VIENNA UNIVERSITY OF TECHNOLOGY

Institut für Technische Informatik
Technical Report 182-1/2007/70

Embedded Security at a Glance: Security Concepts for Embedded Systems

Armin Wasicek

TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

This page intentionally blank.

Armin Wasicek
Institute for Computer Engineering
Treitlstrasse 3/3
1040 Vienna
Austria

tel.:+43 1 58801 18211
email: armin@vmars.tuwien.ac.at

Abstract

Information security is gaining more and better attention from the embedded systems community. It is now widely acknowledged that security and safety are intrinsically tied and may not be torn apart during the design process. embedded systems often sustain a critical infrastructure which is exposed to accidental as well as malicious faults. Acts of vandalism, terrorism, sabotage, or crime pose serious threats to a system's correct operation. In the face of pervasive computing embedded systems play a major role in distributing the computational power of modern microprocessors to business, transportation, governments, public space, and even households. Besides the benefits of this progress, the reverse side is that people may have different intentions and motivations to use the systems. By specifying security threats and counter measures during the system design it must be guaranteed that the embedded system may be utilized only in the way the designer intended, the user requires, and within the boundaries of regulations and legal obligations of the deployment area This report gives an introduction to information security under the aspect of embedded systems. It explains some general security measures, summarizes cryptography and trusted computing, and points out the concepts of intrusion tolerance.

Keywords: Security, Embedded Systems, cryptography, Trusted Computing, intrusion tolerance.

Contents

List of Figures

List of Tables

1 Introduction

With the raise of information processing technologies, digital data processing is penetrating many areas of everyday life and carrying out crucial duties and responsibilities. To guarantee the smooth operation, safety and security measures must deliver a correct and sound service. The importance and demand for information security is increasing equally to the demand for digital control.

In contrast to general–purpose computers, embedded systems are designed to perform a certain task. The term 'embedded' implies that their operation is transparent for the user, who can be completely unaware of their deployment. Embedded systems emerge from the field of control engineering and are now deployed in many areas like transportation, industry, communication, economy, infrastructure, etc. In order to achieve their goal, embedded systems have to act in an *intelligent* manner.

An IT system's *asset* is the value of its service to its users. This can be, e.g., a database serving some precious information. To protect this asset the database's information must be kept secret, thus, it requires *confidentiality*. With embedded systems this is different, because the information has usually either short lifetime before it is consumed, or can be collected by everyone from the environment. The important thing is the functioning of the embedded system. Thus, an embedded system's asset is to deliver an efficient and dependable service. This poses requirements on *availability* and *integrity*. Together, these three attributes form *security* [85].

Until today the embedded system design process focuses on achieving the attributes associated with dependability, but neglects the demand for security. Recently, the security requirements of embedded systems started to gain more attention by the scientific community. This is due to the achievements of pervasive computing, e.g., in the western world nearly everybody is carrying a cellular phone and thus a computer, the ambitious efforts to establish a digital marketplace, the emerging of an infotainment culture, ...Governments grasp embedded systems as a means to enforce regulations and legal obligations, e.g., the digital tachograph system, public surveillance, ...In the industrial field, a unifying network will merge enterprise level, information level, control level, and field level networks, catchword IP *instrumentation* [58]. Among many more, these topics open new challenges and raise new issues concerning security.

This report investigates the state–of–the–art of information security in embedded systems. It highlights some current threat scenarios and lists countermeasures based on the special properties of embedded systems. It gives an introduction to cryptography, its security properties, and its computation on limited devices. Furthermore, the concepts of trusted computing are discussed and the physical security properties of embedded devices are presented. Next, the concept of intrusion tolerance is explained and, finally, some ideas on the economics of security can be found the last section.

1.1 Definitions of Security

The term 'information security' *means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide three core principles: confidentiality, integrity and availability."*[1]

- *Confidentiality*: The assets are accessible for reading, copying, locating only by authorized parties. If secrecy is not maintained, the computer system is susceptible to unauthorized disclosure of data or unauthorized access to its programs.

- *Integrity*: The modification (writing, changing, changing status, deleting creating) of an asset requires authorization. The integrity requirement does not hold, when an unauthorized user or program may modify data or damage the system.

- *Availability*: Authorized parties can access the assets in the manner specified and during the periods specified. Lack of availability results in a *denial–of–service*.

In Figure 1 Avižienis et al. [85] give a definition of security and dependability in the light of their shared and distinct attributes. Security is the concurrent existence of availability for authorized users only, confidentiality, and integrity regarding unauthorized manipulations of the system state. *Authorization* is the right or permission to use a system resource. A *security policy* is a definition what is secure and thus admissible, and which behavior is considered insecure and therefore prohibited.

Secondary attributes to security are composites of primary ones, i.e., they share the properties of two or more primary attributes to a certain degree. They include:

- *Accountability*: availability and integrity of the person who performed an action.

[1]U.S. code collection, Title 44, Chapter 35, Subchapter III, Â§ 3542

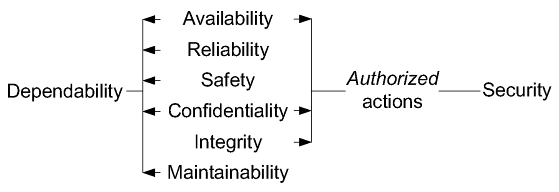

Figure 1: *Relationship between dependability and security after Avižienis et al.*

- *Authenticity*: integrity of a message content and origin, eventually including other types of information, e.g., time.
- *Non–Repudiability*: availability and integrity of the identity of a participant in the communication. The goal is it to provide irrefutable proof of an action in the system to a third party.

1.2 Terminology

A basic terminology for security and dependability was developed during the MAFTIA project [66]. The cited document relates security breaches to the fault model described in [85]. It defines the causal triple fault–error–failure as follows:

- *fault*: adjudged or hypothesized cause of an *error*.
- *error*: part of the system state which may cause a subsequent *failure*.
- *failure*: occurs when the error reaches the service interface and the delivered service deviates from implementing its function.

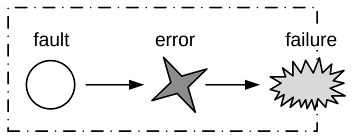

Figure 2: *Fault model after [85].*

This causality chain is as well depicted in Figure 3. The dotted line demarcates the fault containment region FCR, which is defined as *"the set of subsystems that share one or more common resources and may be affected by a single fault"* [52].

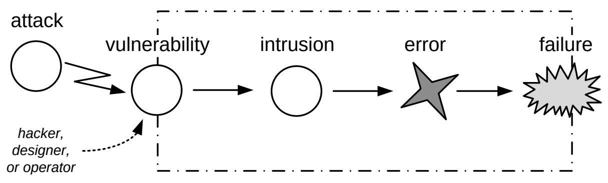

Figure 3: *Intrusion model after [66].*

An *attack* is is an attempt to to exploit a weakness or *vulnerability* in the system in order to perform an unauthorized action. A succesful attack attempt results in an *intrusion*. Figure 4 illustrates this relationship. It further suggests that a vulnerability can be introduced during development or operation by hackers, operators, and designers.

- *attack*: a malicious interaction fault respective an intrusion attempt which aims at deliberately violating one or more security properties.
- *vulnerability*: a fault deployed during operation or system design that can be exploited to create an intrusion.
- *intrusion*: an externally–induced fault resulting from an attack that can be used to alter the system state.

The *intrusion–tolerance* paradigm is introduced in [80]. It assumes that systems remain to a certain extent vulnerable and that attacks on components or sub-systems will happen and some will be successful. Its goal is it to ensures that the overall system nevertheless remains secure and operational, with a measurable probability. This is achieved by error processing mechanisms that make sure that a security failure is prevented.

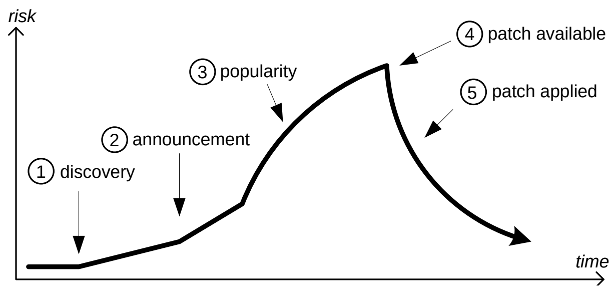

Figure 4: *Vulnerability life cycle after Schneier [72].*

Schneier developed a *vulnerability life cycle* in [72]. Figure 4 depicts a chart divided in five phases: In phase 1 the vulnerability have not been discovered yet and it is dormant. Next, in phase 2 one or more persons discovered the vulnerability and know how to make use of it, but no countermeasures exist. This phase poses the highest risk for a system, because nobody but the potential attacker knows about the vulnerability. At some point in time, it will be announced in phase 3, either through some bug report, scientific publication, exploit code, etc. More people know about it and the attack might gain popularity. In the fourth phase automatic attack tools get available and reduce the technical skills to launch an attack. Now it should be fairly easy and the number of attack jumps up. Finally, a patch is available and installed by the users and the attack rate decreases again in phase 5.

2 Embedded Systems Security

Three factors – also called the *Trinity of Trouble* – were identified to be the major source of vulnerabilities [37, 48].

- *Complexity*: Modern software is a complex clockwork of interacting components. With increasing functionality the code size grows as well. This increments the likelihood of bugs and vulnerabilities. Additionally, engineers program in unsafe languages like C and C++ where typing is rather weak and make use of insecure libraries which offer no protection against simple attacks like buffer–overflows [60] or dangling pointer errors.
- *Extensibility*: Today's software is not a static piece of work, but constantly evolving. Updates and bug fixes change the codebase, eliminate existing vulnerabilities, and maybe open new ones. The article in [36] discusses the implications of patching software on security. Moreover, many operating systems for embedded systems support dynamically loadable device drivers and modules. Bugs shipped with these extensions may render the whole system vulnerable.
- *Connectivity* and remote usage of embedded devices increase their usability and open an incredible variety of possible applications, but unfortunately also give rise to network induced vulnerabilities. An attacker does not have to physically possess the embedded device, but can mount a remote attack. Furthermore, failure propagation among similar devices can cause massive security breaches [51], since there are more possible targets to attack, and attacks can be remotely managed.

In the case of embedded systems, a forth factor may be added, namely the *operation in an untrusted environment*. Many embedded devices have to stay secure even under the physical possession of non–trusted parties. Consider for example a not registered workshop for cars, loss or theft. A malicious person could try to physically break the system cryptographic boundary in order to access the device without permission. When embedded systems are used to implement regulations such as, e.g., road–pricing, they provide a very worthwhile target to crack.

2.1 Attacks on Embedded Systems

The paper in [70] provides an excellent overview on attacks on embedded systems. This section gives a summary and points out the most important facts concerning this topic. Attacks can target either the design of an embedded device, hence the abstract model, or the real device thus its implementation.

Almost all known attacks on embedded systems are implementation attacks. They target to bypass or weaken the security functions. Theoretically, the *functional security mechanisms* protecting the embedded system may satisfy the security requirements. In practice, the adversary will avoid a direct approach and try to circumvent the applied security measures. Functional security mechanisms must be seen in contrast to their implementations, that offer more and new attack possibilities, which might not be obvious during system design, when security functions are considered as 'black box' building blocks. Hence, they are far from being complete security solutions and systems designed this way uncover a broad attack surface.

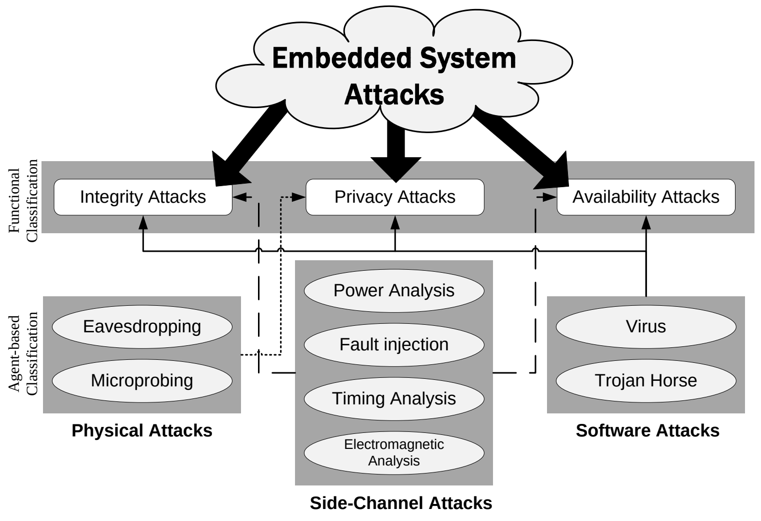

Figure 5: *Attacks on Embedded Systems after [69]*

Figure 5 is a reproduction of the threats listed in [69]. It divides attacks on embedded systems in two classes, a *functional classification* on the one hand, and an *agent–based classification* on the other hand. Functional security mechanisms support the main attributes of security, e.g., an encryption algorithm facilitates privacy, or a (secure) hash function implements an integrity measure.

