CSIT (June 2013) 1(2):143–157
DOI 10.1007/s40012-013-0010-8

ORIGINAL RESEARCH

"A framework for development of secure software"

Kakali Chatterjee · Daya Gupta · Asok De

Received: 28 July 2012 / Accepted: 2 January 2013 / Published online: 12 March 2013

Abstract To design, build and deploy secure systems, we must integrate security into our application development life cycle and adapt current software engineering practices and methodologies to include specific security-related activities. Developers enforces security measures during design phase of software development processes which may end up in specifying security related architecture constraints that are not really necessary. To eliminate this problem, we propose a Framework for Security Engineering Process that involves converting security requirements and threats into design decisions to mitigate the identified security threats. The identified design attributes are prioritized and a security design template is prepared. We have stored various cryptographic techniques and their analytic attributes in the repository to find out the specific cryptographic technique that would eventually help in the later stages of the design process by eliminating unnecessary design constraints in a particular scenario.

Keywords Security engineering process · Security requirement engineering · Security design engineering · Security design template · Cryptographic techniques · Cryptographic attacks

K. Chatterjee (✉) · D. Gupta
Computer Engineering Department, Delhi Technological University, Bawana Road, Delhi 110042, India
e-mail: kakali2008@gmail.com

A. De
NIT, Patna, India

1 Introduction

Security engineering focuses on processes and methods for implementing security in software and related systems.The rapid development of internet based software systems which maintain sensitive information and carry out critical activities has initiated an increasing attention on how to build secure software [1]. Current software development processes enforce security measures during design phase which may end up in specifying security related architecture constraints that are not really necessary. As a result the final software system may use inefficient mechanism with increased cost.

This leads to development of security measure at the early phase of SDLC (Software Development Life Cycle). Firesmith [2] defined security requirement as high level requirements that gives specification of system behavior and distinguish these from security related architectural constraints so that requirement engineers can discover true security requirements. There are proposal on discovering and eliciting of security requirements like abuse case [3], misuse case [4, 5], common criteria [6], attack trees [7]. The root cause of security requirements is the mitigation of different types of risk associated with the threats. To prioritize the elicited security requirements, one needs to measure underlying risks. There are also risk management methods like EBIOS [8], OCTAVE [9], CORAS [10], and CRAMM [11] which enforce security levels based on risk measure. But these proposals of security engineering are not integrated in conventional software development process.

Recently there are some proposals on secure development [12–14] consisting of modelling languages and methodologies like secure troops extension of troops methodology [15], intentional anti model extension of

KAOS methodology with security requirement oriented construct [16], unify security constructs and risk management concepts in information system development method [17]. A framework of security requirements engineering is found in [18] to facilitate production of security requirements, but it does not consider the design constraints in a particular development environment. Researchers are yet to develop a precise method on how to apply those principles to their own software, or how the security concerns are addressed while making architectural, logical, and physical design decisions.

In our earlier work [19–21] we proposed a framework for secure software development where one should have security engineering activities like (i) Security requirements elicitation, analysis & prioritization, specification and management; (ii) Appropriate design decision; (iii) Implementation of all functionalities incorporating design decision; (iv) Testing of security modules. Our research shows that considerable work is found in literature for (i) and (iii), but very little attempt has been made for (ii). Once security requirement engineering is through, one needs to take proper design decision for security requirements analogues to conventional design process for functional and non-functional requirements. If proper design decisions are not taken, then either it can lead to an underdeveloped system with unnecessary design constraints or makes the system vulnerable or unnecessary constraint overflowing the budget.

These design decision for security requirement will require modeling of software environment from systems perspective. In this paper we concentrate on design decision issue (ii). After security requirements are elicited and prioritized, appropriate design decision will consider the security threats in optimal manner which is the need for today's software projects. As these projects are tightly scheduled, they need cost effective solution without disturbing the required availability of needed services.

Our approach involves converting security requirements and threats (which are identified during the security requirement elicitation stage) into design decisions to mitigate the identified security threats. The different types of security requirements are mapped to the different security services. The identified design attributes are prioritized and a security design template (SDT) is prepared based upon the set of important security attributes that influences design choices at the various layers of security engineering process. Finally we find out the specific cryptographic techniques that would eventually help in the later stages of the design process by eliminating unnecessary design constraints in a particular scenario. We have stored various cryptographic techniques and their analytic attributes in the repository. We illustrate our process with suitable case study of Web based Banking using smart cards.

The rest of the paper is organized as follows:

In Sect. 2, background of this research work is discussed; Sect. 3 provides proposed Framework for Security Engineering; Sect. 4 provides case study of web based banking system; Sect. 5 presents implementation and evaluation of the framework; finally we conclude the paper in Sect. 6.

2 Background

Selection of inappropriate software package and security modules may be costly and also adversely affect business processes and functioning of the organization [22]. The common security measures (security goals) are to secure the assets of the organization. Assets are generally defined as anything that has value to the organisation and needs protection from intruder or accidental detriment by recognised actors of the system. Traditionally it is known as information security and is expressed as confidentiality, integrity and availability of information in an organisation which is called as CIA triad.

2.1 Security engineering

Security engineering is a complex process consisting of different security-related activities. Security-related activities[13] include identifying security requirements, prioritizing and management of security requirements, security design, implementing security mechanisms, security testing. With true security requirements specified, most appropriate design decisions can be taken. The design phase specifies how the identified requirements (gathered from the security requirement engineering phase) can be implemented in a given environment and it also proposes an overall structure of the software from a security perspective. Getting a systematic and well organized structure of security functionality and their design decisions is the main goal of this phase.

BLP (Bell-LaPadula model) was a famous security model to achieve multilevel security goal developed in 1973 [23]. The aim of this model was to set rules controlling the information flow to protect sensitive data from unwanted disclosure. BLP model enforced mandatory access control policy based on sensitivity label that was a combination of hierarchical classification levels and non-hierarchical categories. It was adopted by US Department of Defence as the foundation of *Trusted Computer System Evaluation Criteria* and then extensively applied in information security areas, especially in the designation of secure operating systems [24].

In 2002, an aspect oriented design technique is proposed to model and integrate security concerns into design by weaving the aspects in a primary model [12]. A design aspect can be modelled from a variety of perspective. In

this paper only static and interaction aspects views are considered. This paper describes steps for weaving an aspect in primary model. But the technique fails to address the impact of the security concern on each design unit with respect to a given application environment. Also it does not focus on any well defined framework through which developers can make design decisions to develop efficient cost effective secure system.

In 2005, Apvrilla et.al proposes a design methodology using security patterns and design model PICO using UMLsec [13]. The PICO application consists of three different services such as subscription service to handle initial user registration, a presence service to keep a list of online users and an instant messaging service to forward the user's instant message to the recipients. This paper focuses on design of presence service via security patterns and design of the instant messaging service using UMLsec which is just a part of the development process, not a complete solution of the problem.

