A PROVABLE SCHEME FOR HOMOMORPHIC OBFUSCATIONS IN SOFTWARE SECURITY

William Zhu and Clark Thomborson*
Department of Computer Sciences, The University of Auckland,
Private Bag 92019, Auckland, New Zealand
email: fzhu009@ec.auckland.ac.nz, cthombor@cs.auckland.ac.nz

ABSTRACT
Computer and communication industries develop so rapidly that the demand for software becomes larger and larger and the demand for software protections, such as copyright and anti-tampering defense, are more and more important to software users and developers. There are some technical measures for software protections, such as hardware-based protections, network filters, cryptosystems, etc. Software obfuscation is one of these measures. It protects software from unauthorized modification by making software more obscure so that it is hard for the potential attacker to understand the obfuscated software. In software obfuscation, we can obfuscate the control flow and the data flow of the software. In this paper, we explore an obfuscation based on residue number coding, which is a method widely used in hardware design, high precision integer arithmetic, and cryptography [1]. Recently, it has used as an obfuscation that encoded variables in the original program to hide the true meaning of these variables [2]. There is some discussion about the division of residue numbers in [2], but the technique proposed there is incorrect. In this paper, we establish a provable residue number encoding for software obfuscation, especially a provable algorithm for division by several constants in residue numbers.

KEY WORDS
software obfuscation, residue number coding, homomorphic obfuscation.

1 Introduction

The growth of computer and communications networks increases problems in the software security [3, 4]. Software obfuscation [5, 6, 7, 8, 9] is a branch of software security. It transforms a program into a new one that is harder to understand than the original one. In software obfuscation, variable transformation is a major method to transform software into a new one that is hard for attackers to understand. Residue number coding [1] is an approach used in hardware design, high precision integer arithmetic, and cryptography. It is also used for software obfuscation [2, 10]. In this paper, we propose the concepts of homomorphic obfuscations, a potential area for further exploration. Based on these concepts, we establish a sound grounding for residue number coding for software obfuscation. Especially, we use this to develop an algorithm for division by several constants, correcting an error in an earlier publication [2].

This paper is structured as follows. Section 2 describes basic concepts about residue numbers. Section 3 establishes a systematic theory for a solution to the division of residue numbers. Section 4 is the focus of this paper. It proposes a solution to the division of residue numbers which is easy to implement. This paper concludes in section 5.

2 Basic concepts about residue numbers

Let Z be the set of all integers, n a given positive integer. For any $x \in Z$, denote $[x]_n = \{y \in Z \mid y - x$ is divisible by $n\}$ and we call $[x]_n$ the residue class of x modulo n. We omit the subscript when there is no confusion. Let Z/nZ be the set of all these residue classes with respect to modulo n, where $Z/nZ = \{[0], [1], \ldots, [n-1]\}$.

For Z/nZ, we introduce a *natural order* relation $\leq$ among its members as $[0] \leq [1] \leq \ldots \leq [n-1]$

Z/nZ has three operations +, -, $\times$ defined as follows: for any two $[x], [y] \in Z/nZ$, $[x]+[y] = [x+y]$, $[x]-[y] = [x-y]$, $[x] \times [y] = [x \times y]$.

The product $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ also has three operations $+$, $-$, $\times$ defined as follows: for any two $([x_1]_{m_1}, \ldots, [x_k]_{m_k})$, $([y_1]_{m_1}, \ldots, [y_k]_{m_k}) \in Z/m_1Z \times \ldots \times Z/m_kZ$,

$$([x_1]_{m_1}, \ldots, [x_k]_{m_k}) + ([y_1]_{m_1}, \ldots, [y_k]_{m_k})$$

$$= ([x_1 + y_1]_{m_1}, \ldots, [x_k + y_k]_{m_k})$$

$$([x_1]_{m_1}, \ldots, [x_k]_{m_k}) - ([y_1]_{m_1}, \ldots, [y_k]_{m_k})$$

$$= ([x_1 - y_1]_{m_1}, \ldots, [x_k - y_k]_{m_k})$$

$$([x_1]_{m_1}, \ldots, [x_k]_{m_k}) \times ([y_1]_{m_1}, \ldots, [y_k]_{m_k})$$

$$= ([x_1 \times y_1]_{m_1}, \ldots, [x_k \times y_k]_{m_k})$$

*Research supported in part by the New Economy Research Fund of New Zealand.

```
mainProgram {
...
input(x);
licenseCheck(x);
...
}
void licenseCheck(int x){
  int c,d,e,f;
  if((c*x + d)/e == f)
     exit();
}
```