Possible attacks are summarized in the second class. They are executed by an agent and can threaten one or more attributes. Again, they are broken down in three categories: *Software attacks*, which represent by far the biggest threat to all kinds of IT–systems, make use of, e.g., misconfiguration or buffer–overflows to run malicious code like viruses, worms, or trojan horses. Typically, the infrastructure for software based types of attack is much cheaper and easier to acquire compared to physical attacks.

Physical attacks and *side-channel attacks* directly access the device, the board, or the embedded chips. They are either invasive, which means they physically intrude into the device, non-invasive, like some sorts of side-channel analysis, or a combination of both. The first category encompasses types of attacks like reading out memories (microprobing) and listening to inter–component communications (eavesdropping). The second category covers simple and differential power attacks (see Section 6.3), timing attacks, fault injection attacks, and electromagnetic analysis.

2.2 Design challenges for Embedded Systems

The papers in [30, 69, 48] investigate design challenges for embedded systems. Compared to standard IT–systems some gaps can be identified which point out the reasons why applying security measures to embedded systems is a major challenge.

- The *processing gap* highlights that current embedded system architectures are not capable of keeping up with the computational demands of security processing.
- The *battery gap* emphasizes that the current energy consumption overhead of supporting security on battery constrained embedded systems is very demanding.
- The *flexibility* stresses that an embedded system is often required to execute multiple and diverse security protocols and standards.
- The *tamper resistance* emphasizes that secure embedded systems may be facing an increasing number of attacks from physical to software attacks. Side–channel attacks also represent an important threat for these systems.
- The *assurance gap* is related to reliability and stresses the fact that secure systems must continue to operate reliably despite attacks from intelligent adversaries who intentionally seek out undesirable failure modes.
- The *cost* of an embedded system increases with the security measures integrated and the extra computing power needed. Taking a design

decision in favor of security might require an additional chip and hence increase the overall system cost.

2.3 Security Applications in Embedded Systems

On the contrary to standard IT systems, which are mostly exposed only to remote attacks, an attacker of an embedded system mostly has the system physically under his control. This adds physical attack scenarios (side-channel attacks [48, 57, 75], reverse engineering, device tampering [56], etc.) to the range of possible attacks.

These scenarios in addition to the classical challenges, the "Trinity of Trouble" are the reason for the big demand for security in embedded systems. Several challenges can be identified for the near future [62]. To meet the challenges, security design must be considered right from the start.

- *Software maintenance*. Software is by nature ever–evolving and its development doesn't stand still. Because many bugs are found after shipment and requirements are changing, it has to be constantly adapted. To enable software maintenance a mechanisms to perform secure updates on an embedded device is mandatory. Updates and update mechanisms are the gate through which many malicious pieces of code try to slip on their vicious mission.

- *Theft prevention*: A device's operation can be bound to an embedded security mechanism to restrict the usage to its legal owner. A well known–example for this is the electronic immobilizer – a remote control key for opening cars and starting the engine. This challenge is often connected to the area of biometrics which aims at unambiguously identifying individuals.

- *Access control*: An embedded system's operations and data should be protected from unauthorized access to ensure their responsible usage. In many applications embedded systems perform mission–critical tasks and human lives depend on their correct operation. Thus, a protection of these systems is necessary to avert the consequences of a security breach. The worst–case assumption in this case is a nuclear power plant in the hands of a terrorist.

- *Support of new business models (*DRM*):* New business models are being introduced in the infotainment fraction. Embedded devices are used to transport the content to the consumer and must therefore protect the owner's legal rights. A whole new field of business is currently emerging from the digital world. Security measures are a stakeholder in the functioning of this new marketplace.

- *Personalization/identification*: The genuine identification of the user supports the creation of user profiles, which can be used, e.g., to configure a car's comfort functions to a driver's customized settings, or to implement a logger similar to a flight data recorder (FDR), which records the users and the utilized functions of the device.
- *Legal obligations*: Many embedded applications are built to support law enforcement, for example the European tachograph system or road pricing systems. A manipulation of such a device mostly targets the destruction of evidences and must thus be prevented.

3 Common Security Techniques

This section summarizes some common security techniques which can be found in many applications.

3.1 Authentication

An *authentication procedure* establishes a trust relation between two principals. One principal exhibits some sort of credentials which the other principal checks for validity. After a successful test the first principal is acknowledged to the second one who in return transfers some privileges to the authenticated principal.

A well–known application of authentication is access control. An embedded device is only operational after a successful authentication procedure, e.g., entering a personal identification number (PIN) when using a cash machine or after powering a cellular phone. However, a common misconception is that a computer confirms the identity during authentication. It is not possible to establish or prove an identity. The only thing which can be done is to perform a test which is considered as sufficient. Since the creation of such tests must take into account many factors like security and usability, it is not a simple task finding an adequate one.

In the following a short authentication method overview is given:

- *Kerberos (*GSSAPI*)* [49] is based on the Needham–Schroeder Symmetric Key Protocol. It includes a timestamp to fix the vulnerability of the original protocol to replay attacks. Two principals identify themselves by the use of a trusted third party.

- The SSL/TLS *handshake* [26] is responsible for the ciphersuite negotiation, the initial key exchange, and the authentication of the two peers in the SSL/TLS protocol. It uses RSA public key cryptography for authentication.

- SOCKS*v5* [55]. This is a very simple method where a username and password combination is sent in plaintext to a server in order to request its relay service.

- *Spread spectrum* TCP *(*SSTCP*), Tailgate* TCP *(*TGTCP*), and Option–Keyed* TCP *(*OKTCP*)* [11]. Also known as *port knocking*, these lightweight techniques facilitate authentication at protocol level. They make use of protocol specific fields (like the port number field in IP packets) to transmit some authentication code to request access.

3.2 Firewalls

A security policy determines a computer system's access restrictions. More specifically it states who may access what in which manner. A firewall enforces the access controls. From a system designer's point of view, a firewall is *not a single device or a group of devices, but the implementation of a security policy*.

As a perimeter firewall a firewall acts as a special gateway extending a normal gateway for communication control features. It can allow or deny inbound or outbound communication requests and thus govern access to a network. A policy defines how access is granted among users, which usually consists of several rules. Message filtering has to be performed according to these rules. In the distributed case [15], several single firewalls are grouped in a virtual private network (see Section 3.4) to execute a common policy.

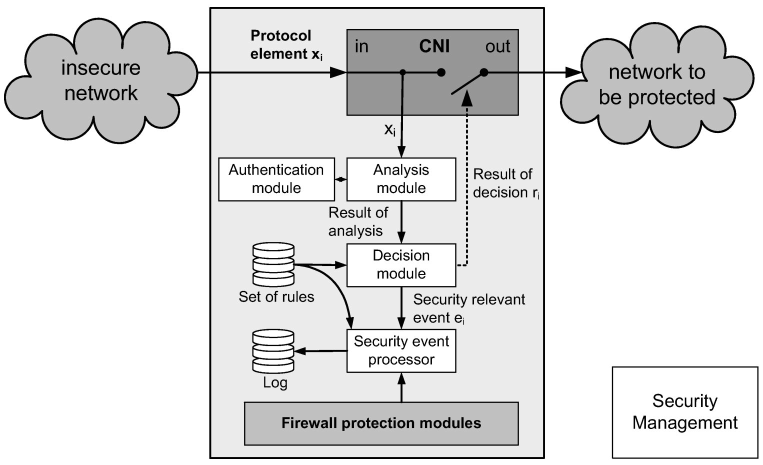

Figure 6: *Structure of a firewall element connecting a secure and an insecure network*

The basic architecture of a firewall implementation is depicted in Figure 6. A protocol element x_i arriving from an non–trusted network is tapped and forwarded to an analysis module, which checks for validity and authenticity or defrags several coherent elements. Based on a set of rules forming the security policy a decision is taken, either to accept the protocol element in the protected network or to trigger a security relevant event.

Since an access policy can change as time goes by, it is necessary to adjust the firewall rules. Changes can be triggered for example by adding a new node or a new service to the cluster, or through the occurrence of a certain system event. An interface must be provided to reconfigure the working rule set efficiently, secure, and without message loss.

Firewalls present an efficient tool to control access to a network, but they also have some restrictions, one has to be aware of following threats when deploying a firewall [10, 14]:

- *Insiders* are a major threat. A firewall can prevent users from outside the network accessing it and users from the inside to leak information, but it can do nothing about attacks carried out from inside the network. This is only partly true for a distributed solution.
- *Hidden channels* are communication channels from and to the network which are not monitored by the firewall. Therefore, they are not part of the security concept.
- *Restricted design:* A firewall can only avert threats for which it is designed. It can be vulnerable to new modes of attack.
- *Malicious logic* (e.g., viruses, worms) delivered within the message body cannot be recognized by simple firewalls, because most of the filtering is based on communication protocol elements.
- *Tunneling* is encapsulating one communication protocol inside another one. This technique is used to transport a network protocol through a network which would otherwise not support it, or to provide various types of a virtual private network (VPN) functionality. It can also be used to bypass a firewall system. In this case, firewall-blocked data is encapsulated inside a commonly allowed protocol. Again, this is only partly true for a distributed solution.

3.3 Intrusion detection

Adopting the definition worked out during the MAFTIA [66] project, *intrusion detection* concerns *the set of practices and mechanisms used towards detecting errors that may lead to security failure, and/or diagnosing attacks*.

An *intrusion detection system* (IDS) monitors network traffic, communication protocols, application specific protocols, system calls, or other modifications to detect attacks. It taps the appropriate resource, e.g., network and kernel, and then analyzes the gathered information in order to recognize malicious attacks, intrusions, or security failures. Finally, it reports back to the user. An *intrusion prevention system* (IPS) augments an IDS with the capability to react on certain recognized events. Figure 7 depicts the basic block diagram of an IDS; a sensor taps a data stream, forwards the data to an analyzing engine, which might make use of a signature database, and a console for user interaction.

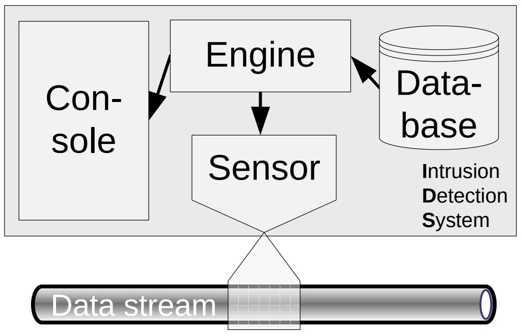

Figure 7: *Block diagram of intrusion detection system*

The detection of attacks on computer systems is to perceive the difference between a system's normal or expected behavior including error system states and the behavior if the system is under attack. Two categories of techniques exist with which the observed system is compared: *Anomaly–detection techniques* compare the observed system against normal usage profiles and *Misuse–detection techniques* check for activity profiles violating the system's security policy.

After [33], intrusion detection may be accomplished

- after the fact (post–mortem audit analysis),
- in near real–time, or
- in real–time (in support of automated countermeasures).

In a system implementing error detection and system diagnosis, it is necessary to differentiate between random errors and malicious tampering with the system. This can be impossible, because an attacker might launch an attack that behaves like an error, which is recognized by the diagnostic system as anomaly, but not detected by the IDS as a misuse. The paper in [12] discusses a methodology for detecting accidental versus attacks in sensor networks. It distinguishes a *benign attack* where the attacker behaves according to the correct system's behavior, and a *malign attack* that changes the observable behavior of a system. The paper addresses solely the latter, distinguishing the malicious or non–malicious nature of the former attack is not a computational matter.

3.4 Virtual Private Networks (VPN)

A *virtual network* links its nodes in a way that the underlying physical network must not correspond to the virtual connections between nodes. To the user a virtual network appears as one large network, whereas physically it may incorporate several networks or parts of networks. Furthermore, a *virtual private network* (VPN) provides a privacy service for these virtual communication lines. VPNs are mostly configured within large public networks like the Internet, in order to virtually create local area networks (LANs) which span large geographic areas.