Lipner et.al [14] discussed a process, "*Trustworthy Computing Security Development Lifecycle* (*or SDL*)" that Microsoft has adopted for the development of software that needs to withstand malicious attack. This paper describes high-level principles for building more secure softwares, such as secure by design, secure by default, secure in deployment and discusses experience with its implementation across Microsoft software. But the software should be architected, designed and implemented so as to protect itself and the information it processes and resist attacks in a particular application environment.

The emergence and fast development of trusted computing technologies in recent years give us some new clues to multilevel security issues and design principles [24], but no proper framework is found to design secure software.

2.2 Cryptographic techniques

To design a secure system, it is essential to identify the kinds of threats and the mechanisms that can be used to protect the system against such threats. Cryptography is a systematic way to secure data from adversaries. A sensitive object (asset) requires encryption, authentication and access control to protect against unauthorized modification. Symmetric or secret key cryptography has been in use for thousands of years and includes any form where the same key is used both to encrypt and to decrypt the text involved. Some of the symmetric algorithms with block size and key size are listed in Table 1 below.

Share of secret key (which is required for both encryption and decryption) can be a major vulnerability where symmetric key systems of cryptography are used. In asymmetric or public key cryptography, this is not an issue in the same way. Two keys (private and public), mathematically related, are used and work together in such a way that plain text encrypted with the one key can only be decrypted with the other. As private key is not shared by the individual, the system avoids the risk of compromising security. Most advanced techniques are based on ECC [25] and HECC [26]. Timings of different operations of some asymmetric algorithms are listed in Table 2 below.

For authentication purpose sometimes hash functions are also used. Hash functions are also called one-way function or collision-resistant one-way function. A hash function takes a message of any size and computes a smaller, fixed-size message called a digest or hash. Some popular Hash Functions are listed in Table 3 below.

2.3 Cryptographic attacks

Usually a system is designed to protect against certain threats, while other threats might not have been addressed [24]. A threat is a harm that can happen to an asset, composed of a threat agent and an attack method [17]. Thus a good authentication scheme should provide protection from different attacks relevant to that protocol. Here we have identified some of the possible attacks for password based authentication and short formed (as UA_1, UA_2 etc.) them. These cryptographic attacks are listed below.

Table 1 Some popular symmetric ciphers [32]

Name of algorithm	Block size (bits)	Key size (bits)	Encryption speed (on 33 MHZ 486SX) (Kb/s)
DES	64	56	35
Blowfish	64	128	182
3DES (Triple DES)	64	168	12
IDEA	64	128	70
AES	128	128	60
CAST	64	128	53
RC5	64	128	86
RC4 (Stream cipher)	One byte at a time	256	164
SEAL (Stream cipher)	One byte at a time	160	381
PIKE (Stream cipher)	One byte at a time	160	62

Table 2 Some popular asymmetric ciphers [32]

Name of algorithm	Encryption (ms)	Decryption (ms)	Sign (ms)	Verify (ms)
RSA (512 bits)	30	160	160	20
RSA (768 bits)	50	480	520	70
RSA (1024 bits)	80	930	970	80
ECDSA (160 bits)	797	281	150	230
ECDSA (233 bits)	882	385	250	521
ECDSA (283 bits)	928	400	25	580
HECDSA (81 bits)	668	191	60	31
HECDSA (83 bits)	893	224	56	32
ElGamal (512 bits)	330	240	250	1370

Table 3 Some popular hash functions [32]

Algorithm	Hash length (bits)	Encryption speed (on 33 MHZ 486SX) (Kb/s)
MD4	128	23
MD5	128	236
HAVAL	128	174
N-HASH	128	29
SHA1	160	75
SHA2	160	70

2.3.1 *Man-in-the-middle attack (UA_1)*

In this type of attack, the attacker intercepts the message sent between the client and the server and replay these intercepted messages within a valid time. The attacker can act as the client to the server or vice versa with recorded messages.

2.3.2 *Denial of service attack (UA_2)*

In this type of attack, an attacker updates password verification information on smart card to some arbitrary value so that legal user cannot login successfully in subsequent login request to server.

2.3.3 *Replay attack (UA_3)*

Replay attack involves passive capture of data and its subsequent retransmission to show unauthorized effect. Any unauthorized malicious user can send duplicate data repeatedly to the receiver which is already sent.

2.3.4 *Perfect forward secrecy (UA_4)*

In perfect forward secrecy, even if the user's password is compromised, it never allows the adversary to determine the session key for past sessions and decrypt them.

2.3.5 *Impersonation attack (UA_5)*

In this type of attack, the attacker impersonates as legitimate client and forges the authentication messages using the information obtained from the authentication scheme.

2.3.6 *Server spoofing attack (UA_6)*

In server spoofing attack, the attacker can manipulate the sensitive data of legitimate users via setting up fake servers. Thus malicious server can get valid login message from the user [27].

2.3.7 *Dictionary attack (UA_7)*

There are two types of dictionary attacks named as Offline dictionary attack and Online dictionary attack. In Offline dictionary attack, the attacker can record messages and attempts to guess user's identity and password from recorded messages. In Online dictionary attack, the attacker pretends to be legitimate client and attempt to login on to the server by guessing different words as password from a dictionary.

2.3.8 *Stolen verifier attack (UA_8)*

An attacker who steals the password-verifier (e.g., hashed passwords) from the server can use the stolen-verifier to impersonate a legal user to log in the system [27].

2.3.9 *Smart card loss attack (UA_9)*

When the smart card is lost or stolen, unauthorized users can easily change the password of the smart card, or can guess the password of the user by using password guessing attacks, or can impersonate the user to log in the system [27].

2.3.10 *Insertion attack (UA_{10})*

In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering client's password or gaining unauthorized access.

3 Proposed framework for security engineering

Security engineering deals with security-related activities which include identifying security requirements, prioritizing and management of security requirements, security design, implementing security mechanisms, security testing.The proposed framework for overall security engineering process (SEP) is shown in Fig. 1.

