer, the **processors** used and their **frequency**, the amount of **RAM**, supply **voltage**, **length**, **width** and **height**, **price** and **energy consumption**;
  - each **selected link** between devices has information about its **name**, **interface**, **protocol** and **type**, for example, the link can be **wireless**, **broadcast**, **transmits data**, supports **access point mode**, has **encryption** and **authentication**.



Experimental evaluation of the methodology for the design of microcontroller-based physical security systems

Experimental evaluation of the methodology

- **The plan of the experiment is as follows:**
  1. Description of the **system** to be designed.
  2. Manual fulfillment of the **database** with data about the system.
  3. Analysis of the software **implementation** compliance with the **functional requirements**.
  4. Analysis of **time consumption** of the application.
  5. Analysis of **resource consumption** of the application.
  6. **Comparison** of the implementation with other solutions.
  7. Investigation of the **dependencies** between the design time and the parameters of the attacker.

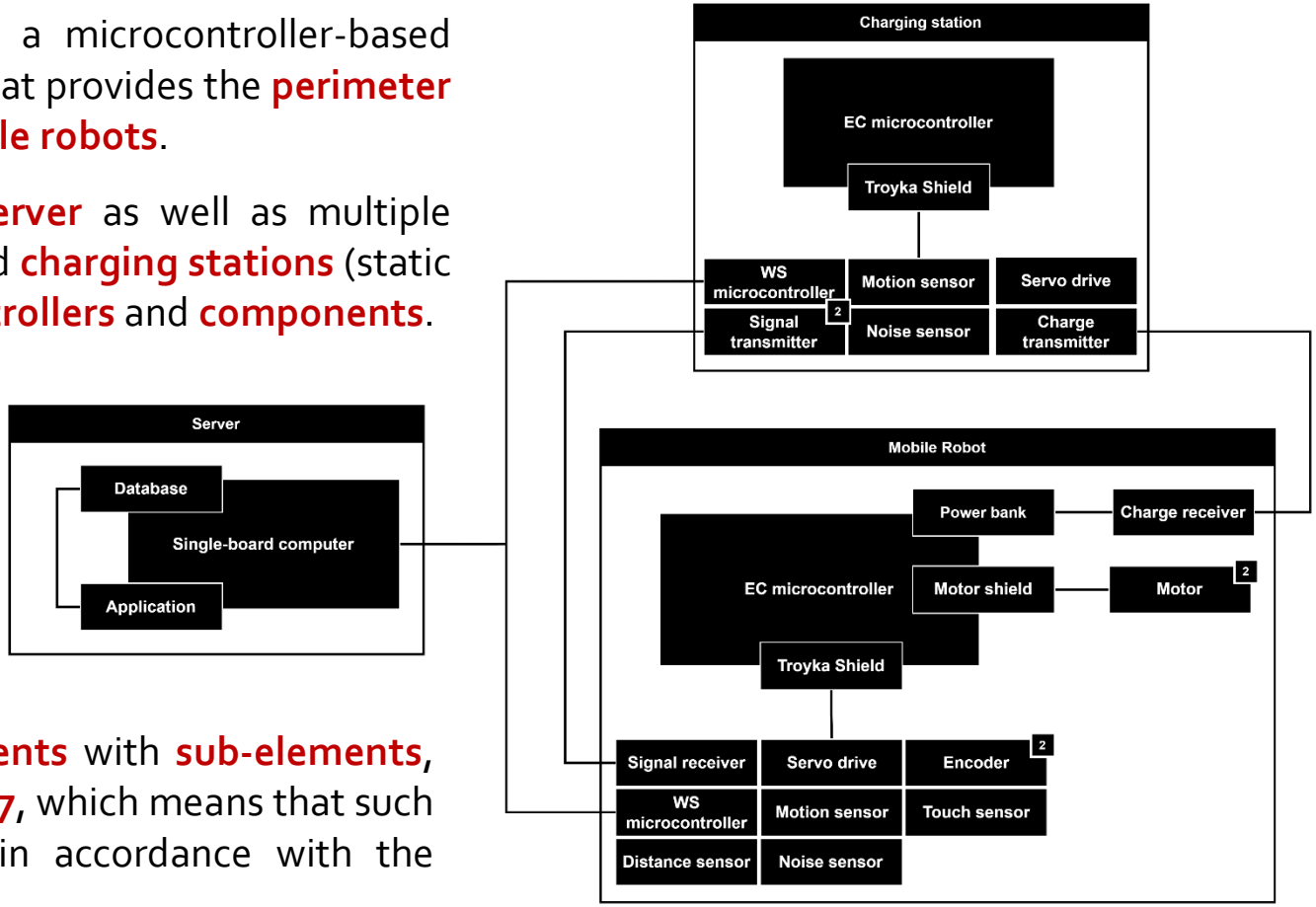
- For the **experiment**, the software **implementation** of the **methodology** was executed on the **computer** with Windows 10 x64 operating system, Intel Core i7-4790 CPU **3.60GHz** (**8 cores**) processor, **2 TB** HDD and **32 GB** RAM.