Secure connections over insecure networks are implemented by wrapping an insecure protocol in a secure protocol. This technique of encapsulation is called *tunneling*. For example, the Internet Protocol's (IP) packets are transmitted in plaintext. By embedding an IP packet as payload data into its secure counterpart IPSec, an regular IP connection can be made secure transparently to the user.

The three core services a VPN must facilitate are *authentication*, *encryption*, and *validation*. More specifically, sender authentication is required for access control and to counter identity spoofing, data confidentiality is optional and established by encryption and blocking packet snooping and sniffing, and, finally, message integrity measures provide validation and mitigate message alteration.

Table 1 gives an overview of some widely used VPN protocols with IP. Figure 8 shows the layering of a virtual network on top of a physical network. The wiring is drawn arbitrary.

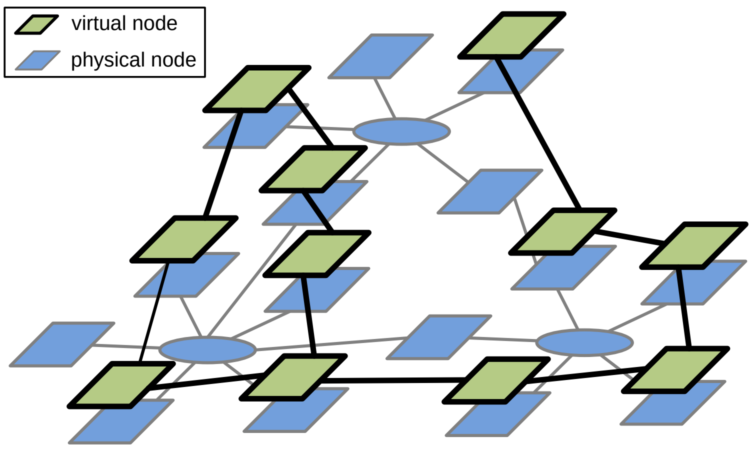

Figure 8: *Virtual network on top of a physical network.*

NAME	DESCRIPTION	REF.
IPSec (IP security)	The security extension for IP. Comes in two flavors: authentication header (AH) support sender authentication and message integrity; encapsulated security payload (ESP) implements all three core services. It is implemented at the transport layer in the OSI model.	[44]
OpenVPN	This open source VPN is based on the SSL/TLS network stack for secure communication and resides on the application layer. The advantage of this approach is that no radical changes have to be made to the operating system's network stack and the SSL/TLS can be run on virtually any client.	[28]
PPTP	The Point–to–Point Tunneling Protocol was developed jointly by a number of companies, including Microsoft. Its popularity is due to the fact that is was the first VPN shipped with Microsoft Windows and its easy configuration.	[34]
L2TP	The Layer 2 Tunneling Protocol was derived from the PPTP protocol. It does not include encryption, but is often used in conjunction with IPSec. A point–to–point protocol PPP is assumed on the underlying layer.	[64]
L2TPv3	The Layer 2 Tunneling Protocol version 3 is a new release.	[54]

Table 1: Overview of VPN protocols used with IP.

4 Cryptography

The basic terminology [4] is that *cryptography* is the science and art to design ciphers; *cryptanalysis* is the science and art of breaking them; *cryptology* is the study of both. Cryptography provides the tools, which underlie most modern security protocols. A *cryptographic scheme* or *cryptographic system* encompasses one or more cryptographic algorithms (*cipher*s), protocols, data processing and storage facilities, and its users. Basically, any computer system that involves cryptography is called a cryptosystem[2]. Because of this, breaking a cryptosystem is not restricted to breaking the underlying cryptographic algorithms. Usually it is far easier to break the system as a whole through finding the weakest link in the system, which is usually not the cryptography.

The ultimate goal of cryptology is hiding information from others. In basic scenario, a sender transmits a message to a receiver and doesn't want anybody else to read the message. For this purpose, the relevant piece of information is encoded by the sender with some cryptographic algorithm, transported over a medium, and then again decoded at the receiver. To prevent a third party to read out the message from the medium in plaintext, but to grant access to the message's information to the appointed receiver, some kind of asymmetry has to be introduced between those two kinds of parties. In most cryptographic algorithms this asymmetry is introduced by some shared secret only known the sender and the receiver, e.g., a password or a numeric key. The security of a cryptosystem thus depends on the concealment of the secret.

Figure 9 sketches a cryptosystem's basic principle of operation: A *plaintext* is used as input for a black box cryptosystem, which then performs an *encryption* operation under key_1 and it yields a *ciphertext* as output. Consequently, the ciphertext can be deciphered by a *decryption* operation with key_1, whose output is the input plaintext from the start.

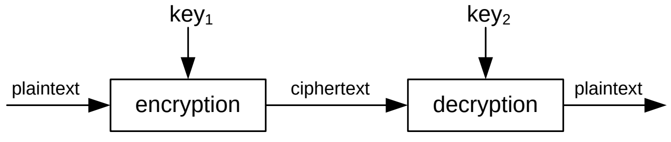

Figure 9: *Cryptosystem: Principle of Operation*

[2]Note that in the context of cryptography, *cryptosystem* refers to a suite of algorithms needed to implement a particular form of encryption and decryption.

There are two major classes of algorithms in cryptography, namely private-key or symmetric algorithms and public-key or asymmetric algorithms. If encryption and decryption operation are performed using the same key ($key_1 = key_2$), the operation is symmetric, if both keys are distinct ($key_1 \neq key_2$), the operation is called asymmetric. An analogy for symmetric schemes is a safe box with a lock: everybody holding the key can open the box and deposit messages inside or retrieve messages from the box. An asymmetric scheme extends this analogy by making a slot in the box (hence a letter box): everybody can insert messages through the slot in the box, but only the receiver can open the box and get his messages out [83].

Tutorials on cryptography can be found on every corner of the Internet and nearly every text book on security has also a chapter on cryptography [4, 17, 18]. The classic book on cryptography is written by Bruce Schneier [73]. The papers in [83] highlight cryptography in the context of embedded systems.

The remainder of this chapter is organized in four chapters. Firstly, some generic attack modes on ciphers are discussed. Next, a short introduction on the two kinds of cryptography, asymmetric and symmetric, is given. The last chapter presents some implementation issues, computation of cryptographic algorithms, and performance measurements.

4.1 Cryptanalysis and Security of Ciphers

An attack on a cryptosystem aims at revealing the concealed information without prior knowledge of the secret key. This challenge can be met by exploiting certain properties of the encryption or decryption operation. Note that this section presents attacks on the cipher directly. In cryptographic schemes there exist other possibilities like, e.g., power analysis (see Section 6.3) to break the scheme. The most simple method to break a cipher is through guessing or (randomly) probing all possible keys. This attack mode is referred to as *Brute–Force–Attack*. The size of the key space determines the worst case assumption for the number of guesses, till one is successful. More sophisticated attacks try to narrow the key space by eliminating some of its portions or by improving the way the guesses are selected.

A cipher's security can be defined as a function of the *algorithm's strength* and *key length*, which spans the key space. The former aspect is more important in terms of security, the latter aspect is easier to demonstrate and hence better perceived by the public. Mounting an attack usually consists of two operations, generating a guess and checking it by a test. Consequently, the *worst case execution time* is for a Brute–Force–Attack the size of the key space multiplied by the duration of key generation plus a subsequent check. Modern cryptosystems have key spaces larger than the number of atoms in the universe[3] and the computation of a Brute–Force–Attacks could last longer than the universe's expected lifetime[4]. Nevertheless longer key length can have a positive impact on a cryptosystem's security, the system should still remain usable; handling key sizes of some kilobyte can be cumbersome and even insecure. Brute–Force–Attacks can usually be highly parallelized by partitioning the key space and thus reduce the time spent for breaking the cipher by a constant factor. This introduces a forth aspect of security, besides strength, key length, and time to break, namely *cost*. This aspects relates to the cost of a machine's computation time (acquisition cost and operating cost) and the number of machines needed to deploy the attack. Implementations of highly parallelized distributed cracking machines range from special hardware supercomputers on the one hand [82], to workstations connected to the Internet and orchestrated by software using the computer's idle time on the other hand [1].

So how can a cipher be kept secure? It has been widely acknowledged, that concealing the details of the algorithm is never secure. There is no such thing as "security by obscurity". The only way to assure oneself of a cipher's strength is to submit it for a public review process. The more experts take a look on the cipher, the higher the probability it is secure. For example, the new Advanced Encryption Standard (AES) has been selected by the NIST after a three year period of public test out of five candidates.

The remainder of this chapter introduces common threat models to ciphers, symmetric and asymmetric cryptography, and, finally, it discusses some performance related issues. Please note that most applications use hybrid cryptosystems, a conjunction of both symmetric and asymmetric cryptography in order to achieve security and performance. The latter criteria is especially important, because the cryptosystem will fail, if it is delaying the user from doing his work..

4.1.1 Threat models and attack modes

In Bruce Schneier's book [73], four primary and three secondary cryptanalytic attack modes are distinguished:

[3]Which is around 10^{80} in the observable universe.

[4]Which ranges between 10^{60} and $e^{10^{50}}$ according to latest estimations [63].

1. **Ciphertext–only.** The attacker has access to a set of ciphertexts, encrypted with the same cipher. His task is it to find out the corresponding plaintexts, or – even better – the secret key. This is a standard scenario of an attacker eavesdropping a communication channel.

2. **Known–plaintext.** The attacker has access to a set of cipher texts and their corresponding plaintexts. His task is it to find out the secret key or an algorithm to decrypt further messages, which have been encrypted with the same key.

3. **Chosen–plaintext.** The attacker has access to a set of cipher texts and can produce their corresponding plaintexts, e.g., he has a cryptographic device and can input arbitrary plaintexts and read the device's output. His task is it to find out the secret key or an algorithm to decrypt messages, which have been encrypted with the same key, i.e., find a way to duplicate the device.

4. **Adaptive–chosen–plaintext.** This method is similar to the chosen–plaintext attack mode, but the attacker can as well vary the subsequently used plaintexts based on the information from the previous encryptions.

5. **Chosen–ciphertext.** The attacker has access to a set of ciphertexts, can decrypt them without knowing the key, and retrieve the output plaintexts. Sometimes this is called the "lunchtime" or "midnight" attack, where an attacker gains access to an unattended decryption machine.

6. **Chosen–key.** The attacker can make use of some knowledge about different used keys in the cryptosystem. This is a rather uncommon attack mode.

7. **Using violence.** Also called *"rubber-hose cryptanalysis"*, this attack is exploiting the human factor and addresses all methods involving physical violence, blackmailing, kidnapping, threatening, corrupting, and taking advantage of someone. In most cases, this is the most effective method.

Practically, a couple of attack modes are combined for attacking a cryptosystem. The following example illustrates an attack on some encrypted messages sent by a central master in a network:

The attacker eavesdrops messages on the network. He fails to mount a successful ciphertext–only attack. So he decides to try a chosen–ciphertext by circumventing some authentication mechanism on the central server. He succeeds and ends up with a set of plaintexts and ciphertexts, which he can use again for a chosen–plaintext attack, or – if he can repeatedly access the server — an adaptive–chosen–plaintext attack.

4.2 Symmetric Cryptography

Symmetric cryptography [53] provides the ability to securely and confidentially exchange messages between two parties. As mentioned before in symmetric cryptography one key is used to encrypt and decrypt a message. This key represents a shared secret between the participants and every participant has a copy of the key. The inverse encryption function corresponds to the decryption function. There are two modes of operation, block ciphers and stream ciphers. The former take a chunk of bits for encryption, the latter operate bit wise.

According to Shannon [77], two complementary concepts can be used to conceal information, namely *confusion*, which is making each output depend upon the key, and *diffusion*, which is each output depend on each of the previous bits input. In block ciphers confusion is understood as substitution (replacing bits), and diffusion is understood as transposition or permutation (changing a bit's position in the block). These functions are implemented as so–called S–boxes and P–boxes, which are aligned and connected in certain ways forming the cipher. The key serves as driver for the boxes. An attacker could assemble the same arrangement of boxes, but without knowledge of the key the boxes don't uncover the secret.

Stream ciphers operate on a stream of digits and encode or decode each digit depending on the current state of the algorithm. Thus, the encoding of one digit depends on last encoded digit. During that process a sequence of pseudo-random numbers (keys) is applied to a data sequence. This pseudo-random property makes the algorithm vulnerable to attacks. Ideally, a stream of truly random numbers is used like proposed for the one-time pad; but if there is no way to reproduce the key stream, there is no way to decode the data stream. An application field for stream ciphers are secure wireless connections, e.g., antenna Pay–TV or mobile communication.