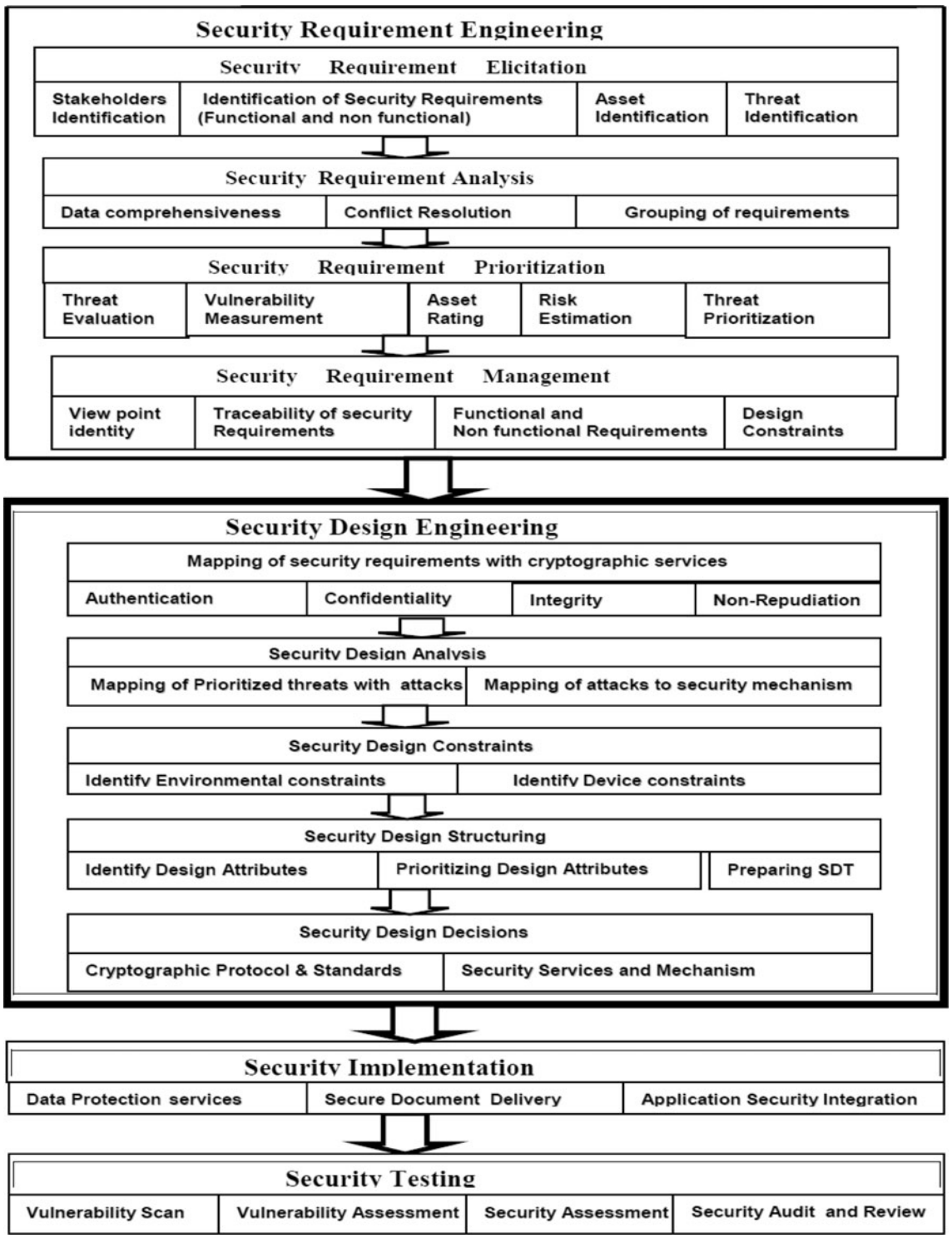

Fig. 1 Framework for security engineering process

The concept of this framework for security engineering process is an attempt to propose a design framework taking the view of stakeholders as well as environmental constraints in the earlier software development phases [21]. We now discuss in detail each activity of this proposed framework.

3.1 Security requirement engineering

This phase involves discovering security requirements, eliciting, analyzing and managing them. It consists of four different stages: Security Requirement Elicitation, Security Requirement Analysis, Security Requirement Prioritization

and Security Requirement Management. All the different activities perform in each stage of this process are explained below:

3.1.1 Step 1: security requirement elicitation

In Security Requirement Elicitation phase different tasks are performed as explain below:

(i) Identify various stakeholders of the system using view-point analysis [28, 29]. We have identified the various abstract classes of actors as direct and indirect actors. Direct actors are those who directly interact with the system such as human, software system and hardware devices. Indirect actors refer to developers who develop software and people who regulate application domain.
(ii) Identify the functionalities of each actor conceptualized in the previous step and also determine associated non-functional requirements.
(iii) Identify the threats associated with each of the functional requirements or data which is used by the functionality.
(iv) Define the security requirements such as authentication, integrity, non-repudiation etc. to mitigate these threats.

3.1.2 Step 2: security requirement analysis

The various tasks perform in analyzing the security requirements are as follows:-

(i) *Checking for completeness* We make a check list to check that the elicited security requirements have mitigated all the threats to the functionality of the system.

(ii) *Conflict resolution* We resolve the contradictions that may exist in the security requirements elicited from different viewpoints.
(iii) *Grouping of requirements* This step consists of identifying the security requirements that can be grouped together.

3.1.3 Step 3: security requirement prioritization

As security requirements are to mitigate threats and avoid vulnerability and risk, they will be prioritized on the measure of threat, vulnerability and risk. So prioritization is done in following two steps:

Evaluation of threats
Prioritization of security requirement.

In the first step, we will evaluate the threats based on the estimated risk value. For this we have to perform the following tasks:

(i) *Threat assembling* After identifying the threats, a repository of the threats will be developed as in common criteria based approach [6, 30]. Actor profiles will be maintained also in this repository. Thus predefined threats can be retrieved from the repository according to the profile of the actor.
(ii) *Threat rating* After threat assembling, we have assigned a value of each threat according to CRAMM [11].
(iii) *Vulnerability measurement* Assigning value to corresponding vulnerability.
(iv) *Asset rating* Identify the concerned affected asset and give them a value.
(v) *Estimate the value of risk* We can measure risk as Risk = value based on measure of (Threat, Vulnerability, Asset). After threat rating, assigning vulnerability value and asset value, we will use the table given by the CRAMM [11].

In the second step, we will prioritize the security requirement after identifying threats. Initially we have identified the measures of risk to all the threats and prioritize them based on value of risk. After finding out the high prioritized assets that are involved with the particular security requirements, we calculate the priority of security requirement just from the value of threat priority.

3.1.4 Step 4: security requirement management

As security requirement also evolve along with functional and nonfunctional requirements, it is necessary to maintain the information about traces of each security requirements and its associated attributes in this phase. The techniques for requirement management presented in [31] can be used for this activity. There are different types of traceability information that must be maintained for the management of security requirements.

3.2 Security design engineering

This phase deals with designing a software structure that realizes the specification. So depending upon the identified security requirements, we identify the cryptographic services to mitigate the identified security threat of the system. Bad decisions made during the design phase can lead to design flaws that can leave the system vulnerable to security threats, so we focus on the design phase through a set of systematic design activities mainly identification of cryptographic services, design structuring and finally design decisions. The steps of this process are explained below.

