Chapter 5
A Security Primer

Manfred Jantscher[1], Raja Ghosal[1], Alfio Grasso[1], and Peter H. Cole[1]

[1] The School of Electrical and Electronic Engineering,The University of Adelaide, Adelaide SA 5005, Australia.{manfred, rghosal, alf, cole}@eleceng.adelaide.edu.au

Abstract. This paper presents an overview of modern cryptography. As the title of this paper suggests this paper is a primer, and provides background information that can assist other researchers in further study, and in developing security mechanisms suitable for inclusion on RFID tags.

Keywords: Security, Authentication, RFID.

1 Introduction

Information technology security or cryptography, nowadays, is a well established area of computer science. Modern applications of cryptography allow us to transfer confidential data over insecure channels, take care of our bank transactions on transfer line using of-the-shelf computers, and sign obligatory contracts over the Internet. These applications are based on cryptographic building blocks, so-called primitives that provide certain desirable services like encrypting and decrypting messages.

Unfortunately, often it is not possible to straightforwardly implement cryptographic primitives in RFID systems because tags normally are very restricted in available power and, because they have to be cheap, chip area plays a crucial role. A demand for new, lightweight cryptographic primitives arises offering security services tailored to the requirements of RFID systems while considering their resource constraints.

This paper is structured as follows: Section 2 discusses information technology security in general presenting a state-of-the-art overview on cryptography. After dealing with desirable security services, common attacks, and security models, Section 3 covers cryptographic primitives divided into the three main groups unkeyed, secret-key and public-key primitives. Finally, a conclusion summarizes the main features of the work.

2 Information Technology Security

In the age of information processing, Internet and digital communication obviously there is a strong need for information technology security. Exchange of confidential messages, online-transfer of money and access to information services are just a few examples of procedures that rely on the security of computer systems and networks. Therefore, information systems and information processed by information systems have to be protected. Cryptography is the science that deals with protection of information [1]. In this section we cover the basics about information technology security and cryptography, and its appropriateness to RFID systems and in particular implementation of cryptographic primitives on an RFID Tag. Possible cryptographic solutions may be applied to the anti-counterfeiting applications for a secure supply chain. The reader may be adverted to [2] for definition of terms used in this section. This may be especially helpful if terms are used prior to their actual definition which is sometimes unavoidable.

2.1 Model

Although cryptography is much more than just encryption, encryption could be seen as the primary goal. Thus, to start to describe information technology security, the following simple model of Figure 1 is provided.

Alice and Bob are two parties who, despite or perhaps because of their somewhat pneumatic appearance, trust each other. They want to communicate using the channel, in their case a river, which they know to be insecure. In more real applications they can be human beings equipped with computing equipment, and communicating with electrical signals. In the above example, Alice sends a message which holds information designated for Bob. Eve is an unknown third party who also has access to the channel. The model could be interpreted in two different ways, either as transmission in space, where Alice and Bob sit at different places, or as transmission in time, where e.g. Alice stores a message onto the hard disk of a computer and Bob recalls it at a later time. Given this model, there are a number of concerns. Can Alice be sure that Bob is the only one who can read her message? Can Bob be sure that the message was sent by Alice and if it was, can he be

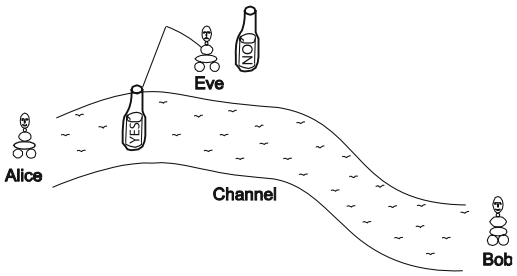

Fig. 1 Encryption model.

sure that it wasn't altered? Isn't Bob able to deny that he has received the message and isn't Alice able to deny that she has sent the message? These questions lead us to the services that are required for information technology security.

2.2 Security Services

As described in the preceding section, there are a number of security concerns when thinking about computer systems and digital communication. Therefore, a number of services have to be provided in order to enhance information technology security. The following paragraphs are dedicated to the most important of them [3], [4].

- **Confidentiality** ensures that only authorized parties are able to understand information.
- **Authentication** refers to the ability for a party to be sure the received information is from the source it claims to be from.
- **Integrity** assures that a message was not altered on the way to its recipient. Hence, it provides the recipient with the certainty to know what it has received is what was sent by the trusted origin of the message.
- **Non-repudiation** ensures that neither the sender is able to deny that it has sent the message nor the receiver is able to deny that it has received it. Therefore, it provides a proof of transmission and reception of a message.
- **Access Control** refers to the ability to restrict and control access to a system.
- **Availability** provides means to ensure that a system is available whenever needed. Thus, availability services guard systems from attacks against loss or reduction in availability.

Most of these services might be achieved by applying an appropriate cryptographic tool, like an encryption algorithm or a cryptographic hash function, or a series of those. If a series of tools has to be applied in a well-defined way, this way is referred to as a cryptographic protocol. Before we will discuss the main types of cryptographic tools we would like to spend a few words about attacks and security models.

2.3 Attacks

Modern cryptography tools (primitives) face a variety of attacks they have to withstand. Before classifying these attacks a basic principle of state-of-the-art cryptography has to be explained. According to Kerckhoffs' principle, stated in the nineteenth century by Auguste Kerckhoffs, the security of a cryptosystem must not depend on the secrecy of data independent details about the system [3]. Data independent details of a cryptographic system are the algorithm and its implementation. Therefore, attacks on modern primitives often aim at the recovery of plaintexts from ciphertexts or even worse on the recovery of secret keys [5].