A cryptographic mode combines a basic cipher with some sort of feedback or some simple operations. The modus specifies how subsequent data blocks are processed. The security relies on the cipher whereas the modus should not mitigate its strength, or even cause a vulnerability. On the other side, a well-chosen modus can improve a scheme's security. Table 4.2 lists some modes mainly for block and stream ciphers.

Electronic Codebook (ECB)	A block of plaintext has exactly one corresponding ciphertext. Each block of plaintext is encrypted separately. Like in a lookup table, every input has the same output assuming the same key.
Cipher Block Chaining (CBC)	A feedback loop is introduced: the plaintext of one block is XORed with the ciphertext of the previous block. The decryption operation works vice versa. The important part is to choose a secure first block to start with, the so–called Initialization Vector (IV). Two identical plaintext messages map to two different ciphertext messages, if two different IVs are used.
Cipher Feedback Modus (CFB)	This modus is very similar to the CBC mode. The ciphertext of the previous block is fed back, encrypted again, and XORed with the current block. Actually the plaintext does not pass the encryption operation. Starting with an IV, a stream of pseudo–random bits is generated which is interweaved with the plaintext. This mode enables a block cipher to work as a self-synchronizing stream cipher.
Output Feedback Modus (OFB)	This mode enables a block cipher to work as a synchronous stream cipher. It works similarly to the CFB mode, but the result of the encryption operation is fed back, not the entire ciphertext. Consequently, a stream of infinite length could be produced. On the contrary in CFB mode, the algorithm blocks until the result of the XOR operation is available.

Table 2: Cyptographic modes

4.3 Asymmetric Cryptography

Contrarily to symmetric cryptography, there is no shared secret in asymmetric cryptography [84]. The encryption is done with a key which is publicly available. The decryption is performed with a secret private key, which is mathematically related to the public key. The key actually used for decryption is never transmitted over an insecure line and thus remains a secret. A drawback is that the computation of an asymmetric algorithm is more resource intensive compared to a symmetric one.

The security of asymmetric cryptography is that a specific mathematically problem can be solved by someone having a hint (the key) easily, but it is unsolvable for someone who doesn't have the hint. In the following some problems which are suitable for this purpose:

- The *factorization of large integers* is the task to compute an integer's prime factors. If the number is composed by large integer primes this computation is currently unsolvable with public known algorithms. The bottleneck operation of every known factorization algorithm is performing primality testing (see Appendix B). If there would be a fast method to compute prime numbers, this would break the security. of factorization–based cryptography (among several other things).
- The *discrete logarithm problem* exploits the fact that there is no efficient algorithm for computing discrete logarithms, while the inverse problem of discrete exponentiation can be computed efficiently, e.g., by using exponentiation by squaring, for example. The discrete logarithm problem applies to finite fields (Diffie–Hellman and DSA, see [27]) and to arbitrary groups, which is the motivating problem for Elliptic Curve Cryptography (ECC).

4.3.1 RSA Algorithm

The most important operation for the RSA cipher is the *modular exponentiation* and represents solving the equations for encryption (1) and for decryption (2):

$$C = M^e \, mod \, n \tag{1}$$

$$M = C^d \, mod \, n \tag{2}$$

p, q ... Large primes	key length
M ... Plaintext message	
C ... Ciphertext message	
e ... Public key	relativ prim zu $(p-1)(q-1)$
d ... Private key	$e^{-1} mod((p-1)(q-1))$
n ... Modulus	$n = p \cdot q$

Table 3: Components of the RSA algorithm

Two large primes (p, q) are used to make a public-key/private-key pair (e and d) and the modulus n. Since n is part of the public key, everybody who can factorize n will be able to recompute d. So the security of RSA is based on the mathematical problem of factorization. A more detailed description of the algorithm can be found in [73].

Due to the political problems concerning cryptography in the United States, there is no standard available from the ANSI. Nevertheless, RSA Security, Inc., published some defacto standards addressing RSA crypto systems.

These *Public Key Cryptography Standards* (PKCS) can be downloaded from the company's website[5]. Table 4.3.1 gives an overview of the different standards. They contain a specification to build public key cryptographic schemes, i.e., a combination of the basic RSA operation, data structures, and a padding scheme. This is necessary to overcome different attacks on the basic RSA operation, which might result from weak keys, chosen plaintext, or chosen ciphertext attacks.

PKCS # 1:	RSA Cryptography Standard
PKCS # 3:	Diffie-Hellman Key Agreement Standard
PKCS # 5:	Password-Based Cryptography Standard
PKCS # 6:	Extended-Certificate Syntax Standard
PKCS # 7:	Cryptographic Message Syntax Standard
PKCS # 8:	Private-Key Information Syntax Standard
PKCS # 9:	Selected Attribute Types
PKCS #10:	Certification Request Syntax Standard
PKCS #11:	Cryptographic Token Interface Standard
PKCS #12:	Personal Information Exchange Syntax Standard
PKCS #13:	Elliptic Curve Cryptography Standard
PKCS #15:	Cryptographic Token Information Format Standard

Table 4: PKCS standards

The PKCS are as well part of other standardization initiatives, several RFCs have been published using the PKCS as a foundation [43, 41, 42, 59].

4.3.2 Elliptic Curve Cryptography (ECC)

ECC cryptography relies on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), that is defined on elliptic curves over finite fields F_p^m forming a group. They have a finite number of points,

[5]`http://www.rsa.com`

and their arithmetic involves no round off error. Elements of the field F_p^m are m–bit strings (vectors). The rules for arithmetic in F_p^m can be defined by either polynomial representation or by optimal normal basis representation [6] Since F_2^m operates on bit strings, its arithmetic computations are well–suited for computers.

Scalar point multiplication is the main cryptographic operation and consists of the rudimentary operations *point addition* and *point doubling* on elliptic curves over finite fields. The discrete logarithm problem is to compute the scalar k in $P = kQ$ (with $P, Q \epsilon F_p^m$). The elements of a group over F_2^m can be computed by a generating element G, a so–called *generator*. In a cryptosystem, parameter m defines the key length and it must be chosen large enough (e.g., $m = 163$), to prevent the efficient generation of a table of elements. By the use of an element table, the discrete logarithm problem can be solved by a simple table lookup. An asymmetric key pair consists of a random number k, which serves as the private key, and a corresponding public key computed by kG.

This computational asymmetry represents the cryptographic primitive which is used to create public–key schemes. Schemes are available for message signature (ECDSA, ECPVS, ECNR), encryption (ECIES), and key agreement (ECMQV, ECDH). The Standards for Efficient Cryptography Group (SECG) published an important standard referencing EC–based algorithms in [23, 24].

The smaller keys sizes used in ECC cryptography result in faster computations, lower power consumption, as well as memory and bandwidth savings.

An interesting tutorial on the mathematics of elliptic curves among lots of other useful information can be found at Certicom's website[7].

4.4 Computation

Summarizing, the rudimentary operations executed by a CPU to compute an asymmetric cipher are modular exponention, modular multiplication, point addition, and point doubling. For symmetric ciphers, the atomic operations are bit–shuffling for permutation or P–boxes, simple non–linear functions for substitution or S-boxes, and linear mixing using XOR. These operations bear the main load of cryptographic computation. Even a slight improvement in one operation can improve a cryptosystem's overall per-

[6]While optimal normal basis multiplication is less insightful than polynomial multiplication, it is in practice much more efficient.

[7]`http://www.certicom.com`

ECC (bits)	RSA (bits)	Key size ratio	AES (bits)
163	1024	1:6	–
256	3024	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Table 5: NIST guidelines for the equivalent strengths of various cryptographic algorithms

8-bit micro controller	RSA-1024	ECC-160	
Intel 8051 (14.75 MHz)	105s	4.6s	20x
Atmel AVR (4 Mhz)	22s	1.62s	14x

Table 6: Supplied by NIST to ANSI X9F1

formance tremendously.

The paper in [45] provides an overview of many modular exponention and modular multiplication methods. The most well–known algorithm is called the *binary method* or *square and multiply method*, and dates back to antiquity.

An efficient implementation for point multiplications uses a discrete Fourier transform (DFT) based method originally proposed for integer multiplication. The article in [9] proposes to use number theoretic transform (NTT), which is found in many digital signal processing applications and because DFT falls short in case of small integers as used for ECC cryptography (uses e.g., 163 bit integers).

Finally some collected performance statistics and reference material. Table 5 shows that ECC key sizes scale linearly, whereas RSA does not. This shows that future security requirements for strong cryptography can be computed with less power by the ECC. This table is widely cited and can be used reliably for a comparison between the different cryptographic schemes [31].

In Table 6 implementations for RSA and ECC are compared for 8-bit architectures. The results show, that the ECC performs 20 times faster for the Intel 8051 and 14 times faster for the Atmel AVR. This information is taken from a Sun presentation available on the Internet[8]. If put in relation with the expected growth in key length in Table 5, the advantage of the ECC over the RSA is even more obvious.

At last a reference to a rock solid comparison study of RSA and ECC for

[8] `http://research.sun.com/sunlabsday/docs.2004/talks/2.03_Chang.pdf`

Algorithm	ATmega128 @ 8MHz			CC1010 @ 14.7456MHz		
	time s	data mem bytes	code bytes	time s	data mem ext+int, bytes	code bytes
ECC secp160r1	0.81s	282	3682	4.58s	180+86	2166
ECC secp192r1	1.24s	336	3979	7.56s	216+102	2152
ECC secp224r1	2.19s	422	4812	11.98s	259+114	2214
Mod. exp. 512	5.37s	328	1071	53.33s	321+71	764
RSA-1024 public-key e = 216 + 1	0.43s	542	1073	>4.48s		
RSA-1024 private-key w. CRT	10.99s	930	6292	∼106.66s		
RSA-2048 public-key e = 216 + 1	1.94s	1332	2854			
RSA-2048 private-key w. CRT	83.26s	1853	7736			

Table 7: Average ECC and RSA execution times on the ATmega128 and the CC1010 after [32]

embedded systems is given in [32]. Table 7 is taken from there.

5 Trusted Computing Platform (TCP)

"The Trusted Computing Group (TCG) is a not–for–profit organization formed to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices. TCG specifications will enable more secure computing environments without compromising functional integrity, privacy, or individual rights. The primary goal is to help users protect their information assets (data, passwords, keys, etc.) from compromise due to external software attack and physical theft."[9]

The TCP is a consortium of about 170 companies from the IT business, incorporating many big players. Corporations like AMD, Hewlett–Packard, IBM, Infineon, Intel, Lenovo, Microsoft, Sun, etc. put a joint effort in developing open industry–wide specifications for trusted computing across multiple platform types. The range of specifications covers:

- Infrastructure Specifications
- Mobile Phone Specifications
- PC Client Specifications
- Server Specific Specifications
- Storage Specifications
- Trusted Network Connect (TNC) Specifications
- Trusted Platform Module (TPM) Specifications
- TPM Software Stack (TSS) Specifications

However, this endeavor is controversial. In the beginning, the TCG has been critized because of its original motivation to establish digital rights management (DRM) and to prevent people from running unlicensed software, which is a rather economical intention under the sheepskin of user security. One main criticism was that the specification formerly known as TCPA takes away control from the user and gives it to the software and hardware selling companies. Nowadays, the biggest concerns are remote censorship and blacklists of computers. Many experts published articles on the web [78, 5, 74] discussing the benefits and dangers of the workings of the TCG.

[9]`https://www.trustedcomputinggroup.org/about/`

5.1 Goals of the TCG

The TCG states four goals of trusted computing. Please note that the emphasis is on 'trusted' and not on 'secure':

- *Attestation*: Proving the system's integrity (data and programs) and validation of the platform to third parties. This is necessary to prove the system's identity to others. Based on one root key, it introduces platform identity aliases, that may be associated with information relating to a specific use or domain.

- *Sealing* is the binding of data to the system configuration. Data is stored encrypted with a corresponding value of the system configuration. It may be unsealed, only if the original system state can be verified. Consider for example some encrypted data stored together with a hashsum of the encryption/decryption algorithm to guarantee its implementation has not been corrupted. Or some multimedia data stored together with a hash sum of a certain codec to ensure it can only be played by the software intended by the provider.