3.2.1 Step 1: mapping of security requirements with security services

After the security requirements have been identified, we proceed to the design phase of the security engineering process i.e. prioritized security requirements are mapped with security services like confidentiality, integrity, authentication and non-repudiation services. The different types of security requirements proposed by Firesmith [2] are mapped to the different security services provided by cryptography. This would eventually help in the later stages of the design process, by specifying which cryptographic techniques would be suitable in a particular scenario. After the security services have been identified for the particular security requirement, we proceed to the next activity i.e. security design analysis.

3.2.2 Step 2: security design analysis

This step will define what the prioritized threats are and which assets are affected by these threats. This step consists of two sub steps as explained below:

(2a) Mapping of the Prioritized threats with related attacks. In security requirement prioritizing process we indentify different threats and prioritized the security requirements according to threat analysis. Now in this step, we first identify the threats which affect the assets with high value. Then we identify what type of attacks can be caused by these threats.

Before developing this framework, we have done a literature survey [27, 32–34] for all security mechanisms. From this knowledge based repository we will map the security requirements with their related attacks. Suppose for authentication requirements most affected assets are smart card information, login information, account information. Now the mapping Table 4 shows how the identified threat and attacks (highlighted in italics) are related with security requirements.

(2b) Mapping of attacks to security mechanism (cryptographic techniques). In this step we map the security attacks with the available techniques of cryptography and calculate the impact of these attacks. From our literature survey, the security analysis result is shown in Table 5, for a password based authentication in wireless network.

Applicability of the attacks on an algorithm shows that whether the algorithm resists the attack (if resists then applicability 'N' else 'Y') or not. For example, AES does not resist the attack UA_1 (Man-in-the-middle attack), thus in Table 5, we have marked it as 'Y' that means this attack can be applicable for AES when it is used in a password based authentication in wireless network. In this mapping table we also calculate the total impact which will help us to find out a set of algorithms which can protect the assets.

In this step the total impact shows the consequence of the attacks, that means if an attack is not applicable on an algorithm (in Table 5 for 'N'), then the impact of the attack is zero. Accordingly we calculate the impact of all ten identified attacks for an authentication scheme which can be based on any one of the above specified algorithm. On the next step we will consider the design constraints to choose a particular security algorithm which is best for a particular environment.

3.2.3 Step 3: identifying security design constraints

A very common cause of protocol failure is that the environment changes, so that assumptions that were originally true no longer hold and the security protocols cannot cope with new environment. A security environment describes the context in which the software is expected to evolve. The environment affects the kind of threats the application is likely to encounter. This is only because each environment has some design constraints. So before design structuring, first we have to find design constraints.

The Environmental Constraints of the target deployment system is considered here depending on whether the system would be implemented on a wireless/mobile/mobile ad hoc environment. For a web based system in wireless network, there are many communicational constraints (like channel capacity, bandwidth, power, through put etc.) and computational constraints (like memory, encryption speed, energy etc.). Suppose some important information is transmitted

Table 4 Mapping of the prioritized threats with related attacks

Threat	Asset effected	Related attack
T.denial_of_service	Smart card information	*Man-in-the-middle Attack (UA_1)*, Denial of Service (UA_2), Replay attack (UA_3), Perfect forward secrecy (UA_4), *Impersonation Attack (UA_5)*, *Server Spoofing Attack (UA_6)*, *Dictionary Attack (UA_7)*, Stolen verifier Attack (UA_8), *Smart Card Loss Attack (UA_9)*, Insertion Attack(UA_{10})
T.pin_stolcn	Login information	*Man-in-the-middle Attack (UA_1)*, *Denial of Service (UA_2)*, Replay attack (UA_3), *Perfect forward secrecy (UA_4)*, *Impersonation Attack (UA_5)*, Server Spoofing Attack(UA_6), *Dictionary Attack (UA_7)*, Stolen verifier Attack (UA_8), *Smart Card Loss Attack (UA_9)*, Insertion Attack(UA_{10})
T_impersonate	Account information	Man-in-the-middle Attack (UA_1), Denial of Service (UA_2), *Replay attack (UA_3)*, Perfect forward secrecy (UA_4), *Impersonation Attack (UA_5)*, *Server Spoofing Attack(UA_6)*, *Dictionary Attack (UA_7)*, Stolen verifier Attack (UA_8), *Smart Card Loss Attack (UA_9)*, Insertion Attack(UA_{10})

Table 5 Applicability of attacks

Attack	Asymmetric algorithms			Symmetric algorithms			Hashing algorithms		Signature algorithms		
	RSA (Tl)	ECC (T2)	HECC (T3)	AES (T4)	DES (T5)	Triple DES (T6)	MD5 (T7)	SHA1 (T8)	RSA + DSA (T9)	ECDSA (T10)	HECDSA (T11)
UA_1	Y	N	N	Y	Y	Y	N	N	N	N	N
UA_7	Y	N	N	Y	Y	Y	N	N	N	N	N
UA_5	Y	N	N	Y	Y	Y	N	N	N	Y	Y
UA_8	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y
UA_9	Y	Y	Y	Y	N	Y	N	N	Y	Y	Y
UA_6	Y	N	N	Y	N	N	N	N	N	N	N
UA_3	Y	N	N	Y	N	N	N	N	N	N	N
UA_4	Y	N	N	Y	Y	Y	N	N	N	N	N
UA_2	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y
UA_{10}	N	N	N	Y	Y	Y	N	N	Y	Y	Y
Total impact	8	2	2	9	6	7	2	2	4	5	5

from a mobile phone in a mobile network. One of the design constraint (encryption speed) here plays an important role for selection of cryptographic technique. While comparing the encryption speed of symmetric/asymmetric ciphers (shown in Tables 1, 2) on that device, some of the suitable crypto techniques are DES, 3DES, CAST, RSA etc. Now the design attributes will help us to select the best suitable technique.

3.2.4 Step 4: security design structuring

In this activity, different design attributes are identified which affects the selection of cryptographic protocols. The sub steps are explained below:

(4a) Identifying design attributes and prioritizing them. While identifying them we have to first look whether the system would be implemented on a wireless/mobile/mobile ad hoc or any other environment. The design attributes like cost, implementation platform etc. greatly affect our design choices because only a subset of cryptographic algorithms can work efficiently on constrained environments. Further, cryptographic algorithm would differ depending upon the service requirement. For example, symmetric key algorithms like AES, 3DES would be more suitable for confidentiality service requirement as they are 1000 times faster than asymmetric key algorithms like RSA which are less efficient for large plain text encryption. But asymmetric key algorithm like ECC is more suitable in constrained devices with limited memory/processing power/energy etc. for its short key size. Mainly we separate the design attributes (listed in Table 6) on the basis of the devices used because their priorities are different for High-end and Low-end devices [34].