Attacks may be classified into passive and active attacks [4]. Passive attacks denote monitoring of channels or side channels, but not alteration of messages. Obviously monitoring of a channel includes directly listening to data being transferred. Monitoring of side channels, is listening to effects that come along with the activities on the channel like electromagnetic emanation or current consumption [6]. Passive attacks on encryption schemes may be further subdivided into the following kinds of attacks [4]:

- **Ciphertext-only attack**: The attacker tries to recover plaintext or the secret key just by analysing the corresponding ciphertext.
- **Known-plaintext attack**: By analysing a given block of plaintext and the corresponding ciphertext the attacker tries to extract useful information for the recovery of plaintext encrypted in different ciphertexts or the secret key.
- **Chosen-plaintext attack**: Given the attacker is able to choose plaintexts and generate corresponding ciphertexts it tries to extract useful information in order to recover new plaintexts from new ciphertexts or even may try to extract the secret key.
- **Adaptive chosen-plaintext attack**: The attacker tries to recover plaintext or the secret key by subsequently applying chosen-plaintext attacks where the choice of the plaintext for later attacks depends on the outcome of prior attacks.
- **Chosen-ciphertext attack**: This attack is based on the assumption that the attacker, for a limited amount of time, is able to access means to decrypt ciphertexts. Therefore, we assume the means to be a black box which is able to decrypt a limited number of ciphertexts. The attacker chooses ciphertexts and analyses the corresponding plaintexts in order to gain useful information for the later purpose of recovering plaintexts from ciphertexts or of recovering the secret key without the availability of the black box.
- **Adaptive chosen-ciphertext attack**: As with the adaptive chosen-plaintext attack the attacker subsequently applies chosen-ciphertext attacks, with ciphertexts for later attacks depending on the outcome of prior attacks.

Although we mentioned the attacks above are aimed at encryption schemes, most of them are also applicable for other primitives like cryptographic hash functions or digital signature schemes.

Active attacks as opposed to passive attacks are based on alteration of data transmitted over a channel or alteration of computation in a device. The later is also referred to as fault attack [6]. Note that fault attacks and side-channel attacks are sometimes denoted as implementation attacks since they do not aim at the cryptographic algorithm but its specific implementation.

As mentioned above, security services might be based on primitives or on cryptographic protocols. Cryptographic protocols face yet another type of attack – so-called protocol attacks. The following list names the most important of them [4]:

- **Known-key attack**: Based on the knowledge of prior keys the attacker might try to obtain new keys.
- **Replay attack**: The attacker who is able to record a series of messages exchanged by the trusted parties replays part of it or the complete series at a later time.

- **Impersonation attack**: In this case, the attacker somehow assumes the identity of one of the trusted parties.
- **Dictionary attack**: This is a well known attack usually applied on password schemes. The adversary somehow manages to test a very large number of probable passwords with the intention guessing the right one.

2.4 Security Models

In order to describe the security of cryptographic primitives there are so-called security models. The following paragraphs describe the most important of them [1].

- **Unconditional security** also known as perfect secrecy is the non-plus-ultra security model. It assumes unrestricted computational power of the adversary. Therefore, for a cryptographic primitive to fall into this category there must not be an algorithm for breaking it, irrespective of the computational power available. An example of a simple primitive offering unconditional security is the one-time pad. In order to generate the ciphertext a plaintext is XOR-ed with a unique secret key of the same length as the plaintext. Because of the possible large key sizes such systems are impractical for conventional message encryption. However, there may be applications in systems with small information sizes like RFID.
- **Computational security** assumes polynomial computational power of the adversary. Therefore, a cryptographic primitive is assumed to be computationally secure if there is no algorithm known to break it within polynomial time. Modern primitives are supposed to fall into this category.
- **Practical security** also refers to the computational power of the adversary. However, as opposed to computational security there are no relative bounds. Instead, for a primitive falling into this category there must not be a breaking algorithm which requires less than N operations. The number of operations N is chosen sufficiently high. Modern cryptographic primitives typically offer practical security.
- **Provable security** means that it is possible to show that the complexity of breaking a primitive is equivalent to solving a well known supposedly hard mathematical problem like the integer factorization problem. Typical cryptographic primitives based on public keys fall into this category.

3 Cryptographic Primitives

So far, we discussed why information technology security is important, what potential threats are and how the security of cryptographic tools can be classified. What was not covered yet are cryptographic tools themselves, the primitives that provide us with the required services discussed above.

Table. 1. Some useful cryptographic primitives.

Some useful cryptographic primitives	
Un-keyed primitives	Hash functions One-way functions Random sequences
Secret-key primitives	Secret-key ciphers Message identification codes Identification primitives
Public-key primitives	Public-key ciphers Digital signatures

Cryptographic primitives may be separated into three large groups: un-keyed primitives, secret-key primitives and public-key primitives. Each of these groups may further be subdivided into primitives that serve different purposes. Table 1 shows a classification of some useful cryptographic primitives [4]. Each of these groups is further discussed the following sections.

Important differences between cryptographic primitives include the level of security, basic functionality, methods of operation, performance and the ease of implementation. Basic functionality deals with the objectives discussed in Section 2.2 whereas the methods of operation regard to specific ways primitives can be used, e.g. for encryption or decryption. Ease of implementation refers to both software and hardware environments [4].

3.1 Un-keyed Primitives

Un-keyed primitives, as the name betrays, are cryptographic tools that are not based on any keys at all. Examples for un-keyed primitives are cryptographic hash functions, one-way functions and random number generators. Taking into account Kerckhoffs' principle, because they are not based on keys, they do not fulfil security objectives on their own but often are part of a security system or a cryptographic protocol.