- *Secure storage* of cryptographic keys. Provide a secure storage for sensitive information. This encompasses space for a unique platform identification key and several derived keys. Generally, the TCG specifies *migrateable* and *non–migrateable* keys, i.e., key–pairs which may leave the TPM, and keys which don't.

- To provide secure *cryptographic primitives* like random number generator, hash function calculator, etc. in hardware. Cryptosystems implemented in software can be insecure for two reasons, they can be replaced or circumvented or the processed keys can be tapped, e.g., by power analysis attacks.

5.2 TPM Specification Overview

The idea to boot a computer system into a trusted state before giving control to the operation system appeared first in [8]. It is based on the assumption that it is much easier to manipulate software than hardware. The solution to this problem is to implement trust anchor in hardware, a mechanism to record what software is/was executed. This is the vision of the "Trusted Computing" functionality. One step into this direction is TCG's Trusted Platform Module (TPM) specification for such a *hardware root of secrecy*.

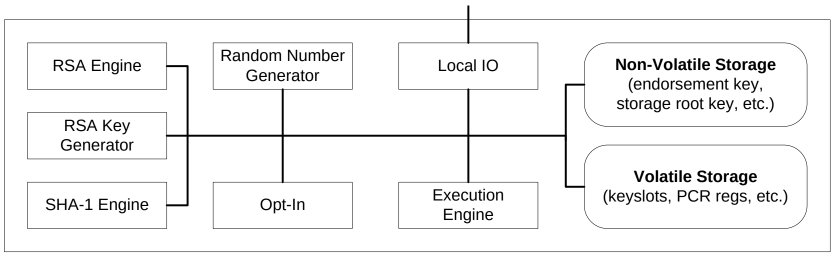
RSA Engine
RSA Key Generator
SHA-1 Engine
Random Number Generator
Opt-In
Local IO
Execution Engine
Non-Volatile Storage (endorsement key, storage root key, etc.)
Volatile Storage (keyslots, PCR regs, etc.)

Figure 10: *Block diagram of a Trusted Platform Module (*TPM*)*

The schematic view of a TPM is depicted in Figure 10.
A TPM consists of following components:

- The RSA *engine* is a hardware implementation of the RSA algorithm (see Section 4.3.1), which supports 512, 1024, 2048 bit keys.
- The RSA *key generator* generates public/private key pairs. Mostly used for temporary session keys.
- The SHA–*1 engine* is a cryptographic primitive used by the TPM as its trusted hash algorithm.
- The *hardware random number generator* provides a source of true random numbers. This is for the nonce used in cryptographic protocols, key generation, etc.
- *Opt–in/out* User can decide whether or not he wants to enable or disable TPM functions.
- *Execution Engine* performs the coordination of the TPM. It processes incoming requests and controls the other on–chip peripherals.
- The *local* I/O manages the communication from and to the device. Typically a Low PinCount Bus (LPC) is used.
- *Volatile storage*: To use a key it has to be loaded into the TPM. This is the location where session keys or external keys are stored for operation. The second purpose is to store integrity measurements in the so–called Platform Configuration Registers acpcr.
- *Non–volatile storage*: This storage is implemented tamper proof and used for the *endorsement key* and the *storage root key*.

A TPM's *root–of–trust* emerges from two keys, the *Endorsement key* (EK) which provides a unique platform identity and is physically bound to the TPM device, and the *Storage Root Key* (SRK) managing the on–chip key storage. The SRK is a 2048 bit RSAS key and is the top level element – the root – of the TPM key hierarchy. This key is recreated with each "TakeOwnership" operation, which binds a TPM to a particular user. Other keys stored in the TPM are encrypted with the public part of the SRK when transferred to the outside. Thus, they are useless without the TPM, which encrypted them. For usage, they have to be loaded onto the TPM, where the corresponding branch of the key chain is rebuilt. This forms a *Root of Trust*, which is a "hardware or software mechanism one implicitly trusts." A set of storage keys for different purposes are wrapped by the Root of Trust.

The TPM implements the concept of transitive trust; during initialization, a trusted component checks the next component to initialize. Hence, a tree of trust emerges with the initially trusted component as its root of trust. The steps to be taken are (a) compute the hash value of the next entity, check for correctness, and, if successful, pass control to the measured entity. This process starts with the BIOS initialization up to user level applications. This way the system can be forced by the operating system to execute only trusted applications. Because a TPM is a slave device which monitors and checks the system, it cannot alter itself the execution flow of the system.

TPMs are designed to be relatively low cost devices that is important for market acceptance. They provide only limited resistance against sophisticated hardware attacks, that is not perfect security, but better than a pure software solution.

5.3 Related projects and implementations

In order to comply with the specifications of the TCG, chipmakers have developed varying implementations that integrate the TPM functions into a normal chipset. See Section C for an overview of names and short descriptions.

The usual approach is to build a secure part of the operation system on top of the TPM. Most security critical operations are then performed in this secure system partition. The hardware guarantees that information flows only from the secure (or trusted) partition to the non–trusted partitions. A similar security pattern was firstly published as the *Bell–LaPadula Model* [13]. In this model one property says that information from the level above may not be read (*no read–up*) and information may not be disclosed to lower levels (*no write–down*). In order to pass information below it must be *declassified*, e.g., encrypted. The key again is stored with a higher security level than the message. Hence, the encrypted message can be read only from subjects belonging to the key's security level, but may be delivered by a subject from a lower level.

Microsoft's initiative is called Next Generation Secure Computing Base (NGSCB, formerly Palladium). NGSCB consists of a security kernel called the *Nexus* that is part of the operation system, and *Nexus Computing Agents* (NCAs), which are trusted software agents invoked by the applications and interfacing the Nexus.

The ARM's TrustZone security extension features two virtual processors backed by hardware based access control. These two processors represent two worlds of trust. The access controls prevent information from leaking from the more trusted world to the less trusted world. This approach facilitates the same design pattern as in Microsoft's NGSCB: a rich operation system executes in the less trusted part and a small security kernel in the more trusted world.

6 Physical Security

In an embedded system two sub–fields of security merge, on the one hand *physical security* which is concerned with preventing or deterring attackers from accessing a facility, resource, or information stored on physical media, and on the other hand *information security* which is protecting programs and data against unauthorized access or modification, whether in storage, processing, or transit, and against denial of service to authorized users.

Section 2 lists several properties of embedded systems and stresses that they require to stay operational and secure even in an *untrusted environment*. Consequently, they can not always trust their users or operators. Attackers may try, e.g., to seek out failure modes, weak default values, or simply use brute force in order to gain illegal access to information stored on the device or to operate the device offside it's specification.

The remainder of this section first introduces the terminology on tamper resistance. Next it presents some hardware security measures. Finally, it focuses on physical and side–channel attacks. These types of attacks refer to attack modes, which use physical properties of the embedded system. They can be further classified into invasive (e.g. microprobing, reverse engineering) and non–invasive attacks (e.g. timing or power analysis). Often, attackers take a combined approach, in the first phase they collect information during an invasive attack, which is then used to design a non–invasive attack.

6.1 Tamper resistance

Tamper resistance [56, 70] is concerned with protecting devices from unwanted physical access. On the contrary to side channel attacks (see Section 6.3), which is a passive, physical implementation attack, device tampering is an active, physical attack, executed to gain access to device internals like the bus subsystem, I/O pins, or even CPU registers. Some basic design principles for building tamper–resistant devices can be found in [50].

Figure 11 is a reproduction from [70]. It shows at which points in time contermeasures are active during an attack. Before an attack starts *attack prevention* measures are enabled. These can aim at avoiding the start of an attack and can be achieved by deterrence, deception, monitoring, access control, etc. If an attacker has successfully launched an attack the next stage is *attack detection*, which is facilitated by intrusion detection systems. After an attack has been detected and is still going on, a response action can be taken, or, if the attack is over, *attack recovery* must be facilitated and a secure system state has to be recovered. To notice that an attack has occurred, measures to provide evidence are necessary to establish.

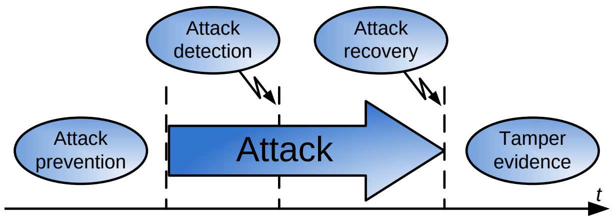

Figure 11: *Tamper response modus operandi*

According to [56], devices can be aware of tampering in three ways:

- *Tamper–evident characteristics* is to provide evidence that an attack has been attempted. This is achieved by applying security seals, using special covers, or enclosures, thus, something breaks when tampering to prove the act to an authority.
- *Tamper-resistant characteristics* is to provide passive physical protection against an attack. Chip–design measures include the encryption of internal bus lines and memories which contain critical persistent data. Moreover, the layout should contain special characteristics, such as scrambling of bus lines and memories as well as special logic styles [56]. If an attacker manages to successfully tamper the device, he can get only useless information out of it.
- *Tamper-responsive characteristics* is to provide an active response to the detection of an attack, thereby preventing its success. For example zeroisation achieves this by the deletion of all security relevant data (e.g. keys), when an attack is detected rendering the device useless. Of course, it is crucial to minimize the delay between when an attack is carried out and when the response action is taken to prevent the attacker from interfering with the response procedure.

6.2 Hardware measures

One of today's microprocessor design goals is to maintain observability and controllability to support testability of the chip. These hardware measures and interfaces built on a chip may severely decrease the chip's security by opening a backdoor to everyone who is able to plug and operate (e.g., a JTAG) debugger to the device. On the other hand a correct behavior and well–tested design is even more stringent when building secure chips [35]. Disabling all on–chip debugging resources after production may be possible either through software by locking the chip through a special register, or by hardware means, e.g., cover test points with epoxy, or, as with smart cards, scrub off test circuitry from the chip. Furthermore, provide a logic enforcing that security functions do not execute with JTAG–enabled hardware.

Some best–practices for a robust hardware design are listed in [68]:

- Put critical data bus traces below the surface of the board
- Use a highly–integrated chip design or high–density packaging
- Use on–chip RAM to secure keys during decryption
- Use a CPU that supports segment-level cache locking
- Support the use of a smart card, SIM card, or TPM
- Use chassis tamper–detecting hardware

A recently discovered way [65] to imprint a unique identity to chips is a particular integrated circuit called *physically uncloneable function* (PUF). This technique is embedded in the design process of the semi–conductor. Because of process variations, no two Integrated Circuits are identical. The idea is to extract information from a complex physical system, a specially designed circuit. This circuit implementing the PUF is loaded with a so–called *challenge* and yields a response. A Silicon PUF can be used as an unclonable key. The lock has a database of pre–computed challenge–response pairs. These pairs act as shared secret and authenticator of the PUF. To open the lock, the key (PUF) has to show that it knows the response to one or more challenges. Figure 12 depicts this relationship. Technically, the PUF delays depend on overlaid metal layers and chip package. Any invasive attack (e.g., package removal) changes PUF delays and destroys PUF. Non–invasive attacks are still possible.

6.3 Side–Channel Attacks

"One Cannot Not Communicate." Paul Watzlawick [81] states that every kind of behavior is a kind of communication. Even intentionally sending out no signals is a signal. In the context of embedded systems, we want to interpret this quote that an embedded system does not only communicate

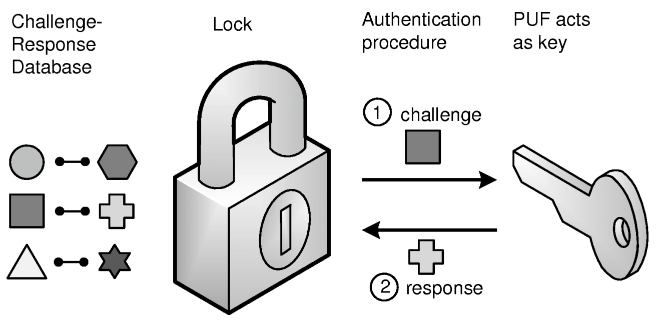

Figure 12: *How a* PUF *can be used as an uncloneable key.*

over the intended channels, but also has other non–intentional channels, so–called side channels. For a side channel attack [47, 57] non–obvious data like power consumption and target temperature is analyzed.

Cryptographic schemes are designed as black–boxes with concisely defined interfaces. During the functional design phase of an embedded device this assumption of a closed environment which does not leak any sensitive information perfectly holds. However, in a real–world application it strongly contradicts the physical nature of every implementation, which will leak side–channel information, if no appropriate counter measures are taken.