(4b) Preparation of security design template (SDT). After the security requirement and threats have been identified in the requirement phase and security services and design attributes identified in the first phase of the design process, we proceed with the next step in which a security design template (SDT) is prepared to take care of each security requirement. A design template is shown in next section. This template will store each specification of the design constraints and design attributes of a particular environment for further processing.

Table 6 List of design attributes and their priorities

High-end devices		Low-end devices	
High priority	Low priority	High priority	Low priority
Scalability	Cost	Security	Interoperability
Security	Power consumption	Complexity	Flexibility
Interoperability	Complexity	Power consumption	Algorithm agility
System architecture		Cost	Scalability
Target platform		System architecture	
Algorithm agility		Target platform	
Flexibility		HW/SW	
HW/SW			

3.2.5 Step 5: security design decision

This is the final step through which we can come to the design decisions. We propose knowledge based approach to judge the best suitable security protocol depending upon the design attributes supplied in the Security Design Template. We have stored various cryptographic techniques and their analytic attributes in the repository. After impact analysis a decision is made for choosing the optimum security protocol for system under consideration. A detailed survey has been made for collecting data of different security protocols available in literature. These templates are used to access different data for selection process in a systematic manner. In the next section, we explain how these steps will be applicable in a particular environment (smart card based banking system).

3.3 Security implementation

In this phase all functionalities are implemented incorporating design decisions of the system. This includes implementing specific techniques that are suggested in the design phase of the security engineering process. We have elaborated it with an example in subsequent Sects. 4 and 5.

3.4 Security testing

It involves evaluating the system security and determining the adequacy of security mechanisms, assurances and other properties to enforce system security policies. In this phase different tasks like vulnerability scan, vulnerability assessment, security assessment, security audit and review etc. are performed which can be done in future work.

4 Case study of web based banking system

In this section, we implement the framework in web based banking system for specifying what cryptographic protocol is best to mitigate the identified security threat. In web based banking, the direct actors/stakeholders can be Customer, Bank Employee, and Bank Database while the indirect actors can be System Administrator, Maintenance Manager. Existing authentication methodologies in such banking system follows password based authentication using a smart card.

Now applying the proposed framework, we first identify the security requirements as shown in Table 7.

Here we apply Viewpoint Oriented Security Requirement Engineering Process (VOSREP) [29] for security requirements elicitation. Once the threats have been identified we can define security requirements to mitigate these threats as shown in Table 7 below.

After identifying the threats, we can prioritize security requirements considering various assets affected as discussed in our framework. Table 8 shows the SR prioritization in web based banking system. We map security requirements with prioritized threats. Here threat rating is the value which defines the probability of occurrence of a particular threat in a project. The identified threats have been assigned a value according to CRAMM [11].

Now we come to the Security Design Engineering phase.

Table 7 Web based banking system using VOSREP

Viewpoints	Services	Non-functional requirements	Threats	Security requirements
Customer	Balance inquiry	Reliability	T. Impersonate	Authorization requirement
	Transaction information	Performance	T Data_Theft	Privacy requirement
	Fund transfer	Supportability	T. Disclose_Data	Nonrepudiation requirement
	Cash management		T PIN_Violated	
	Bill payment		T.Change_Data	
			T.Certificate_steal	
			T.Denial_of_service	
Bank Employee	Update Account	Correctness and consistency of information	T.Unauthorize_access	Integrity requirements.
	Transaction processing		T.Change_Data	Authentication requirements
	User query process	Minimize response time in accessing account	T.PIN_Violated	
	Report generation		T. System_Failure	Identification requirements
	Maintenance	Usability	T. Outsider	
Bank Database	Maintain reports of the database	Reliability	T.Change_Data	Authentication requirement
	Recovery system	Integrity	T.Copy	Security auditing requirements
		Recovery	T_Replace	
		Performance		Intrusion detection

Table 8 SR prioritize for web based banking system

Security requirement	Threat	Threat rating	Vulnerability	Assets affected	Asset value	Risk	SR prioritize
Authentication	T.PIN_Violated	3.5	.1	User login info(l,2)	4	3,4	7
	T.Denial_of_service	.1	.5	Smart card info(l,2,3)	9	3,4,5	12
	T.Impersonate	.34	.5	Account info(1,3)	6	4,5	9
Authorization	T.Change_Data	3.35	.5	Certificate info(2)	9	6	6
	T.Certificate_steal	1	.5	Account info(1,2)	6	6,4	10
Privacy	T.PIN_Violated	3.5	.5	User login info(l,2)	4	3,4	7
	T.Unauthorize_access	.1	.5	Account info(l,2)	6	4,5	9
Integrity	T_replace	1	.1	Customers info (l,2)	5	5,6	11
	T.Data_Theft	1	.1	Transaction info (l,2)	8	5,6	
Non-repudiation	T.Certificate_steal	3.33	.5	Customers infof (1)	5	4	4

4.1 Step 1: mapping of security requirements with security services

Here we find out what security services need for each security requirements. Mapping process is shown in Table 9 below.

4.2 Step 2: security design analysis

For mapping the prioritized threats with attacks, it is essential to find out the assets that are affected by the identified threats. Now we list out what are the possible attacks that can be caused by these threats which are already discussed in Sect. 3. We map these attacks with the most affected assets in Table 10 below.

Applicability of the attacks on assets shows that whether the asset is affected by the attack (if affected then applicability 'Y' else 'N') or not. Now we mapped these attacks with security requirements shown in Table 11. We calculate average impact for each security requirements and the values are compared with the values of Table 5. Here we consider only lower range values.

This table shows that for fulfillment of all the security requirements, the suitable cryptographic algorithms are RSA + DSA (T9), ECDSA (T10), HECDSA (T11) because these are the common techniques for each case. It implies that all the security requirements will be achieved by public key cryptography. Now the next step will help us to choose a particular algorithm.