3.1.1 Hash Functions

"A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values."[4]

The above definition is very basic. For hash functions used as cryptographic tools a set of requirements has to be fulfilled. First of all, it has to be a one-way function, which means, given a message m it should be easy to calculate the hash-value $h(m)$ but it should be computationally infeasible to do the inverse operation namely to find a message m given the hash value $h(m)$ so that $m = h(m)$. Second, good hash functions should be collision resistant. Theoretically this would mean that there are no two messages *m1* and *m2* so that $h(m1) = (m2)$. However, since there are an infinite number of inputs but a finite number of outputs collisions are unavoidable. Therefore, in

practice collision resistance means it should be computationally infeasible to find collisions. The third important requirement says that the hash-function should be a random mapping which in practice means it should be computationally infeasible to distinguish a hash-function from a random mapping [7].

Basically there are two types of attacks on hash functions, collision and pre-image attacks. Collision attacks in parallel try to find two different messages *m1* and *m2* that lead to the same hash value $h(m1) = h(m2)$. If we consider digital signatures (see Section 3.4.2) where normally hash values of messages are signed for efficiency reasons this leads to a serious security threat. By deliberately designing two messages with the same hash value a digital signature for one message automatically is valid for the second message. Pre-image attacks try to construct a message m that leads to a given hash value $h(m)$.

Hash functions are very important primitives in practice. They are used whenever fixed length values are required instead of arbitrary length messages. They may also be used to generate various pseudorandom keys from a single input.

Un-keyed hash functions are often called modification detection codes (MDC) because of their ability to detect whether a message has been altered. Well known MDCs used in practice are MD5 and SHA-1. MD5 has an output length of 128 bits. The best known attack recently was described by Vlastimil Klima and is able to find MD5 collisions in about one minute using a state-of-the-art notebook computer (Intel Pentium 1.6 GHz) [8]. SHA-1 has an output length of 160 bits. The best known attack, illustrated on Bruce Schneier's Weblog on behalf of the author Xiaoyun Wang, shows a time complexity of 2^{63} which is even better than a brute force attack which would lead to a time complexity of 2^{80} for an output length of 160 bits [9]. Although, so far only collision attacks but no pre-image attacks are known, MD5 and SHA-1 cannot be seen as practically secure any longer. Hence there is a need for new hash functions. The US government standards agency NIST (National Institute of Standards and Technology) in 2001 published a new group of SHA algorithms collectively known as SHA-2. This group includes algorithms with output lengths of 256, 384 and 512 bits. [7]. So far no practical attacks are known for SHA-2.

3.1.2 One-Way Functions

As already mentioned with hash functions, one-way functions are mappings $f: X \rightarrow Y$ which are easy to compute but hard to invert. Expressed in a more formal way this means a polynomial time algorithm exists for computation but no probabilistic polynomial time algorithm for inversion succeeds with better than negligible probability [10].

There are special one-way functions called trapdoor functions with the additional property that given special information, called the trapdoor information, it is possible to calculate the inversion.

One-way functions and trapdoor functions are very important primitives in cryptography and a lot of other primitives are based on them. Examples would be hash functions which are based on one-way functions or public-key cryptography which is based on trapdoor functions.

In more detail, a trapdoor function is defined as a function that is easy to compute in one direction, yet believed to be difficult to compute in the opposite direction (i.e. finding the inverse) without special information, called the "trapdoor". We define "difficult" or "infeasible" in this section to mean computationally intractable, i.e. not possible to perform in a reasonable amount of time, e.g. one year, based on current state of technology. Trapdoor functions are widely used in cryptography.

To provide an illustration of a trapdoor function we will use as a context the RSA encryption system.

In RSA [4] the encryption operation performed is $c = m^e \bmod (n)$ (were n is the product of two large primes p and q) with the encryption key e and the modulus n (and the encryption formula) being disclosed, (and the primes p and q not being disclosed). We observe here for use in the discussion below that finding the primes p and q from their publicly disclosed product is believed to be infeasible when the primes are large. Here the ciphertext c is regarded as a function of the plaintext m.

The RSA problem, i.e. finding the inverse, is defined as taking the e^{th} roots modulo a composite number n. It is regarded as infeasible to solve except in the circumstances described below.

The inverse of the encryption, i.e. finding the plaintext m from the ciphertext c, can be performed if we have some additional information called trapdoor information. It is done in practice in the RSA system by using a decryption key d and the formula $m = c^d \pmod{\text{n}}$, but finding the decryption key d from the publicly disclosed e and n is believed to be difficult to the point of being infeasible.

The decryption key d (and the decryption formula) could be regarded as the trapdoor. The two large primes p and q could alternatively be regarded as the trapdoor. The path to find d from them is tortuous, and will not be described here, but is feasible to traverse.

It should be mentioned that a rigorous justification of the existence of one-way functions is an open problem in theoretical computer science.

3.1.3 Random Number Generators

The third important un-keyed primitive in cryptography are random numbers or random number generators respectively. Many keyed primitives or even cryptographic protocols are based on random number sequences. Examples of use are keys used in public-key cryptography, session keys often used in secret-key cryptography or random sequences (called nonces) in cryptographic protocols.

Before looking at sources of random numbers we should discuss what is randomness? In general, in order to be able to talk about randomness a context has to be defined [4]. For example, we cannot talk about a random number 7 in general but 7 could be a number randomly selected or generated out of a container holding the numbers 1 to 10 which is the context in this case. Furthermore randomness is based on uniform distribution and independence [3]. Uniform distribution means that there is equal probability for each of the numbers in our container to be selected, i.e. the probability to randomly select or generate 7 out of the container 1 to 10 is assumed to be 1/10. Independence refers to sequences of random numbers.

In order to talk about a sequence of random numbers there must not be any coherence in the sequence of these numbers, i.e. in the example with a container holding the numbers from 1 to 10, after randomly selecting 7, probabilities of selecting any of the numbers 1 to 10 should be the same.