Cryptographic devices encompass data processing components like dedicated logic circuits implementing a cipher and memory components. Cryptographic operations can be implemented either entirely in hardware or on a general–purpose processor in a hardware–software co–design approach. Sources for side–channels attacks are exposed by *timing information*, *power consumption*, *electromagnetic leaks*, or even *sound* of the device. In the general attack pattern, the cryptanalyst uses one or more side–channels to gain extra information about secrets in the system.

During a cryptanalytic analysis like, e.g., a chosen–plaintext attack (see Section 4.1.1 for more attack modes), the attacker captures additional information like a power trace of an encryption run. The collected data can the be used for guessing key bits after applying noise reduction or similar stochastic techniques to preprocess the information.

6.3.1 Power Analysis

This section summarizes power analysis attacks on cryptographic devices. The first article on this topic was written by Paul Kocher and appeared in [46]. There are a comprehensive master's thesis [76] in german language and a book [57] explaining basic and advanced techniques for performing power analysis.

The basic principle of operation and setup for a power analysis attack is depicted in Figure 13. The setup consists of a cryptographic device (upper left), a workstation for command and control (upper right), and a measurement device like an oscilloscope (bottom). The workstation starts the cryptographic operation on the device in (1). Subsequently, the device triggers the beginning of a measurement on the oscilloscope (2), e.g., through some particular waveform or it sets a pin to signal that the operation starts. Now the oscilloscope does its work and samples the power consumption of the device during the run (3). Usually, it measures the voltage drop across a small ohmic resistor, a *measuring shunt*. Finally, the captured power trace is transmitted back to the workstation (4), which will need to collect a couple of them for a further analysis. A sample power trace is depicted in Figure 14.

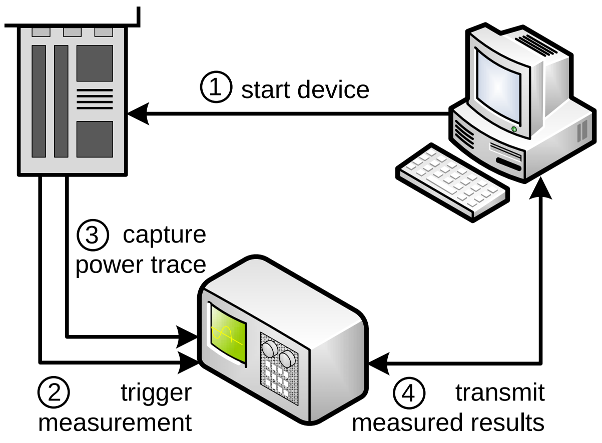

Figure 13: *Setup for a power analysis attack*

After capturing power traces of a cryptographic operation, the analysis starts in a second step. To be able to interpret the measurements correct, first a *power consumption model* has to be derived. In a CMOS[10] circuit, the total power consumption is the sum of the power consumption of all logic cells assembling the circuit. A CMOS logic cell has a (low) static consumption originating from the leakage current between the complementary transistors. The dynamic power consumption is due to the switches

[10]Complementary Metal–Oxide–Semiconductor, one of the most common technologies used by the chip manufacturing industry. It uses complementary and symmetrical pairs of p–type and n–type field–effect transistors for logic functions

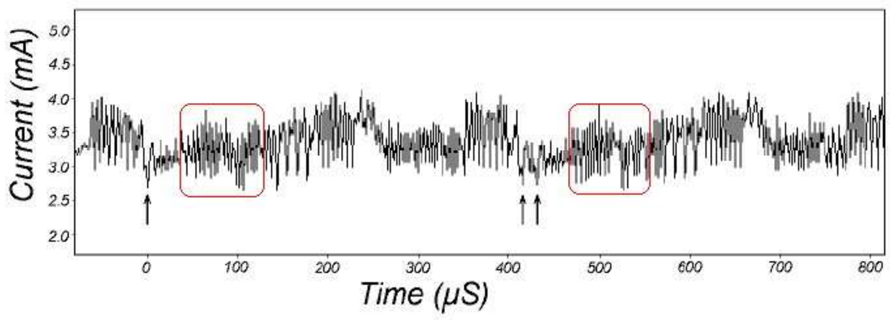

Figure 14: *Example power trace*

of the input or output signal of a logic cell. Basically, a signal has four possible transitions: during $0 \rightarrow 0$ and $1 \rightarrow 1$, the signal stays constant and only static power is drained; when $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions occur, dynamic power is drained, which is *data–dependent*.

Basically two types of power analysis can be distinguished, *Simple Power Analysis* (SPA) and *Differential Power Analysis* (DPA). In SPA the attacker tries to read the sensitive data (e.g., the key) directly out of the power trace. A DPA attack compares the actual power traces to a stochastic hypothesis called the oracle. It aims at tuning the oracle that it will always make a correct guess. The two types of attacks differ in several ways, SPA uses a small number of power traces and requires a detailed knowledge of the device, whereas DPA needs a large number and not many details on the device are necessary.

The goal of every countermeasure against is to make a device's power trace independent on the processed data. One well–studied technique is called *masking*. Each intermediate value is assigned a random value called mask which is generated inside the cryptographic device and not known to the attacker. The mask changes for every operation. At the end of the computation the mask is removed. A *masking scheme* describes how masks are subsequently applied to and removed from the data and the key during the algorithm execution. The power traces still give evidence of the values processed, but they are useless without knowing the masks. Another countermeasure to power analysis attacks is called *hiding*, which aims at removing the dependency between data values and power consumption. There are two ways to solve this, one approach is to randomize power consumption through additionally using power for some bogus operations, and another one is to build devices in a way that they consume equal amounts of power for all operations carried out.

7 Information Security Economics

Anderson and Schneier state in [7] that economic considerations of security are at least as important as the technical ones.

The definition of *risk* incorporates two sides: the chance a risk event will occur and the loss or harm resulting from the occurrence. Thus, on the one hand there is security, defined as freedom from harm and loss, and on the other there is risk, which is the possibility of suffering a risk loss. *Risk management* is then the process to optimize the security measures taken and the risk losses experienced. Information system security actions taken today work to reduce future risk losses. Because of the uncertainty of future risk losses, perfect security, which implies zero losses, would be infinitely expensive.

In order execute a *risk management procedure* the first step is to perform a *risk assessment*, which produces a quantitative, monetary measure of risk. The goal is to compare risks to another and to the cost risk mitigation techniques, which are the next step to be taken. On this basis one can decide, whether a measure to deflect a threat is worth the money spent on it. An organizations' *security management* consists of its risks and its risk mitigation measures. They are controlled by security auditing which goal is to provide an independent evaluation of the risk management.

7.1 Motivation

This section gives a short motivation why security will be a major factor for the success of an embedded system. For instance oil, gas, and electricity companies are shifting to a new way of distributing and accounting their goods. To hold off fraud and misuse they need security mechanisms to prevent severe financial losses.

"In a five to ten years perspective, it is expected that Integrated operations (IO) by smart use of real–time monitoring, real–time control, visualisation and new work processes based on Information and Communication Technology (ICT), will constitute the single most important driver for increased efficiency in the Norwegian oil and gas industry" [11]

7.2 Security Metrics

"If you can not measure it, you can not improve it." Bellovin uses this citation from William Thomson, the first Baron Kelvin, [79] to state that in the

[11] `http://www-05.ibm.com/no/solutions/chemicalspetroleum/io.html`

foreseeable future it will be impossible to create a metrics for security [16]. Finding a formula to numerically express a system's security is a difficult problem to solve.

"Security is not falsifiable" (Popper). We can prove that there has been a security failure, but we cannot prove that there has not. Popper goes even further and puts the presence of security into question. The quote can be thought of a binary measure, security is present or it is not. Can a system be considered secure, where some but not all security holes are closed?

The lowest expected cost for anyone to discover and exploit a vulnerability in a system is called the cost–to–break (CTB). The concept of this method has its origin in cryptography and reflects the cost/benefit proportion of an attacker seeking financial gain. It would be uneconomic to spend more money than the system's asset expected value to break into system [71].

7.3 Vulnerability markets

Nowadays, software is vulnerability–prone not because programmers don't know how to do better, in fact better tools and training are available than ten years ago, but in order to deliver the product to the market earlier than a competitor, security is often neglected. A software's security is hard to measure so there is no direct impact on the selling of the product. Moreover, it is hard for a customer to discern between a good security–enabled product and a bad one. So the incentives for a software company are low to integrate security in their product [6].

Two approaches exist to tackle this problem, *vulnerability markets* and *insurance*. Software vendors and security companies have established a legitimate market for software vulnerabilities. They offer rewards for vulnerabilities and exploits to anyone who is willing to share his knowledge. On the other side, their customers are interested to be informed about weaknesses in their IT–systems before anyone else can make use of this vulnerability. Sooner or later bugs are reported to the public and thereafter the attacks targeting a recently published vulnerability jump up. The paper in [67] discusses the existence of a vulnerability black market and makes suggestions towards a legal vulnerability market.

The subject of openness of vulnerability reports is widely discussed. Anderson concluded that it helps attackers and defenders equally [3]. Ozmet and Schechter found that it improves system security over the long term [61]. However, public disclosure of vulnerabilities can have some desireable side–effects, for instance it motivates vendors to come up with fixes more quickly.

The second approach mentioned above is to insure the IT infrastructure, which targets at transferring the risk from the owner to an insurance company. To deal with its customers systems, insurance companies send out expert assessors who analyse the risks and threats. This information helps both the insurer and the insured. With increasing experience, insurers learn to assess the risks in a contract more precisely.

8 References

[1] distributed.net – project page. Online, 2007. `http://www.distributed.net/`.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160(2):781–793, 2004.

[3] Ross Anderson. Open and Closed Systems are Equivalent (that is, in an ideal world). In *Perspectives on Free and Open Source Software*, pages 127–142. MIT Press.

[4] Ross Anderson. *Security Engineering*. John Wiley and sons, Inc., 1st edition, 2001.

[5] Ross Anderson. 'Trusted Computing' Frequently Asked Questions. Online, August 2003. `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`.

[6] Ross Anderson and Tyler Moore. Information security economics – and beyond. In *Proceedings of the 27th Annual International Cryptology Conference (CRYPTO'07)*, 2007.

[7] Ross Anderson and Bruce Schneier. Economics of information security. *IEEE Security & Privacy Magazine*, 3(1):12–13, January 2005.

[8] William Arbaugh, Davis Farber, and Jonathan Smith. A secure and reliable bootstrap architecture. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 65–71, May 1997.

[9] Selcuk Baktir and Berk Sunar. Frequency domain finite field arithmetic for elliptic curve cryptography. Preprint, january 2007. `http://ece.wpi.edu/~sunar/preprints/jrnl_paper.pdf`.

[10] Lokesh K Balakrishnan, Srinivasan Krishnamurthy, Vinod K Kumarasamy, and Lakshmikantha C Pothula. Firewalls. http://www.utdallas.edu/ sxk010620/Firewalls/firewalls.pdf, November 2001.

[11] Paul Barham, Steven Hand, Rebecca Isaacs, Paul Jardetzky, Richard Mortier, and Timothy Roscoe. Techniques for lightweight concealment and authentication in IP networks. Technical Report IRB-TR-02-009, Intel Research Berkeley, July 2002.

[12] Claudio Basile, Meeta Gupta, Zbigniew Kalbarczyk, and Ravi K. Iyer. An approach for detecting and distinguishing errors versus attacks in sensor networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 473–484, June 2006.

[13] D. Eliott Bell and Leonard J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, MITRE Corporation, Bedford, Massachusetts, March 1973.

[14] Steve M Bellovin and William R Cheswick. Network firewalls. *IEEE Communications Magazine*, 32(9):50–57, 1994.

[15] Steven M. Bellovin. Distributed firewalls. *;login: The usenix magazine*, pages 37–39, November 1999. Special Issue on Security.

[16] Steven M. Bellovin. On the brittleness of software and the infeasibility of security metrics. *IEEE Security & Privacy Magazine*, 4(4):96–96, July-Aug. 2006.

[17] Matt Bishop. *Computer Security : Art and Science*. Addison-Wesley, 5th edition, 2004.

[18] Seymour Bosworth and Michel E. Kabay. *Computer Security Handbook*. John Wiley and sons, Inc., 4th edition, 2002.

[19] BSI–Standard 100–1. Information Security Management Systems (ISMS). Bundesamt für Sicherheit in der Informationstechnik, December 2005. `http://www.bsi.de`.