Table 9 Mapping of security requirements with security services

Security services	Security requirements
Confidentiality	Privacy requirements
Authentication	Authentication requirements
	Authorization requirement
Non-Repudiation	Identification requirement
	Non-repudiation requirement
Integrity	Integrity requirement

4.3 Step 3: security design constraints

In this phase we will consider the environmental constrains which affect the design decisions. The application environment depends on different communicational constraints (like channel capacity, bandwidth, power, through put etc.) and computational constraints (like memory, encryption

Table 10 Mapping the applicability of attacks with assets

Assets affected attack	User login info()	Smart card info()	Account info()	Certificate info()	Customers info()	Transaction info()
UA_1	Y	Y	N	Y	Y	Y
UA_7	Y	N	Y	Y	Y	Y
UA_5	Y	Y	Y	Y	Y	Y
UA_8	N	N	N	N	N	N
UA_9	Y	N	Y	N	N	Y
UA_6	N	Y	Y	N	N	Y
UA_3	N	N	Y	N	N	N
UA_4	Y	Y	N	N	N	N
UA_2	Y	N	N	N	N	Y
UA_{10}	N	N	N	Y	Y	N

Table 11 Mapping of attacks to security mechanism (cryptographic protocol)

Security requirement	Threat	Assets affected	Attack	Impact analysis (Avg. impact)	Suitable algorithms
Authentication	T.PIN_Violated	User login info (1,2)	UA_1, UA_9, UA_2, UA_7, UA_5, UA_4	15/3 = 5	T10, T11
	T.Denial_of_service	SmartCard info (1,2,3)	UA_1, UA_6, UA_4, UA_5		
	T.Impersonate	Account info (l,3)	UA_7, UA_6, UA_3, UA_9, UA_5	9/2 = 4	T9, T10, T11
Authorization	T.Change_Data	Certificate info (2)	UA_7, UA_1, UA_{10}, UA_5		
	T.Certificate_steal	Account info (1,2)	UA_7, UA_6, UA_3, UA_9, UA_5		
Privacy	T.PIN_Violated	User login info (l,2)	UA_1, UA_6, UA_3, UA_7, UA_5, UA_8	11/2 = 5.5	T10, T11
	T.Unauthorize_access	Account info (l,2)	UA_7, UA_6, UA_3, UA_9, UA_5		
Integrity	T.Replace	Customers info (1.2)	UA_7, UA_1, UA_{10}, UA_5	10/2 = 5	T9, T10, T11
	T.Data_Theft	Transaction info (l,2)	UA_7, UA_1, UA_6, UA_9, UA_2, UA_5		
Non-Repudiation	T.Certificate steal	Customers info (l)	UA_7, UA_1, UA_{10}, UA_5	4	T9, T10, T11

speed, energy etc.). If we choose public key cryptography for low end devices (nodes in sensor networks), it will increase unnecessary cost. But for high end devices (used in Banking, Defenses) where the security is the ultimate concern, public key techniques are preferable for its small key size and high security. In this environment, Elliptic Curve Cryptography (ECC) is one of the best public key techniques for its small key size and high security in smart card based authentication.

4.4 Step 4: security design structuring

Now all the design attributes and the priorities are assigned to prepare SDT. Assuming the devices used in WLAN are all high end devices and their design attributes are given in Table 6, we prepare the template as shown in Table 12 below.

4.5 Step 5: security design decisions

If we want to consider some public key algorithm in this environment, the design decision should depend upon which technique among various available techniques can be considered. Here we consider only encryption speed as design constraint as shown in the result of Table 13 below. The decision template provides necessary data of different public key algorithm which help us to choose a suitable cryptographic technique in a particular environment.

Final decision template shows that ECDSA (160 bit key) is the most suitable technique which fulfills all the security requirements for web based banking. We implemented some of the most common cryptographic algorithms using Crypto++ library. All were coded in C++ and ran on Intel Core 2DUO CPU T6400@2.00 GHz PC with 4 GB RAM and windows vista operating system. We store the result data (encryption/decryption speed, no of iterations etc.) in a

Table 12 Table shows SDT for web based banking

Environment	Design Constraint	Design Attribute	Priority: High (Value 1.0)	Priority: Medium (Value 0.5)	Priority: Low (Value 0.1)	Security Mechanisms
LAN ☐	Encryption	Throughput	☑	☐	☐	Public keyEncryption ☐
WLAN ☑	Speed ☑	Storage	☐	☑	☐	Publickey Encryption With signature ☑
WPAN ☐	Range ☐	Target platform	☑	☐	☐	Symmetric Encryption(BlockCiphers) ☐
WWAN ☐	Channel	Cost	☐	☐	☑	Symmetric Encryption(StreamCiphers) ☐
WMN ☐	Capacity ☐	Power consumption	☐	☐	☑	Symmetric Encryption with signature ☐
WSN ☐	Frequency ☐	Security	☑	☐	☐	Systemsecuritytechniques ☐
MANET ☐	Bandwidth ☐	Scalability	☑	☐	☐	Challenge-response techniques ☐
		Flexibility	☑	☐	☐	Digital Signatures ☐
		Algorithm agility	☐	☑	☐	Hash based authentication ☐
		Complexity	☐	☑	☐	MAC based authentication ☐
		Interoperability	☑	☐	☐	

Table 13 Decision template for web based banking system

Environment	Design Constraints	Design Attribute	Priority			ECC operation on GF(p)		
			High (Value 1.0)	Medium (Value 0.5)	Low (Value 0.1)	Operation	Iterations	Total Time
LAN ☐	Encryption ☑	Throughput	☑	☐	☐	168 bit Encryption	368	30.1
WLAN ☑	Speed ☑	Storage	☐	☑	☐			
WPAN ☐	Range ☐	Target platform	☑	☐	☐	168 bit Decryption	736	30
WWAN ☐	Channel Capacity ☐	Cost	☐	☐	☑	168 bit Signature	735	30
WMN ☐	Frequency ☐	Power consumption	☐	☐	☑	168 bit Verification	606	30
WSN ☐	Bandwidth ☐	Security	☑	☐	☐	155 bit Encryption	417	30
MANET ☐		Scalability	☑	☐	☐	155 bit Decryption	822	30
		Flexibility	☑	☐	☐	155 bit Signature	835	30
		Algorithm agility	☐	☑	☐	155 bit Verification	689	30
		Complexity	☐	☑	☐			
		Interoperability	☑	☐	☐			

repository and find the most suitable data using searching algorithm.

5 Implementation and evaluation of the framework

The framework which we described in the previous section is used to construct a tool which we call the Security Design Engine (SDE). This tool can be used to judge the best suitable security technique depending upon the design attributes supplied in the 'Design Template'. To develop this tool we use Microsoft.NET framework that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. This design tool consists of four parts to cover all four steps of security engineering process. The screen shots are shown in Fig. 2a and b which is the main screen of our design tool. Here we mainly focus on the design module.

For a particular environment, suppose in WLAN environment we want to develop software for web based banking system. Now for secure transaction of every detail of customers by the systems which is operated by a central server, a security module will be introduced to fulfill the security requirements as shown in Fig. 2b.

After specifying the security requirements and design constraints, the tool will help us to take the final decision of selection of suitable security technique for that particular environment. In Fig. 3a, we set the values of the design attributes. After performing the operations specified in Sect. 4, the final design decisions will come as shown in Fig. 3b.