- **Functional requirements: abstract** representation, **trade-off** between security and resources, **no restrictions** on platforms and architectures, **extensibility** of the approach, **physical layer** security is taken into account.

Experimental evaluation of the methodology

- It was decided to design a microcontroller-based physical security **system** that provides the **perimeter monitoring** based on **mobile robots**.

  This system contains a **server** as well as multiple **robots** (mobile objects) and **charging stations** (static objects) with different **controllers** and **components**.



- Server consists of **8 elements** with **sub-elements**, station — **12** and robot — **17**, which means that such a system is appropriate in accordance with the provided requirements.

Experimental evaluation of the methodology

Time consumption:

- $TIME_1^{ACC} = 1\ sec$ — acceptable time for the design of the **abstract** system model;
- $TIME_2^{ACC} = 4\ sec$ — acceptable time for the design of the **detailed** system model.

Stage	T_i^{min}, ms	T_i^{max}, ms	$T_i = \frac{3T_i^{min} + 2T_i^{max}}{5}$	$\sigma^2(T_i) = 0.4(T_i^{max} - T_i^{min})^2$
1	3.99	6.98	5.19	3.58
2	11.95	14.72	13.06	3.07
3	51.86	59.48	54.91	23.23
Total for the stage, ms			73.16	29.88

$P(TIME \le TIME_1^{ACC}) = 0.9999$, the **requirement is satisfied**

Stage	T_i^{min}, ms	T_i^{max}, ms	$T_i = \frac{3T_i^{min} + 2T_i^{max}}{5}$	$\sigma^2(T_i) = 0.4(T_i^{max} - T_i^{min})^2$
1	13.01	13.49	13.20	0.09
2	503.19	505.77	504.22	2.66
3	32.58	36.92	34.32	7.53
Total for the stage, ms			548.14	10.28

$P(TIME \le TIME_2^{ACC}) = 0.9999$, the **requirement is satisfied**

Experimental evaluation of the methodology

- **Resource consumption** is divided into particular indicators: **CPU**, **HDD** and **RAM**. The values were measured with the help of **psutil** Python library. According to the **requirements**:

$$P_R(RES_N \leq R^{ACC}) \geq P_R^{ACC}$$

where P_R — probability that the resources RES_N spent on the **design process** of microcontroller-based physical security system do not exceed the **acceptable value** RES^{ACC} (0.25); P_R^{ACC} — **acceptable value** of probability (0.99).

- **Results:**
  - CPU: Core 1: th1 — 0.6%, th2 — 0.7%; Core 2: th3 — 6.7%, th4 — 0.0%; Core 3: th5 — 2.4%, th6 — 0.5%; Core 4: th7 — 1.1%, th8 — 0.0%; Core 5: th9 — 0.0%, th10 — 0.0%; Core 6: th11 — 0.3%, th12 — 0.2%; Core 7: th13 — 0.0%, th14 — 0.5%. $R_{CPU} = 0.0089$ — **satisfied**.
  - HDD: size of the implementation is 42.1 MB (app – 27.1, db – 15.0). $R_{HDD} = 0.000021$ — **satisfied**.
  - RAM: during the design process an additional 100290560 bytes of memory were used. $R_{RAM} = 0.0029$ — **satisfied**.