[20] BSI–Standard 100–2. IT–Grundschutz Methodology. Bundesamt für Sicherheit in der Informationstechnik, December 2005. `http://www.bsi.de`.

[21] BSI–Standard 100–3. Risk Analysis based on IT–Grundschutz. Bundesamt für Sicherheit in der Informationstechnik, December 2005. `http://www.bsi.de`.

[22] Bundeskanzleramt Österreich. Österreichisches informationssicherheitshandbuch v2.3, April 2007. `http://www.a-sit.at/de/sicherheitsbegleitung/sicherheitshandbuch/`.

[23] Certicom. SEC 1: Elliptic Curve Cryptography. Standard, September 2000. `http://www.secg.org`.

[24] Certicom. SEC 2: Recommended Elliptic Curve Domain Parameters. Standard, September 2000. `http://www.secg.org`.

[25] R. Crandall, Apple ACG, and J. Papadopoulos. On the implementation of AKS-class primality tests. Online, March 2003. `http://images.apple.com/acg/pdf/aks3.pdf`.

[26] Tim Dierks and Christopher Allen. The TLS protocol. RFC 2246, Internet Engineering Task Force, January 1999. `http://www.rfc-editor.org/rfc/rfc2246.txt`.

[27] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[28] Markus Feilner. *OpenVPN: Building and Integrating Virtual Private Networks*. Packt Publishing Ltd, 2006.

[29] FIPS PUB 140-2. Security requirements for cryptographic modules. National Institute of Standards and Technology, 2002.

[30] Guy Gogniat, Tilman Wolf, and Wayne Burleson. Reconfigurable security support for embedded systems. In *Proceedings of 39th Hawaii International Conference on System Science (HICSS-39)*, pages 23–28. IEEE Computer Society, January 2006.

[31] Vipul Gupta, Sumit Gupta, and Sheueling Chang. Performance analysis of elliptic curve cryptography for SSL. In *Proceedings of the 5th ACM workshop on Wireless security*, pages 87–94. ACM, ACM Press, 2002.

[32] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer, August 2004.

[33] Lawrence R. Halme and R. Kenneth Bauer. Intrusion detection faq: Aint misbehaving: A taxonomy of anti-intrusion techniques. Internet FAQ. `http://www.sans.org/resources/idfaq/aint.php`.

[34] Kory Hamzeh, Gurdeep Singh Pall, William Verthein, Jeff Taarud, W. Andrew Little, and Glen Zorn. Point–to–Point Tunneling Protocol (PPTP). RFC 2637, Internet Engineering Task Force, July 1999. `http://www.rfc-editor.org/rfc/rfc2637.txt`.

[35] David Hély, Marie-Lise Flottes, Frédéric Bancel, Bruno Rouzeyre, Nicolas Bérard, and Michel Renovell. Scan design and secure chip. In *Proceedings of the 10th International On–Line Testing Symposium*, 2004.

[36] Greg Hoglund. Security band-aids: More cost-effective than s̈ecurec̈oding. *IEEE Software*, 19(6):56, 58, November/December 2002.

[37] Greg Hoglund and Gary McGraw. *Exploiting Software: How to Break Code*. Addison Wesley, February 2004. http://www.exploitingsoftware.com/.

[38] ISA. Integrating Electronic Security into the Manufacturing and Control Systems Environment. Technical Report ISA–TR99.00.02, Instrumentation, Systems, and Automation Society, April 2004.

[39] ISA. Security Technologies for Manufacturing and Control Systems. Technical Report ISA–TR99.00.01, Instrumentation, Systems, and Automation Society, March 2004.

[40] ISF. The standard of good practice for information security, January 2005. http://www.isfsecuritystandard.com.

[41] Jakob Jonsson and Burton Kaliski. Public-key cryptography standards (pkcs) #1 : Rsa cryptography specifications version 2.1. RFC 3447, Internet Engineering Task Force, February 2003. http://www.rfc-editor.org/rfc/rfc3447.txt.

[42] Burton Kaliski. PKCS #7: Cryptographic Message Syntax. RFC 2315, Internet Engineering Task Force, March 1998. http://www.rfc-editor.org/rfc/rfc2315.txt.

[43] Burton Kaliski. PKCS #5: Password–Based Cryptography Specification. RFC 2898, Internet Engineering Task Force, September 2000. http://www.rfc-editor.org/rfc/rfc2898.txt.

[44] Stephen Kent and Randall Atkinson. Security architecture for the internet protocol. RFC 2401, Internet Engineering Task Force, November 1998. http://www.rfc-editor.org/rfc/rfc2401.txt.

[45] Cetin Kaya Koc. High–Speed RSA Implementation. Published by RSA Laboratories, November 1994. ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf.

[46] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Lecture Notes in Computer Science*, 1666:388–397, 1999.

[47] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In *Advances in Cryptology, Proceedings of CRYPTO'96*, volume 1109 of *Lectures Notes in Computer Science*, pages 104–113. Springer, 1996.

[48] Paul C. Kocher, Ruby B. Lee, Gary McGraw, Anand Raghunathan, and Srivaths Ravi. Security as a new dimension in embedded system design. In *Proceedings of the 41th Design Automation Conference, DAC*, pages 753–760. ACM, June 2004.

[49] John Kohl and B. Clifford Neuman. The kerberos network authentication service (V5). RFC 1510, Internet Engineering Task Force, September 1993. `http://www.rfc-editor.org/rfc/rfc1510.txt`.

[50] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *Proceedings of the USENIX Workshop on Smartcard Technology, Chicago*, pages 9–20, May 1999.

[51] Philip Koopman. Embedded systems security. *IEEE Computer*, 37(7):95–97, July 2004.

[52] Hermann Kopetz. Fault Containment and Error Detection in TTP/C and FlexRay. Research Report 23/2002, Institute for Computer Engineering, August 2002.

[53] Sandeep Kumar and Thomas Wollinger. Fundamentals of symmetric cryptography. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 125–143. Springer-Verlag New York, Inc., 2006.

[54] Jed Lauand, W. Mark Townsley, and Ignacio Goyret. Layer two tunneling protocol – version 3 (l2tpv3). RFC 3931, Internet Engineering Task Force, March 2005. `http://www.rfc-editor.org/rfc/rfc3931.txt`.

[55] Marcus Leech, Matt Ganis, Ying-Da Lee, Ron Kuris, David Koblas, and LaMont Jones. SOCKS protocol version 5. RFC 1928, Internet Engineering Task Force, March 1996. `http://www.rfc-editor.org/rfc/rfc1928.txt`.

[56] Kerstin Lemke. Embedded security: Physical protection against tampering attacks. In Kerstin Lemke, Christof Paar, and Marko Wolf,

editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 207–217. Springer-Verlag New York, Inc., 2006.

[57] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secret of Smart Cards*. Springer, 2007.

[58] Akira Nagashima. Technologies for Achieving Field–Ubiquitous Computing. In *Proceedings of the 5th International Conference on Industrial Informatics*, page 18, 2007.

[59] Magnus Nystrom and Burton Kaliski. PKCS #10: Certification Request Syntax Specification. RFC 2986, Internet Engineering Task Force, March 1998. `http://www.rfc-editor.org/rfc/rfc2986.txt`.

[60] Aleph One. Smashing the stack for fun and profit. *Phrack*, 7(49):14, 1996. `http://www.phrack.org`.

[61] Andy Ozmet and Stuart Schechter. Milk or wine: Does software security improve with age. In *15th Usenix Security Symphosium*, 2006.

[62] Christof Paar. Embedded it security in automotive application - an emerging area. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 95–109. Springer-Verlag New York, Inc., 2006.

[63] Don N. Page. The lifetime of the universe. *Journal of the Korean Physical Society*, 49:711–714, 2006.

[64] Gurdeep Singh Pall, Bill Palter, Allan Rubens, W. Mark Townsley, Andrew J. Valencia, and Glen Zorn. Layer Two Tunneling Protocol L2TP. RFC 2661, Internet Engineering Task Force, August 1999. `http://www.rfc-editor.org/rfc/rfc2661.txt`.

[65] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One–Way Functions. *Science*, 297(5589):2026–2030, September 2002.

[66] David Powell and Robert Stroud. Conceptual model and architecture of MAFTIA. MAFTIA deliverable D21, January 2003. `http://www.maftia.org`.

[67] Jaziar Radianti and Jose J. Gonzalez. Understanding Hidden Information Security Threats: The Vulnerability Black Market. In *Proceedings of the 40th Hawaii International Conference on System Sciences*, 2007.

[68] Avni Rambhia and John C. Simmons. Building Secure,DRM-Enabled Devices. Presentation, May 2004. `http://www.microsoft.com/whdc/winhec/pres04-tech.mspx`.

[69] Srivaths Ravi, Anand Raghunathan, Paul Kocher, and Sunil Hattangady. Security in embedded systems: Design challenges. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(3):461–491, 2004.

[70] Svrivaths Ravi, Anand Raghunathan, and Srimat Chakradhar. Tamper resistance mechanisms for secure embedded systems. In *Proceedings of 17th International Conference on VLSI Design*, pages 605– 611. IEEE Computer Society Press, 2004.

[71] Stuart Schechter. Quantitatively differentiating system security. *The First Workshop on Economics and Information Security*, 2002.

[72] Bruce Schneier. Full disclosure and the window of exposure. Crypto-Gram Newsletter, Online, September 2000. `http://www.schneier.com/crypto-gram-0009.html`.

[73] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* Wiley, 2nd edition, 2006.

[74] Seth Schoen. Trusted Computing: Promise and Risk. Online, 2003. `http://www.eff.org/Infrastructure/trusted_computing/20031001_t`

[75] Kai Schramm, Kerstin Lemke, and Chridtof Paar. Embedded cryptography: Side channel attacks. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 187–205. Springer-Verlag New York, Inc., 2006.

[76] Hermann Seuschek. DPA–Analyse von Implementierungen symmetrischer kryptographischer Algorithmen. Master's thesis, Lehrstuhl für Datenverarbeitung, Technische Universität München, 2005.

[77] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[78] Richard Stallman. Can You Trust Your Computer? Online, 2002. `http://www.gnu.org/philosophy/can-you-trust.html`.

[79] William Thomson. On an absolute thermometric scale founded on Carnot's theory of the motive power of heat, and calculated from Regnault's observations. *Math. and Phys. Papers*, 1:100–106, 1848.

[80] Paulo Veríssimo. Intrusion tolerance: Concepts and design principles. a tutorial. Technical Report DI/FCUL TR02-6, Department of Informatics, University of Lisboa, 2002.

[81] Paul Watzlawick, Janet H. Beavin, and Don D. Jackson. *Menschliche Kommunikation*, chapter Pragmatische Axiome - ein Definitionsversuch, page 53. Verlag Hans Huber, Bern, 1967.

[82] Michael J. Wiener. Efficient DES key search, technical report TR-244, carleton university. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. 1996.

[83] Thomas Wollinger, Jorge Guajardo, and Christof Paar. Cryptography in embedded systems: An overview. In *Proceedings of the Embedded World 2003 Exhibition and Conference*, pages 735–744. Design & Elektronik, Nuernberg, Germany, February 2003.

[84] Thomas Wollinger and Sandeep Kumar. Fundamentals of asymmetric cryptography. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 145–165. Springer-Verlag New York, Inc., 2006.

[85] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, January-March 2004.

A Security Standards and Documents Overview

The first part of the chapter lists some important standards in the area. This section overlaps with the next one which focuses on certification of security.

A.1 International Standardization Organization (ISO)

The first standard on Informantion Security was published by the British Standards Institute BSI in 1995 under identification number ″BS 7799 Part 1″. It is defining a *′code of practice′* comprising statements of generally accepted good practice for maintaining information security rather than technical means. In 2000 it was adopted as ″ISO/IEC 17799″. Then in 2005 the standard ISO 27001 emerged from ″ISO/IEC 17799″ and ″BS 7799 Part 3″, representing the state–of–the–art. This new standard includes the possibility of certification of an organisation's IT–systems through an accredited certification body, i.e. an organisation accredited to perform such an assessment of management systems.

From July 2007 on, there are two standards concerning information security available from the ISO (quoted from the homepage[12]):

> ISO 27002 is a code of practice for information security. It details hundreds of specific controls which may be applied to secure information and related assets. It comprises 115 pages organized over 15 major sections.″
>
> ISO 27001 is a specification for an Information Security Management System, sometimes abbreviated to ISMS. It is the foundation for third party audit and certification. It comprises 34 pages over 8 major sections. ″

Both standards can be purchased online for approximately $299.