The sources of ideal random numbers as discussed above, if there are ideal random numbers at all, are based on physical means. Examples could be gas discharge tubes or leaky capacitors. However, often they tend to be costly or slow in the generation of random numbers. Another interesting source are so-called physical unclonable functions (PUF). Based on manufacturing variations of ICs they might be used for secret key generation in electronic devices [11]. However, nowadays most of the random number generators used for cryptography are based on software algorithms. Since pure software algorithms are deterministic the sequences generated are not really statistically random. Therefore, algorithms based random number generators are called pseudo-random number generators and the sequences generated are called pseudo-random number sequences. The sequences generated are based on short random sequences called seeds. Most of the time, the generation algorithms are known but the seed is unknown. Sequences generated by good pseudo-random number generators will pass many tests of randomness and therefore they are applicable for cryptographic purposes.

3.2 Secret-Key Primitives

Secret-key cryptography denotes information technology security systems that are based on keys secretly shared between trusted parties. In the literature also the word symmetric-key cryptography can be found meaning the same systems. As Figure 2 shows, there are various applications of secret-key cryptography. The following sections will highlight the most important of them.

3.2.1 Secret-Key Ciphers

Ciphers deal with encryption and decryption of information. In the world of secret-key cryptography basically there are two types of ciphers in use today, block ciphers and stream ciphers. Both are based on the same model which we will explain first with the aid of Figure 2.

Assuming the basic encryption model already described in Section 2.1 (Figure 1) of this work, Alice wants to send a message *m* to Bob secretly, i.e. Eve should not be able to understand what she sees on the channel. Therefore, Alice and Bob agree on a secret key-pair (only known to them) for encryption and decryption before the

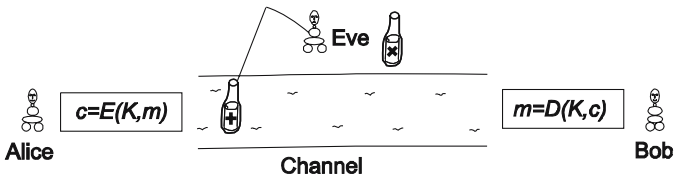

Fig. 2 Secret-key encryption.

actual start of the transmission. An important property of secret-key primitives is that the encryption key can be easily derived from the decryption key and the other way round. However, in nearly all modern secret-key encryption primitives the encryption and decryption key are the same and therefore are denoted as key K. Before sending the message m, Alice encrypts it applying an encryption function E and using the secret key K. The result of this operation is called ciphertext c which is transmitted over the channel. Therefore, $c = E(K, m)$. The size of the ciphertext is at least the size of the plaintext. This follows from the Pigeonhole Principle [12] which, converted to this scenario, states that if the output size of an encryption is smaller than its input size there must be necessarily 2 various inputs that lead to the same output which is not acceptable with encryption. On the other side, the size of ciphertexts could be bigger than the size of plaintexts if the plaintexts are padded before encryption for whatever reason. Bob, by applying the appropriate decryption function D and using the secret key K, recovers the original message m. Therefore, $m = D(K, c)$. Eve, who also received the ciphertext, is unable to understand it because she is missing the secret information, i.e. the key, to decrypt it. Assuming a good secret-key encryption primitive, it is computationally infeasible to recover the message from the ciphertext without knowing the key. What Bob will be able make of any bogus information that Eve may insert into the channel cannot be stated. A further question that is not covered in this scenario is how to exchange the key secretly. Unless Alice and Bob are unable to meet personally and need to exchange the key over a communication channel this may lead to a "Chicken and Egg problem" well known as the key distribution problem. However, there are smart solutions to this problem which may be gleaned in [4].

Note that although both concepts, encryption based on blocks and streams, also appear in public-key cryptography, the literature uses the names block cipher and stream cipher to denote secret-key encryption concepts.

Block ciphers form a special subset of secret-key ciphers where the message to be encrypted is divided into fixed length blocks. Each of these blocks, which are elements of the set of plaintexts, is transformed into an element of the set of ciphertexts. This happens under the influence of the secret key [1]. There is no change of size of the blocks during the transformation, i.e. ciphertext-blocks are of the same size as plaintext-blocks. The encryption is reversible, which means given the secret key it is also possible to recover plaintext blocks from ciphertext blocks. Block ciphers may be applied directly to messages with lengths equal to the size of a single block. If the size of the message is smaller than the block size padding is applied, i.e. additional symbols are added at the end of the message to fit the block size. Several padding techniques exist that allow distinguishing between message and padding. There are so-called modes of operation for messages longer than the block size [7].

Examples for block ciphers in use today are DES (data encryption standard), Triple DES (3DES) and AES (advanced encryption standard) [5]. DES was invented nearly 30 years ago and was one of the most widely used secret-key encryption algorithms. However, it is based on very short 64 bit blocks and 56 bit keys which, because of the increasing computational power available, seem not to be appropriate any more. There have been successful exhaustive key search attacks on DES already. Therefore, Triple DES (3DES) has been invented where DES is

applied three times with different keys. Therefore, it has a key size of 168 bits but inherits the disadvantages of DES like the small block size of 64 bits [7]. Triple DES is sometimes used because it offers more security compared with DES and for legacy reasons (Triple DES can be executed on DES hardware). However, in the meantime AES was invented to replace DES and 3DES. It is based on a block size of 128 bits and a selectable key size of 128, 192 or 256 bits. So far, there have been no successful attacks to the AES algorithm apart from side channel attacks which are implementation attacks rather than attacks on the algorithm [13].

Stream ciphers can be understood as block ciphers with block size one and the additional feature that the encryption transformation changes with every symbol processed. An advantage of stream ciphers is that they may have limited or no error propagation which means they may be able to deal with flipped bits or even with missing or inserted bits depending on their specific implementation. Therefore, often they are a good choice if errors are highly probable in transmissions.