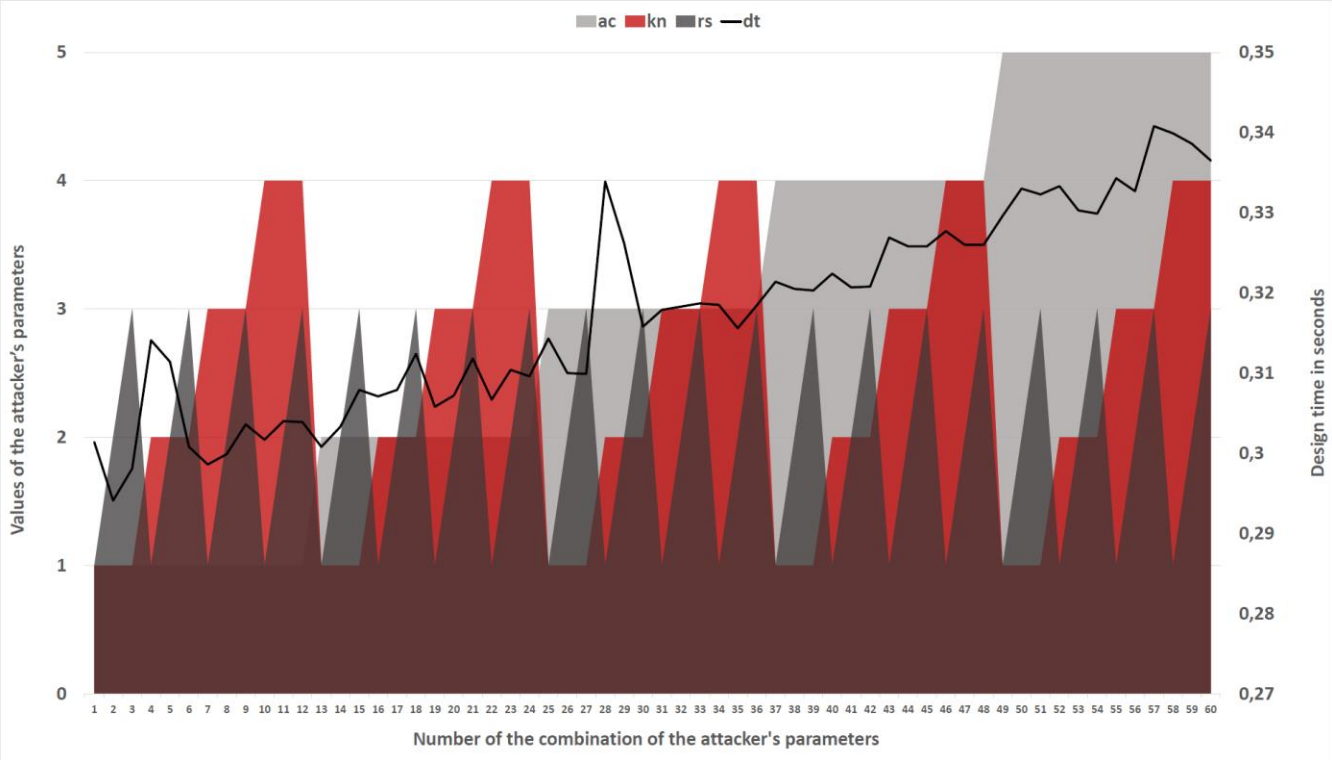
Experimental evaluation of the methodology

Solutions	Levels of the system				Classes of attack actions			
	cn ↔ cr	*cr ↔ cr*	*dv ↔ dv*	*st ↔ st*	*cn*	*cr*	*dv*	*st*
[Hu et al.]	–	–	–	–	–	*	*	–
[Penas et al.]	–	–	–	–	–	*	*	–
[Scott-Hayward]	+	+	–	–	+	+	–	–
[Lin et al.]	+	+	–	–	–	+	–	–
[Google IoT]	–	–	–	+	–	–	+	+
[ARM PSA]	–	–	–	+	–	–	+	+
[Kaspersky ICS]	–	–	–	+	–	–	+	+
[Azure IoT]	–	–	–	+	–	–	–	+
[Intel IoT]	–	–	–	+	–	–	+	+
[MindSphere]	–	–	–	+	–	–	–	+
[Desnitsky et al.]	+	+	–	–	+	+	–	–
[SecFutur project]	–	–	+	–	–	+	+	–
Developed	+	+	+	+	+	+	+	+

- **Comparison** was made based on the **publicly available** data. For each **parameter**, the **presence** or **absence** of its consideration during the **design process** was determined.

Experimental evaluation of the methodology

- The attacker's model characterizes his ***capabilities*** in accordance with 3 parameters: ac – type of ***access*** (from 1 to 5), kn – type of ***knowledge*** (from 1 to 4) and rs – type of ***resources*** (from 1 to 3).



min — 0.2941
max — 0.3408

- The scale on the left from **0** to **5** reflects changes in the attacker's **parameters** (ac, kn and rs as **area charts**), while the scale on the right from **0.27** to **0.35** — design **time** (dt as **black line**).