6 Performance evaluation

Now we will evaluate our framework with the other models available in the literature such as AOD [12], PICO [13] in different phases of SDLC. Our evaluation parameters are different phases of SDLC such as security requirements analysis, security design and implementation. In security requirement engineering phase, we have (i) elicited security requirements using viewpoint oriented approach, (ii) specified different type of security requirements such as authentication requirements, privacy requirements etc., (iii) prioritized these security requirements using techniques of CRAMM. Prioritization helps us to focus a vital security issue in case of conflicting requirements and implement security in optimal manner. Whereas other models like PICO and AOD have elicited security requirement as threat model or attack tree, they have not specified and prioritized them.

In security design engineering phase, we convert security requirements and threats (which are identified during the security requirement elicitation stage) into design decisions to mitigate the identified security threats. We adopt a structural approach for design engineering. Our decisions are based on analysis of environment constraints, design attributes and possible attacks on threats. Our approach generates a template based on this analysis to enable security engineer to choose optimal cryptographic services. This chosen security mechanism is just appropriate to mitigate possible attack based on environment. Hence it not only mitigates the threats/attacks, but also does not unnecessarily constraint the availability of the system. But other models like PICO and AOD etc. take ad hoc design decisions without considering design attributes and other factors.

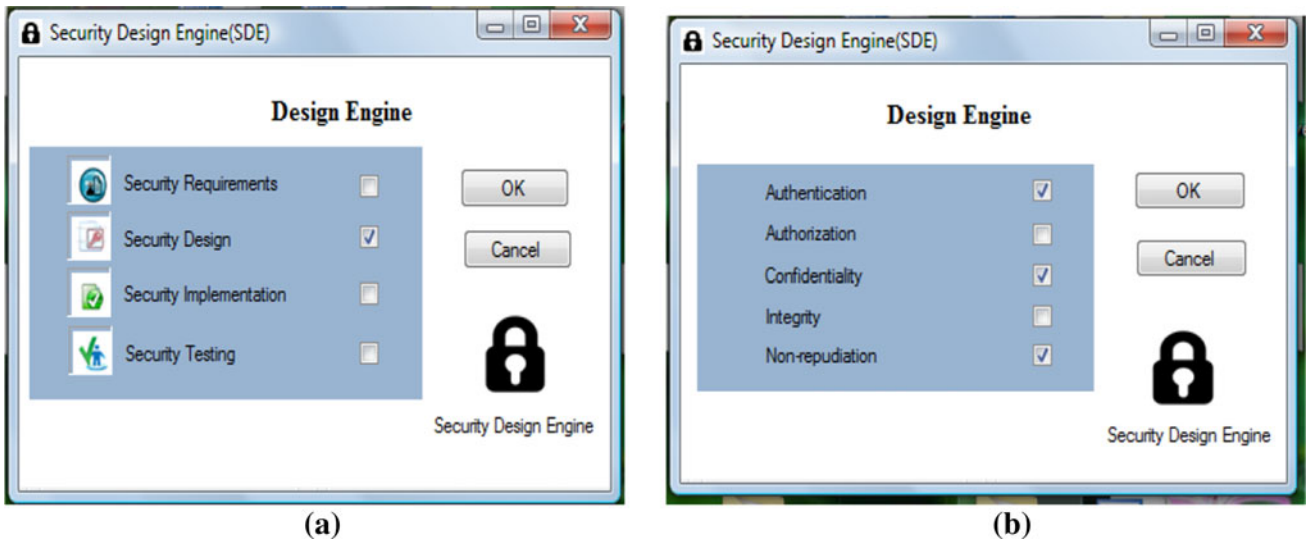

Fig. 2 **a** Security design engine, **b** Selection of security requirements for a particular environment

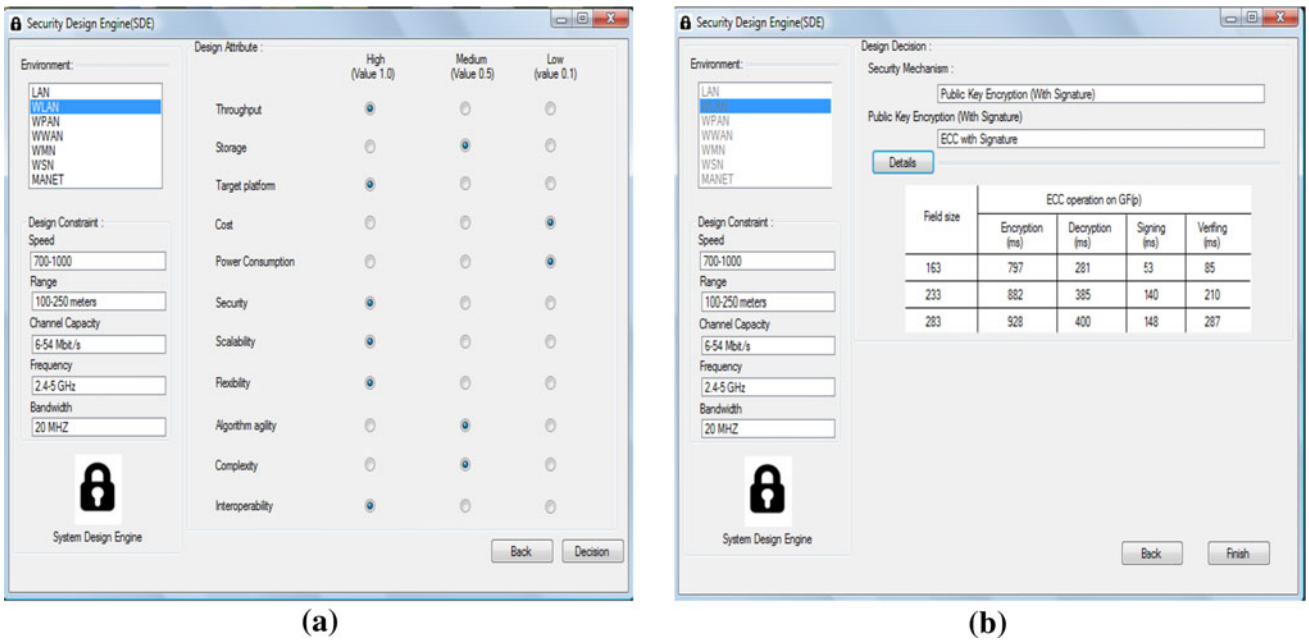

Fig. 3 **a** Selection of design attributes for a particular environment, **b** Design decision for above specified environment

The advantage of these design decisions are reflected in our implementation mechanism. While implementing design decisions, we have considered parameters like key size, encryption/decryption time. In this phase also other models have taken no specific approach considering the above parameters. We now summarize the comparison in Table 14.

In our framework, we focus on design decisions and for this we have done all the steps like mapping of prioritized threats with attacks, identifying design constraints in a particular environment which other methods have not yet considered. While selecting a key size for secure data communication, we are able to select optimum key size for a particular application as compared to PICO [13]. PICO relies on the well known SSH. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet. We perform key length comparison with PICO as shown in Table 15.