A very simple example should clarify the operation of stream ciphers. The so-called Vernam Cipher is a stream cipher for binary streams [4]. The inputs are a binary message stream $m_1m_2m_3\ldots m_t$ and a so-called binary key stream $k_1k_2k_3\ldots k_t$. The binary key stream is a random binary sequence of the appropriate length, that is the length of the message stream. For encryption each binary symbol of the message stream is XOR-ed with the corresponding binary symbol of the key stream, that is $c_i = m_i$ XOR k_i. Obviously, to decrypt the ciphertext, the process has to be repeated, i.e. $m_i = c_i$ XOR k_i. The Vernam Cipher is exactly what we called one-time pad in Section 3.2.3 provides unconditional security (assumed the key stream is a truly random sequence with the same length as the message) but is hardly used in practice because of the large key size. However, stream ciphers used in practice are based on the one-time pad with the difference that the key stream is generated from a short random sequence by a deterministic algorithm. Thus, only the short random sequence has to be exchanged by the trusted parties.

As discussed above, practical stream ciphers require generators to create pseudo-random key-streams based on keys short enough to be exchanged conveniently. Often so-called linear feedback shift registers (LFSR) are used for this purpose. They exist of a series of delay blocks which are most often initialized with the secret key. Triggered by a clock signal the contents of the delay blocks are shifted. The input of the first delay block is a XOR of a subset of the delay blocks. Figure 3 shows an example of an LFSR.

Advantages of LFSRs are that they are easy to implement in hardware and software and they can produce sequences of large periods with reasonably good statistical properties [4]. The longest unique sequences are generated by so-called maximum length LFSRs. The period length of such LFSRs equals the maximum number of states that can be represented by a certain number of bits minus the zero

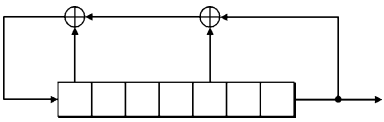

Fig. 3 Linear Feedback Shift Register (LFSR) [14].

state. Hence, a 3 bit maximum length LFSR has an output period length of $2^3 - 1 = 7$ unless it is initialized with 3 zeros.

However, with time mathematical methods have been developed to analyse LFSRs and they are not considered secure any longer [14]. Nevertheless, they form the building blocks of more secure key-stream generators in use today called nonlinear feedback shift registers (NLFSR).

NLFSRs generally consist of LFSRs as building blocks used in combination with methodologies that destroy the linearity of the output of simple LFSRs. In [4] three methods are discussed. So-called **nonlinear combination generators** apply a nonlinear function that combines the outputs of two or more LFSRs. An example for a nonlinear combination generator is the "Geffe generator" consisting of three maximum length LFSRs of pairwise relatively prime period lengths that are combined applying the function x_1x_2 XOR x_2x_3 XOR x_3. **Nonlinear filter generators** consist of just one maximum length LFSR which output is generated by a nonlinear combination of several stages of the LFSR. The generators discussed so far are clocked regularly, i.e. at each time step each LFSR is clocked. So-called **clock-controlled generators** introduce nonlinearity by clocking down-stream LFSRs based on the state of upstream LFSRs. An example is the "alternating step generator" which consists of three LFSRs. The output of this generator is the XOR result of the output of two LFSRs which are clocked depending on the output of the third LFSR. If the output of the third LFSR is 1, one of the other two LFSRs is clocked, if it is 0 the other one is clocked.

Although there is no real standard for stream ciphers so far, RC4 is most widely used and can be seen as de-facto standard [15]. As with most stream ciphers, it is based on one-time pad with a pseudo-random key-stream generator. Other than using LFSRs the key-stream generator is designed to be easily implemented in software. RC4 is in very heavy use today. It is the cipher used in WEP and WPA and may be optionally selected to be used in SSL (secure socket layer) and SSH (secure shell). Nevertheless, there are known attacks with the best of them being able to distinguish a random sequence from the pseudo-random sequence generated by RC4 given about one gigabyte of output data [16].

When comparing stream ciphers with block ciphers, there are clear practical advantages of stream ciphers. They are much easier to implement in software and hardware, generally they are faster, they do not require large memories to store blocks and they can deal with errors in the way that there is no error propagation. However, relatively few fully specified stream ciphers are published in the literature, and as opposed to block ciphers there are no standardised stream ciphers so far [4], [15].

The secret-key primitives described so far ensure that Eve is unable to understand, what is transmitted over the channel. Ciphers do not provide security against alteration of messages.

3.2.2 Message Authentication Codes

In order to be able to detect changes of messages transmitted over insecure channels so-called message authentication codes (MAC) or cryptographic checksums are used [3]. They can be understood as hash functions on data that includes a secret key.

Figure 4 shows the principle of MACs. Note that the example does not consider encryption of the message. Before sending the message, Alice generates a message authentication code mac applying a MAC-function which processes the message m and a secret key K. Then, Alice transmits both, the message and the message authentication code. Bob receives them and also generates the message authentication code using the same MAC-function which processes the received message and the shared secret key. Thereafter, Bob compares the MAC he generated with the MAC he received along with the message. If they match, because of the shared secret key, Bob knows that the message was not altered and he knows that the message was sent by Alice. In the Figure 4, Eve, who also receives the message and the MAC, can understand and alter the message but there is no way to change it prior to forwarding it to Bob without Bob knowing that something went wrong. Additionally, Eve cannot create messages and send them to Bob as if she were Alice [7]. Therefore, message authentication codes provide data integrity and data origin authentication [4].

In practice there exist several types of MAC-functions. They might be based on block ciphers, like the DES-CBC MAC, be based on stream ciphers, be constructed from un-keyed hash functions applying a secret key, like the MD5 MAC, or they might be based on one-time pad cipher [17]. Construction of MAC-functions from un-keyed hash functions means that the original algorithm, like MD5, is altered to incorporate a secret key into the compression function [4].