A.2 Bundesamt für Sicherheit in der Informationstechnik

The german Bundesamt für Sicherheit in der Informationstechnik (BSI) published the *″IT–Grundschutz Manual″*, which is compatible to the ISO standard 27001 (see Section A.1). It consists of three parts:

[12]http://17799.standardsdirect.org/

- BSI Standard 100–1 Information Security Management Systems (ISMS): Part one [19] addresses standard security measures, the solution of common security problems like misconfiguration, IT infrastructure and environment.
- BSI-Standard 100–2: IT–Grundschutz Methodology [20]: This part introduces a methodology aimed at producing an IT security concept for common IT applications and systems and establishing a continuous and effective IT security process.
- BSI-Standard 100–3: Risk Analysis based on IT–Grundschutz [21]: The last part covers issues for security requirements going beyond those defined for standard IT systems as described in the other two parts. It contains chapters for determination of additional threats, threat assessment, and handling risks.

The BSI defines the goal of the manual as follows:

> The aim of the IT–Grundschutz Manual is to achieve a security level for IT systems that is reasonable and adequate to satisfy normal protection requirements and can also serve as the basis for IT systems and applications requiring a high degree of protection. This is achieved through the appropriate application of organisational, personnel, infrastructural and technical standard security safeguards.

The standard is available in German and English language for download at the BSI's homepage[13].

A.3 Austria Secure Information Technology Center (A–SIT)

The A–SIT is an independent association and comprehends itself as center of excellence for IT–security. It's members are the Bundesministerium für Finanzen (BMF), Österreichische Nationalbank (OENB), and the Technical University in Graz. The organisation publishes the *Österreichisches Informationssicherheitshandbuch* [22].

The first part titled *"Informationssicherheitsmanagement"* comprises chapters to management of Information Security, development of an organisational security policy, risk assessment, implementation of a security policy, maintaining security during operation, and industrial security.

[13]`http://www.bsi.de/english/gshb/`

The second part titled *"Informationssicherheitsmassnahmen"* concerns itself with human and structural measures, the IT systems security, and disaster recovery and contingency planning.

The manuals can be download freely from the organisation's homepage[14].

A.4 Instrumentation, Systems, and Automation Society (ISA)

The organisation formerly known as ANSI and now called ISA deliverd in 2004 two standards for establishing, maintaining, and auditing information security in industrial applications.

The first report [39] titled *"Security Technologies for Manufacturing and Control Systems"* contains chapters on authentication and authorization technologies, access control technologies, ecryption technologies and data validation, auditing tools, and physical security controls.

The second report [38] titled *"Integrating Electronic Security into the Manufacturing and Control Systems Environment"* focuses on the process of implementing security measures in the industrial system. It encompasses chapters on risk assessment, how to define and deploy countermeasures, periodic audit, compliance measures, routine security reporting and analysis. Furthermore, it contains example sheets to facilitate these processes at a production site.

The standards are available for download or can be ordered. Each of them costs approximately $120.

A.5 FIPS 140-2: Security Requirements for Cryptographic Modules

This standard [29] is probably one of the best known standards on Information Security. It origins from the financial world, which was the first civil application with highest security requirements.

The standard defines the security requirements that must be satisfied by a *cryptographic module* used in a security system protecting unclassified information within IT systems. A cryptographic module in this context is *"the set of hardware, software, and/or firmware that implements approved security functions (including cryptographic algorithms and key generation) and is contained within the cryptographic boundary."*

Four increasing levels of requirements for physical security are available:

[14] `http://www.a-sit.at`

Level 1: No physical security mechanisms are required in the module beyond the requirement for production–grade equipment.

Level 2: Tamper evident physical security or pick resistant locks. Level 2 provides for role-based authentication. It allows software cryptography in multi-user timeshared systems when used in conjunction with a C2 or equivalent trusted operating system.

Level 3: Tamper resistant physical security. Level 3 provides for identity-based authentication.

Level 4: Physical security provides an envelope of protection around the cryptographic module. Environmental failure protection and testing techniques (EFP and EFT) must be applied.

The National Institute of Standards and Technology (NIST) issued the FIPS 140 Publication Series for both hardware and software components. It can be downloaded from tthe NIST homepage[15].

A.6 Information Security Forum (ISF)

The ISF is an organisation comprising 270 leading companies and public sector organisations, whose membership is evenly spread over a wide range of sectors. It publishes *The Standard of Good Practice for Information Security"* [40], which understands itself as complementary to other standards such as ISO (see Section A.1).

The focus is on Information Security for companies and businesses. It defines five key aspects of Information Security: Security Management (enterprise-wide), Critical Business Applications, Computer Installations, Networks, and Systems Development.

The standard is available for free download from the organistion's homepage[16].

A.7 Common Criteria (CC)

The Common Criteria is an international approach to security evaluation. It draws from the strength of previous evaluation schemes. Participants are governmental or related organizations.

[15]`http://csrc.nist.gov/cryptval/140-2.htm`
[16]`http://www.isfsecuritystandard.com/`

Two kinds of evaluations are possible, for products and for *protection profiles* (PP). *A* CC PP *is an implementation–independent set of security requirements for a category of products or systems that meet specific consumer needs.* It provides a through description of threats, environmental issues and assumptions, security objectives, and CC requirements for a familiy of products. In order to evaluate a single product, a so–called *security target* has to be either derived from a PP or developed on its own, which is a set of requirements and specifications for a particular product.

The homepage[17] contains a lot of useful information on members, evaluated products, protection profiles, national security authorities, and other information.

The seven *Evaluation Assurance Level*s (EAL):

EAL *1*: *Functionally Tested* Analysis of of security functions based on functional and interface specifications. It is applicable to systems where security threats are not serious.

EAL *2*: *Structurally Tested* Analysis of of security functions including the high–level design. Evidence of developer testing based on functional and interface specifications, independent confirmation of developer test results, strength–of–functios analysis, and a vulnerability search for obvious flaws must be provided. This level is applicable in absence of ready availability of the complete development record as, e.g., when securing legacy systems.

EAL *3*: *Methodically Tested and Checked* Same evaluation criteria as in EAL2 and, in addition, the use of development environment controls and configuration management. This level provides a moderate level of security.

EAL *4*: *Methodically Designed, Tested, and Reviewed* This level requires a low–level design, complete interface description, and a subset of the implementation for the security function analysis. Additionally, an informal model of the product or system security policy. This level targets systems with a moderate to high security requirement. Examples for EAL4 products are operating systems like Novell NetWare, SUSE Linux Enterprise Server 9, Windows 2000 Service Pack 3, Red Hat Enterprise Linux 5, and Trusted Solaris.

EAL *5*: *Semiformally Designed and Tested* A formal model, a semiformal functional specification, a semi–formal high–level design, and a semi–formal correspondenceamong the different levels of secification are

[17]`http://www.commoncriteriaportal.org`

all required. This level is applicable for smart cards and multilevel secure devices.

EAL 6: *Semiformally Verified Design and Tested* Semi–formal low–level design and structured presentation of the implementation in addition to the inputs for the security analysis in EAL6.

EAL 7: *Formally Verified Design and Tested* The highest level of evaluation requires a formal representation of the functional specification and a high–level design, and formal and semiformal demonstrations must be used in correspondence. The only product that has been evaluated at EAL7 is the Tenix Interactive Link Data Diode Device, which is a hardware device that allows data to travel in one direction, while preventing data from travelling in the opposite direction.

B Primality Testing

Primality testing is concerned with testing if a given number n is a prime number. Prime numbers are a subset of the natural numbers and a prime is charaterized by the property that its sole divisors are one and itself.

On the contrary to prime numbers, all other numbers are called composite, because they can be written as a composition of primes. The task of primality testing is to create a sieve for the set of natural numbers, which separates composites and primes. There are many ways to build such a sieve, which differ in efficiency of computation and probability of correctness.

Sieve of Eratosthenes This is the oldest known method to compute primes. It starts by making a list of all the natural numbers less than or equal to n, which is the number to test. Starting with two, strike out the multiples, continue with succeeding not striked number, then the numbers that are left are the primes. Actually, this method is fast, but its implementation uses lots of memory. It's running time is denoted with $O(n)$, while the sieve of Atkin, an improvement to Eratosthenes' algorithm, runs in sublinear time $O(\frac{n}{log\ logN})$.

As more recent algorithm aim at testing large numbers, the Sieve of Erstosthenes algorithm can still perform better on computing primes up to a certain bound for smaller numbers (e.g. 200). Thus, it can improve performance by pre-computating a list of primes and then checking whether the input number is divisible by any prime from the list before continuing with a more sophisticated algorithm.

Rabin–Miller. The Miller–Rabin test is a probabilistic primality test. A found prime is with a small probability a composite number. The error rate is smaller the 0,5 %, thus negligible. The running time of this algorithm is $O(k \times log^3 n)$, with k being the number of random values used for testing. The probability of an error may be reduced by repeating the test more often. The algorithm relies on modular exponention (see **??**). FFT-based multiplication can push the running time down to $O(k \times log^2 n)$. An implementation of this algorithm is shipped with some standard mathematics libraries, e.g., the GNU MP Bignum Library (GMP).

AKS The AKS algorithm [2] is the most recently discovered method to do primality testing. The algorithm works deterministic and executes in polynomial time. The running time of the algorithm is given in the paper with $O(log^{12+\epsilon}(n))$, but since the publication of the paper lots of improvements were made, which decrease the execution time significantly. A good guide for an implementation is given in the paper in [25], which takes the new insights on the algorithm concerning efficiency into account.

C Chipset manufacturers and TPM functionality

CORPORATION	TPM NAME	DESCRIPTION
Intel	Trusted execution technology	*…provides hardware-based mechanisms that help protect against software-based attacks and protects the confidentiality and integrity of data stored or created on the client* PC.
AMD	Secure Virtual Machine (SVM)	*…enhances virtualization and provides efficient virtual machine memory isolation for improved security and support of virtual users.*
Transmeta	Transmeta Security eXtensions (TSX)	*Transmeta will provide interfaces to this hardware encryption engine via cryptographic instructions that are an extension to the x86 instruction set architecture. Named the Transmeta Security Extensions* (TSX)*, these instructions will support key preparation and the* DES, DES– *and Triple–*DES *ciphers*
IBM	Embedded Security Subsystem	*…is a chip on the ThinkPads mainboard that can take care of certain security related tasks conforming to the* TCPA *standard.* It provides basic TPM functions.
	ThinkVantage Technology.	*…can help protect yoour systems and data from unauthorized access and certain types of unexpected accidents.*
Winbond Corporation	SafeKeeper.	…is a fully compatible with the TCG 1.2 specification, features true interoperability, and best–in–class performance across PC platforms.

Phoenix Technologies	Core Managed Environment (CME)	*. . . is a standards–based set of enabling technologies and applications built into the foundation of PCs, servers, and other digital devices. Independent of the operating system (OS), Phoenix CME technology enables manufacturers to deliver products with core system management capabilities that are always available and always secure.*
Fujitsu	FirstWare Vault	*. . . is a combination of a host protected area (HPA) – a special, protected environment on the computer for storing data – and a Windows application for accessing "virtual CD" data, placed by the manufacturer into the HPA.*
Hewlett Packard	ProtectTools	*. . . is a hardware security chip, called the Trusted Platform Module (TPM) that integrates the core elements of trust into the subsystem.*

Chip implementations of TPMs are available from several manufacturers. See Table reftab:tpman for a product table.

Manufacturer	Model name	Interfaces	EAL
Atmel	AT97SC 320	LPC, TwoWire	3+
Infineon	SLD 9630TT1.2	LPC	4
National-Semiconductors	PC21100	LPC	4
Sinosun	SSX35	LPC, ISO 7816	3
STMicroelectronics	ST19WP18-TPM	LPC, ISO 7816	3
Winbond	WPCT200	LPC	–

Table 9: Trusted Platform Module (TPM) product table

Hardware certification of TPMs is performed through the *Evaluation Assurance Level* (EAL). according to the *Common Criteria* (CC), an international standard (ISO/IEC 15408) for computer security. The EAL consists of seven increasing assurance levels. Higher grades indicate a degree of confidence that the system's principal security features are reliably implemented.

D Acknowledgements

This work has been fully supported by the TTTech Computertechnik AG, A – 1040 Vienna, Austria. The author thanks Hermann Kopetz for the opportunity to report on this topic and his collegues on the institute for the many fruitful discussions.

Index