The above comparisons clearly state that our framework is a better approach to find a suitable design decision to develop efficient cost effective secure system.

Table 14 Comparing models for first three phases of secure SDLC

Model	Secure software development life cycle (first three phases)					
	Security requirements engineering		Security design engineering		Security implementation	
	Elicitation techniques	Requirements evaluation	Design decisions	Attack evaluation	Security techniques	Performance evaluation
AOD [12]	Static role models (SRM) are used for finding requirement elicitation process.	Nil	Design decisions are taken using weaving strategy without considering design constraints.	Yes	No specific approach is taken.	Nil
PICO [13]	Eliciting security requirements as a form of threat models.	Yes	Design decisions are as a form of security patterns without considering design constraints, other factors.	Nil	No specific approach is taken.	Only for SSH by comparing key length.
Secure-Banking Model [28]	Eliciting security requirements using attack tree.	Nil	Ad hoc design decisions are taken. No proper design steps are found.	Yes	General approach is considered. It is not based on attack evaluation.	Nil
Our framework (SDE)	Eliciting security requirements using view point oriented approach.	Requirements are evaluated based on risk measure and also security requirements are prioritized.	Provide framework for environment specific proper design decisions.	Attack evaluation based on mapping of prioritized threats with attacks, identifying design attributes and constraints of a particular environment.	Tool based approach considering environment specific design attributes.	Evaluation is done using parameters like key size, encryption/ decryption speed etc.

Table 15 Comparing key lengths with PICO

	Symmetric key algorithm	Asymmetric key algorithm	Hybrid algorithm	Message digest size (bits)
Recommended Size	76 bits	1279 bits	No	152
PICO [13] for SSH	128 bits	1024 bits	No	160/128
SDE for environment specific	56 bits [WPAN]	80bits [WMAN]	160bits [MANET]	160/128

7 Conclusion

We have developed a Framework for Security Engineering Process with a strong focus on security design engineering. We insert security concerns in each step of the life cycle. In our framework the different types of security requirements are mapped to the different security services. The identified design attributes are prioritized and a security design template is prepared to find out the specific cryptographic techniques in a particular scenario. This framework is helpful to the developers to identify and implement the appropriate cryptographic technique efficiently in a particular environment. This process is coherent with the conventional software engineering process so that eliciting security requirements and security design become an integral part of system engineering and security engineering.

References

1. Fabian B, Gurses S, Heisel M, Santen T, Schmidt H (2010) A comparison of security requirement engineering methods. Requir Eng 15:7–40

2. Firesmith DonaldG (2003) Engineering security requirements. J Object Technol 2(1):53–68
3. Dermott JM and Fox C (1999) Using abuse case models for security requirements analysis. Department of Computer Science, James Madison University, pp 55–64
4. Alexander IF (2003) Misuse cases, use cases with hostile intent". IEEE Software, pp. 58–66
5. Guttorm S, Opdahl AL (2005) Eliciting security requirements with misuse cases. Requir Eng 10:34–44
6. Ware M, Bowles J, Eastman C (2006) Using the common criteria to elicit security requirements with use cases. IEEE Computer Society
7. Ellison RJ (2005) Attack trees. Software Engineering Institute, Carnegie Mellon University
8. EBIOS (2004) Expression of need and identification of security objectives. DCSSI, France
9. Alberts C, Dorofee A (2001) OCTAVE Method Implementation Guide v2.0. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://www.cert.org/octave
10. CORAS: A Platform for risk analysis of security critical systems. IST-2000-25031. http://www.nr.no/coras/
11. The logic behind CRAMM's assessment of measures of risk and determination of appropriate countermeasures, www.cramm.com. Accessed 23 April 2007
12. Georg G, Ray I, France R (2002) Using aspects to design a secure system, ICECCS. IEEE Computer Society, Washington, pp. 117–126
13. Apvrille A, Pourzadi M (2005) Secure software development by example. IEEE Secur Priv 3(4):10–17
14. Lipner S, Howard M, (2005) The trustworthy computing security development lifecycle" security engineering and communications, security business and technology unit, Microsoft Corporation
15. Giorgini P, Manson G, Mouratidis H, Philip I (2002) A natural extension of tropos methodology for modelling security. Workshop on Agent-oriented methodologies, OOPSLA
16. Van Lamsweerde A (2004) Elaborating security requirements by construction of intentional anti-models, ICSE'04. IEEE Computer Society, Washington, pp 148–157
17. Mayer N, Heymans P, Matulevičius R (2007) Design of a modelling language for information system security risk management, RCIS, pp 1–11
18. Haley CB, Laney R, Moffett JD (2008) Security requirements engineering: a framework for representation and analysis. IEEE Trans Software Eng 34(1):133–153
19. Gupta D, Agarwal A (2008) Security requirement elicitation using view points for online system. International conference on emerging trends in engineering and technology, IEEE Computer Society, pp 1238–1243
20. Gupta D, Jaiswal S (2009) Security requirement prioritization, Software Engineering Research and Practice, pp 673–679
21. Chatterjee K, Gupta D, De A (2011) A framework for security design engineering process. In proceedings of fifth international conference on information processing, ICIP, Springer Verlag Berlin Heidelberg, CCIS 157, pp 287–293
22. Jadhav AS, Sonar RM (2011) Framework for evaluation and selection of the software packages: a hybrid knowledge based system approach. J Syst Softw 84(8):1394–1407
23. Bell DE, La Padula LJ (1973) Secure computer systems: Vol. I—mathematical foundations, Vol. II—a mathematical model, Vol. III—a refinement of the mathematical model. Technical Report MTR-2547 (three volumes), Mitre Corporation, Bedford, MA, March–December
24. Anderson RJ (1996) Security engineering, Wiley
25. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48:203–209
26. Koblitz N (1989) Hyperelliptic cryptosystems. J Cryptol 1(3):139–150
27. Stallings W (2003) Cryptography and network security principles and practices, 3rd edn. Pearson Education, Upper Saddle River
28. Dimitriadis CK (2007) Analyzing the security of internet banking authentication mechanisms. Information systems control journal 3:1–8
29. Sommerville I (2003) Software engineering. Pearson Education, London. ISBN 8129708671
30. Common Criteria for Information Technology Security Evaluation (2006) Version 3.1 http://www.Commoncriteriaportal.Org/public/expert
31. Kotonya G, Sommerville I (1995) Requirement engineering with viewpoints
32. Schneier B (1996) Applied cryptography, Wiley
33. Hiltgen A, Kramp T, Weigold T (2005) Secure internet banking authentication. IEEE Secur priv 1540(7993):24–32
34. Hankerson Darrel, Menezes Alfred, Vanstone Scott (2004) Guide to elliptic curve cryptography. Springer, New York. ISBN 0-387-95273-X