In practice there exist several types of MAC-functions. They might be based on block ciphers, like the DES-CBC MAC, be based on stream ciphers, be constructed from un-keyed hash functions applying a secret key, like the MD5 MAC, or they might be based on one-time pad cipher [17]. Construction of MAC-functions from un-keyed hash functions means that the original algorithm, like MD5, is altered to incorporate a secret key into the compression function [4].

3.2.3 Identification Primitives

One of the most important classes of primitives in today's computer systems are identification techniques which are sometimes also called entity authentication or identity verification techniques [4]. The purpose of entity authentication techniques is to allow one party to gain assurance about the identity of another party. Thus, entity authentication tries to prevent impersonation attempts. When compared with message authentication techniques, entity authentication techniques

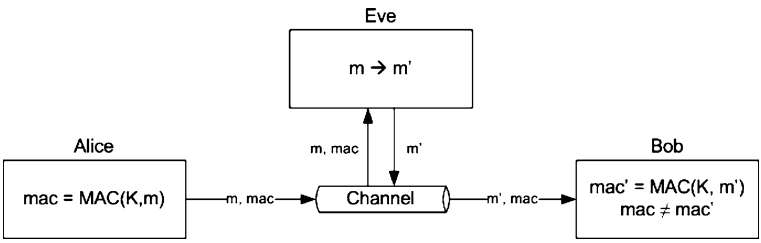

Fig. 4 Message Authentication Code (MAC). Modified from Figure 3.4 of Ferguson N. and Schneier B.: Practical Cryptography. Wiley Publishing, Indianapolis (2003).

typically involve no meaningful message but normally they are based on a real-time process, i.e. both parties are active at the same time).

Identification techniques may be divided into three categories. They may be based on something known by the identifying party, like a password or a secret key. They may be based on something possessed by the identifying party, like a magnetic stripe card or a smart card. Or, they may be based on something inherent to the identifying party, like voice, fingerprints or handwritten signature.

Typical applications of entity authentication include access control to resources, like information systems, sometimes accompanied with the need to track resource usage, e.g. for billing purposes.

Fixed passwords are a very simply scheme of entity authentication based on something known. They are shared secrets between users and an information technology system. For identification typically the system asks the user for a user-id and the appropriate password. If the data entered by the user matches the data stored in the system the identification was successful and the user is granted access. Different fixed password schemes may store the passwords either in plaintext, or in encrypted form. A major security problem of fixed passwords is the so-called replay attack, when an attacker records the password transmitted over a channel and replays it at a later time to be granted access to a system. For that reason fixed passwords often are refereed to as weak authentication [4]. Countermeasures include encryption of the channel or even better the use of one-time passwords.

There are different kinds of **one-time password** schemes. Two parties may either share a list of passwords using one after another, or they may sequentially update their password, or they may generate one-time passwords with the help of one-time functions [4].

The problem with both of the above described password schemes, fixed and one-time, is that the actual secrets are released by the party which tries to identify itself. Therefore, whenever an active attacker is able to gain access to the secret (e.g. the list of passwords in a one-time password scheme) it is subsequently able to impersonate the party to which the secret belongs to. **Challenge-response identification** schemes address this vulnerability. Rather than releasing the actual secret they generate a response which is based on the secret and a time-variant challenge. That is why challenge-response schemes are also known as strong authentication mechanisms [4]. The time-variant challenge is provided by the verifying party every time an unknown party wants to be identified. A challenge includes a so-called nonce being the time-variant parameter. Nonces could be, for example, pseudorandom numbers, sequence numbers, or time stamps. The use of nonces prevents replay attacks as described with fixed passwords. It should be mentioned that although challenge-response schemes often are based on secret-key cryptography there are also implementations based on public-key primitives.

3.4 Public-Key Primitives

In addition to secret-key primitives, so-called public-key primitives form the second large group of keyed cryptographic tools [4]. Public-key primitives are based

on key pairs instead of single shared keys. Each key pair is made up of a public key and a private key that are linked together mathematically. As their names betray, one of the keys has to be kept secret by the owner and the other one is shared publicly. Because public-key primitives are based on two different keys, they are often called asymmetric-key primitives. The main motivation to have public-key primitives is that with secret-key primitives each pair of trusted parties has to share one secret key [7]. Since this is very complex, if there is a larger number of parties involved, public-key primitives provide an appropriate solution because only one key has to be shared publicly for each party. On the other side, public-key primitives are less efficient and that is why there is still a need for secret-key primitives. The following sections describe the two most important applications of public-key cryptographic tools, public-key ciphers and public-key signature schemes.

3.4.1 Public-Key Ciphers

As already mentioned with secret-key primitives, ciphers deal with the encryption and decryption of messages. Figure 5 shows the basic principle of public-key encryption [7].

The key pair used in this example is the secret key of Bob (SBob) and the public key of Bob (PBob). As already mentioned earlier these keys are linked together mathematically in order to be able to use the public key for encryption and the private key for decryption. Therefore, a major premise for public-key cryptography is that it should be computationally infeasible to derive the secret key given a public key. Now, if Alice wants to send a message m to Bob secretly she encrypts it using the encryption function E under the influence of Bobs public key PBob. The resulting ciphertext c can be transferred over the insecure channel because only Bob is in possession of the secret key SBob which is necessary for decryption. Hence, in order to be able to read the received encrypted message c Bob decrypts it using the decryption function D under the influence of the secret key SBob. The example also shows Eve, the passive attacker, who is able to receive the public key PBob and the ciphertext c but cannot extract any useful information thereof.