Summary of the work done

Summary of the work done

		Result	Conclusion
Functional requirements	Abstract	Methodology represents the system as an abstract model	✓
	Trade-off	The number of integrated security elements depends on the attacker's parameters	✓
	Any platform	Implementations of any platforms or architectures can be used, until they can be connected with the elements of the abstract model	✓
	Extensibility	It is possible to add additional layers of abstraction, integrate additional algorithm, change the model of the attacker and attack actions	✓
	Physical layer	The developed model takes into account hardware elements and links between them	✓
Non-functional requirements	Time consumption	$P_{NE}(TIME^{AM} \leq TIME_1^{ACC}) = 0.9999,$ $P_{NE}(TIME^{DM} \leq TIME_2^{ACC}) = 0.9999$	✓
	Resources consumption	$P_{RES}(RES_{CPU} \leq R^{ACC}) = 1,$ $P_{RES}(RES_{HDD} \leq R^{ACC}) = 1,$ $P_{RES}(RES_{RAM} \leq R^{ACC}) = 1$	✓
	Validity	$\lvert LEVELS_N \rvert \geq max(\lvert LEVELS_{s \in S} \rvert),$ $\lvert ATTACKS_N \rvert \geq max(\lvert ATTACKS_{s \in S} \rvert),$ $\lvert LEVELS_N \times ATTACKS_N \rvert \geq max(\lvert LEVELS_{s \in S} \times ATTACKS_{s \in S} \rvert)$	✓
Objective function		$O_F: \lvert LEVELS_N \times ATTACKS_N \rvert \rightarrow max$	✓

It means that the ***goal*** of this work ***is reached***

Summary of the work done

- The main results are published in ***22 articles*** of which ***15 publications*** in journals peer-reviewed by ***Web of Science*** or ***Scopus***, ***7 publications*** in journals from the list of ***Russian Higher Attestation Commission***.

  The main publications are as follows:

  - Dmitry Levshun. Formation of requirements for the design process of secure cyber-physical systems. Journal of Instrument Engineering. 2020. Vol. 63, N 11. P. 1040—1045. DOI: 10.17586/0021-3454-2020-63-11-1040-1045. [in Russian] (**RHAC**)
  - Dmitry Levshun, Andrey Chechulin, Igor Kotenko. Design of secure microcontroller-based systems: application to mobile robots for perimeter monitoring. Sensors. 2021. Accepted 08.12.2021. (Scopus, WoS, **Q1**)
  - Dmitry Levshun, Yannick Chevalier, Igor Kotenko, Andrey Chechulin. Design and verification of a mobile robot based on the integrated model of cyber-physical systems // Simulation Modelling Practice and Theory, Vol. 105, 2020. DOI: 10.1016/j.simpat.2020.102151. (Scopus, WoS, **Q2**)
  - Dmitry Levshun, Andrey Chechulin, Igor Kotenko. The application of the methodology for secure cyber-physical systems design to improve the semi-natural model of the railway infrastructure // Microprocessors and Microsystems, November 2020, P. 103482. DOI: 10.1016/j.micpro.2020.103482. (Scopus, WoS, **Q3**)
  - Dmitry Levshun, Diana Gaifulina, Andrey Chechulin, Igor Kotenko. Problematic Issues of Information Security of Cyber-Physical Systems // Informatics and Automation. Vol. 19. No. 5. 2020. P. 1050-1088. DOI: 10.15622/ia.2020.19.5.6. (Scopus, **Q3**)

- In addition, ***11 certificates*** of state registration were received, namely, ***8 computer programs*** and ***3 databases*** (database and script are registered).

Summary of the work done

- The research results presented in this work were used in the ***7 research and development projects***. The main projects are as follows:
  - "Models, techniques and methodology for design and verification of secure cyber-physical systems". Research grant #19-37-90082 "PhD students" of Russian Foundation of Basic Research, **2019-2022**.
  - "Security Aspects of Cyber-Physical Systems". Research grant #19-17-50205 of Russian Foundation of Basic Research, **2019-2020**.
  - "Research and development of an integrated security system based on embedded intelligent microcontrollers". Grant from the Fund for Assistance to the Development of Small Forms of Enterprises in the Scientific and Technical Sphere (Fund for Assistance to Innovation). START-2 project. Contract #2485GS2/22645 dated 04/11/2018, **2018-2019**.
  - "Development of methods for vulnerability detection for human-computer interaction interfaces of the Smart City transport infrastructure". Research grant #19-29-06099 of Russian Foundation of Basic Research, **2019-2022**.
  - "Methods, Models, Methods, Algorithms, Protocols and Applications for ensuring Information Security of Cyber-Physical Systems". NIR-FUND #717075 of ITMO University, **2017-2019**.

- Moreover, the research results were used by the department of secure communication systems of ***St. Petersburg State University of Telecommunications*** in the educational process of the direction of training 10.03.01 "***Information security***" within the discipline "***Fundamentals of designing secure infocommunication systems***" (work program No. 21.05/446-D).

Thank you for your attention!

The software implementation of the presented design methodology is available for download:

- https://github.com/levshun/PhD-mcbpss_design