It may now seem that the problem of secretly exchanging keys in secret-key cryptography is solved because public keys can be transferred over insecure channels. In fact, many practical systems are based on a mixture of secret- and asymmetric encryption. Often asymmetric-key encryption is used to agree on a shared

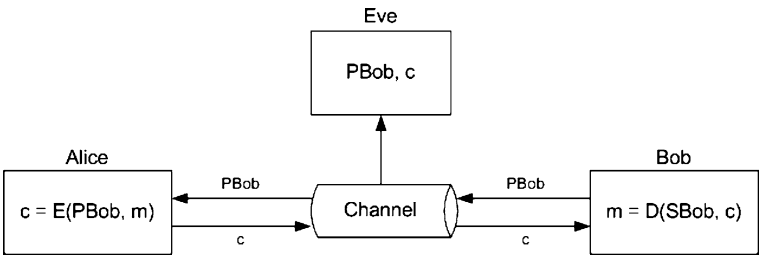

Fig. 5 Public-key encryption.

secret short term key (session key) which further on is used in secret-key encryption to exchange the actual information. The term session key is based on the fact that it is used for a limited time (the session) only. This approach combines the advantages of both, the publicly shared key of public-key cryptography and the efficiency of secret-key cryptography [4]. However, there is a remaining issue with public-key cryptography. Since public keys normally are exchanged over insecure channels an active attacker would be able to impersonate another party by providing a public key which seems to belong to this other party but actually is part of a key pair of which the attacker is in possession of the private key. Consequently, the attacker would be able to decrypt messages which were intended for the impersonated party. For that reason public keys often are exchanged using a so-called public key infrastructure (PKI) which solves the impersonation problem by issuing certificates. Certificates basically store identities and corresponding public keys. Each certificate is digitally signed (see next section) by a trusted third party who consequently prevents impersonation attacks as described above. Ferguson and Schneier ([7]) provide further information about PKIs for the interested reader.

Probably the most important public-key technique is the **RSA** cryptosystem [4], [5]. It was invented in 1978 by R. Rivest, A. Shamir and L. Adleman and is based on the well known integer factorization problem. The idea is to multiply two sufficiently large prime numbers p and q to obtain $n = p \times q$. Since it is believed (not proven) to be a hard mathematical problem to factorize n into its factors p and q, n is part of the public key and p and q are parts of the private key.

No practical attacks are known on the RSA cryptosystem provided it is based on sufficiently large keys (size of n). In respect to the computational power available, current references recommended to have keys of at least 2048 bits tending to 4096 or even 8192 bits for future applications.

The security of public-key cryptosystem is based on keys made up of very large integers. Usage of large keys, however, leads to less efficient execution of algorithms. Especially when considering mobile devices which are limited in computing power and energy efficient cryptosystems play an important role. Therefore, much attention is paid to **elliptic curve cryptography** (ECC) which offers public-key techniques that are much more efficient than traditional algorithms. ECC-calculations are based on points on elliptic curves. Since elliptic curves used in cryptography are defined in terms of modular arithmetic they only contain a limited number of points. The main operation used in ECC is called scalar point multiplication which means deriving a point P which satisfies the equation $P = kQ$ for a given point Q and a given integer k. This is a one-way function, i.e. the security of ECC is based on the so-called elliptic curve discrete logarithm problem (ECDLP), namely finding a k in $P = kQ$ for a given P and Q [1]. Solving the ECDLP is considered to be computationally infeasible if k is sufficiently large. Since the best known algorithm for solving the ECDLP shows exponential complexity key sizes of more than 224 bits are regarded to be sufficient large taking into account the computational power available at the moment [18].

3.4.2 Digital Signatures

Digital signatures are an essential cryptographic primitive in providing authentication, authorization and non-repudiation services. Signing primitives used in digital signatures provide a method to bind the identity of the signatory entity to the message to be transmitted.

While there are many digital signature algorithms, all of them are based on public key algorithms. That is there is some secret information that only the signing entity has knowledge of and there is some public information that allows any other entity to verify the signature. In this context the process of signing is called encrypting with the private key while the process of verification is called decrypting with a public key. Unfortunately the usage of terminology in signature schemes can be confusing when considered with that of public key ciphers.

Figure 6 shows the principle of a digital signature scheme [7]. We assume that Alice has already generated her key pair, i.e. SAlice, her private key, and PAlice, her public key. Whenever Alice wants to sign a message m she applies the signing algorithm S under the influence of her private key to this message with the result of a signature s. Subsequently, she distributes the message and the corresponding signature over the insecure channel. Bob, who wants to verify her message, applies the verification algorithm V under the influence of Alice's public key to the received message and the signature. The outcome of the verification could be either 'valid' or 'invalid'. 'Valid' would mean that the message was signed by Alice and it was not altered during its transmission. In our example Eve altered the message. Therefore, the result of Bob's verification is 'invalid'. Due to the fact that public-key techniques generally are less efficient, in practice often hash values of messages are signed rather than the actual messages.

One simple method of implementing a digital signature scheme is by using the RSA public key cipher where the encryption and decryption functions are both inverse operations of the other. This property is unique to the RSA cipher. Other examples are DSS (digital signature standard), ElGamal (named after its inventor Elgamal), and algorithms based on elliptic curve techniques. Obviously, a fact that all these algorithms share in common is that they are based on trapdoor functions with the trapdoor information being the private key [5].

The **ElGamal** digital signature scheme is partly based on the discrete exponential function and partly based on the Diffie-Hellman key agreement. The discrete exponential function was already covered in Section 3.1.2 of this work. Since the description of the ElGamal scheme goes beyond the scope of this work

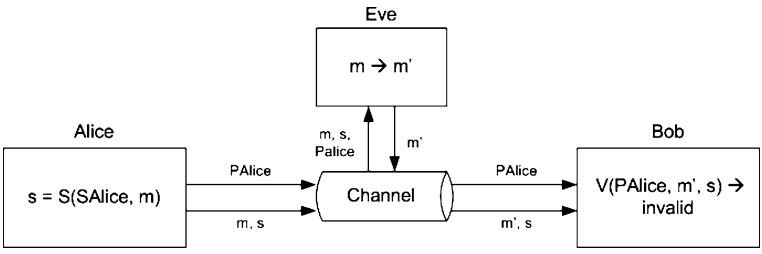

Fig. 6 Digital signature scheme.

the interested reader is adverted to [4] and [5] which provide detailed information about this topic and also cover the digital signature standard (**DSS**) which is another public-key technique used for digital signing and which is very similar to the ElGamal scheme.

3.5 Comparison of Secret-Key Primitives with Public-Key Primitives

Most of the advantages and disadvantages of secret-key primitives and public-key primitives were already discussed in the paragraphs above. This section is intended to summarize them and to provide a few additional notes.

The main advantages of secret-key primitives are that they are based on relatively short keys and that there are often efficient hardware implementations for them. Hence, they are applicable for processing large amounts of data. However, secret-key primitives come along with the disadvantage that a large number of keys have to be managed since each pair of trusted parties has to share an individual secret key. Additionally, it is considered as good practice to change the secret key regularly. This is to keep the amount of data being processed using a single key low in order to minimize the amount of input for potential attacks.

Public-key primitives, on the other side, have the advantage of easy key management. That is, only one key (the own private key) needs to be kept secret and public keys can be distributed over insecure channels. But most public-key techniques are based on very large key sizes which lead to less efficient execution of algorithms.

Considering the advantages and disadvantages of the large groups of primitives, today's cryptographic systems often are based on a mixture of both. These systems apply public-key cryptography to agree on so-called session keys which are shared secrets that are used for a limited time (the duration of a session) only. Subsequently, session keys are used in secret-key cryptography for processing the actual information. Therefore, the advantages of easy key management and efficient data processing have been combined in those mixed systems.

Elliptic curve cryptography, in future, may partially replace mixed systems since they combine the advantages of easy key management and fast data processing in one public-key scheme [19].

4 Conclusions

This paper provides an overview of state-of-the-art cryptography. Starting with information technology security it covers desirable security services, attacks and security models in general. Divided into the three groups un-keyed, secret-key and public-key, modern cryptographic primitives are presented.

Although standard cryptographic primitives offer aid to secure low cost RFID systems, resource constraints impede us from implementing most of the ordinary

cryptographic tools. RFID tags are very restricted in available operating power and chip size and unfortunately most of the cryptographic primitives require both. Hence, there is a need for new lightweight cryptographic primitives to be used in RFID technology.

Such lightweight primitives should be based on the well established knowledge on cryptography but tailor the services to the requirements and constraints of RFID.

References

1 Oswald, E.: IT security lecture notes. Institute for Applied Information Processing and Communications, Graz University of Technology, Austria (2005)
2 Grasso A. and Cole P., Definition of Terms Used by the Auto-ID Labs in the Anti-Counterfeiting White Paper Series. Available from http://autoidlabs.eleceng.adelaide.edu.au/static/Definition%20of%20Terms.pdf (7.06.2007)
3 Stallings, W.: Network and internetwork security: principles and practise. Prentice-Hall, New Jersey (1995)
4 Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. 2nd edn. CRC Press, Boca Raton (1997)
5 Delfs, H., Knebl, H.: Introduction to Cryptography: Principles and Applications. Springer-Verlag, Berlin, Heidelberg, New York (2002)
6 Oswald, E.: Introduction to Information Security lecture notes. Institute for Applied Information Processing and Communications, Graz University of Technology, Austria (2004)
7 Ferguson, N., Schneier, B.: Practical Cryptography. Wiley Publishing, Indianapolis (2003)
8 Klima, V.: Tunnels in hash functions: MD5 collisions within a minute. In: Cryptology ePrint Archive. Available from: http://eprint.iacr.org/2006/ 105.pdf (4.05.2006)
9 Schneier, B.: New cryptanalytic results against SHA-1. In: Weblog: Schneier on Security (2005). Available from: http://www.schneier.com/blog/archives/2005/08/ new_cryptanalyt. html (4.05.2006)
10 Goldreich, O: Foundations of Cryptography: Basic Tools. Cambridge University Press, Cambridge (2001)
11 Ranasinghe, D., Lim, D., Devadas, S., Abbott, D., Cole, P.: Random numbers from metastability and thermal noise. In: IEE Electronic Letters, Vol. 41, Iss. 16 (2005) 13–14
12 Grimaldi, R.P.: Discrete and Combinatorial Mathematics: An Applied Introduction. 4th ed. (1998) 244–248
13 AES Lounge: AES security. Available from: http://www.iaik.tu-graz.ac.at/research/krypto/AES/index.php#security (6.03.2006)
14 RSA Laboratories: What is a linear feedback shift register? Available from: http://www.rsasecurity.com/rsalabs/node.asp?id=2175 (4.04.2006)
15 RSA Laboratories: What is a stream cipher? Available from: http://www.rsasecurity.com/rsalabs/node.asp?id=2174 (4.04.2006)
16 Fluhrer, S.R., McGrew, D.A.: Statistical analysis of the alleged RC4 keystream generator. In FSE (2000) 19–30

17 RSA Laboratories: What are Message Authentication Codes? Available from: http://www.rsasecurity.com/rsalabs/node.asp?id=2177 (4.04.2006)
18 Chang, S., Eberle, H., Gupta, V., Gura, N.: Elliptic curve cryptography – how it works. Sun Microsystems Laboratories (2004). Available from: http://research.sun.com/projects/crypto/ (9.04.2006)
19 Gura, N., Shantz, S., Eberle, H., et al.: An end-to end systems approach to elliptic curve cryptography. Sun Microsystems Laboratories (2002). Available from: http://research.sun.com/projects/crypto (9.04.2006)