Figure 1. An unobfuscated program

```
obfMainProgram {
...
input(x);
int[k] xe =residue_encode(x, m_1,...,m_k);
licenseCheck(xe);
...
}
void obfLicenseCheck(int[k] xe){
  int[] temp = ce * xe + de;
  if(temp == ee * fe))
     exit();
}
```

Figure 2. An obfuscated version of the program in Fig. 1

In software obfuscation, sometimes we need to hide some constants. Generally, we will use some coding methods to encode these constants and decode them. A simple example is shown in Fig. 1 and 2.

To obfuscate the program of Figure 1, we choose suitable moduli $m_1, m_2, ..., m_k$. We then use these moduli to compute obfuscated constants ce, de, ee, and fe corresponding to the original constants c, d, e, and f. The obfuscated constants are k-vectors of integers. When the user of the obfuscated program of Fig. 2 inputs a valid license key x, this key is converted into k-vector residue format xe, and subsequent operations on xe are conducted in the residue-encoded format.

As said in [6], software obfuscation makes a program hard to understand, but, attackers can still find out its meanings with sufficient expertise and time. Additional protection can be obtained by additional layers of obfuscation, making it more difficult for an attacker to discover the values of $m_1, m_2, ..., m_k$; de-obfuscating the sequence of arithmetic operations to discover the acceptance function for keys x; and then inverting this function to develop a ”cracked” key-generator for alternative acceptable keys x'. Additionally, it would be important to tamper-proof or obfuscate the conditional branch and the exit call, to prevent an adversary from ”cracking” the program itself. See [5] and [11].

There are an infinity of potential coding methods for encoding and decoding, but, in practice, we should choose one that is easy to implement. In this paper, we consider only integer constants and variables. For integers, there are 4 common operations: +, -, $\times$, /.

3 Homomorphic obfuscations for product of sets of integers

3.1 Basic definitions

Definition 1 *If a function $f : Z/nZ \rightarrow Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ satisfies the condition that*

for any two $[x], [y] \in Z/nZ$, we have
$$f([x] + [y]) = f([x]) + f([y]).$$

then we call it a **homomorphic obfuscation** *from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$.*

If a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ is also a bijection, we call it an **isomorphic obfuscation** *from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$.*

For any Z/nZ and $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, there is always a trivial homomorphic obfuscation as follows.

$$f([x]_n) = ([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k}) \text{ for any } [x] \in Z/nZ.$$

For a homomorphic obfuscation $f : Z/nZ \rightarrow Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ and $[x], [y] \in Z/nZ$, it is easy to prove the following properties.

1. $f([0]_n) = ([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k})$
2. $f([x] - [y]) = f([x]) - f([y])$
3. $f([x][y]) = f([x])f([y])$, so $f([x]_n) = ([x]_{m_1}, [x]_{m_2}, \ldots, [x]_{m_k})f([1]_n)$

3.2 Examples of homomorphic obfuscations

1. For any Z/nZ and Z/mZ, there is always a trivial homomorphic obfuscation $f : Z/nZ \rightarrow Z/mZ$:

   $$f([x]_n) = [0]_m \in Z/mZ, \text{ for any } [x]_n \in Z/nZ$$

2. For any Z/nZ, there is an identity homomorphic obfuscation $f : Z/nZ \rightarrow Z/nZ$ as follows:

   $$f([x]_n) = [x]_n, \text{ for any } [x]_n \in Z/nZ$$

   In fact, it is an identity isomorphic obfuscation from Z/nZ to itself.

3. For any Z/nZ and $Z/2nZ$, there is a homomorphic obfuscation $f : Z/nZ \rightarrow Z/2nZ$ as follows:

$$f([x]_n) = [2x]_{2n} \in Z/2nZ, \text{ for any } [x]_n \in Z/nZ$$

4. For $Z/10Z$ and $Z/5Z$, there is a homomorphic obfuscation $f : Z/10Z \to Z/5Z$ as follows:

$$f([x]_{10}) = [2x]_5 \in Z/5Z, \text{ for any } [x]_{10} \in Z/10Z$$

that is

$$f([0]_{10}) = [0]_5,\ f([1]_{10}) = [2]_5,\ f([2]_{10}) = [4]_5,\ f([3]_{10}) = [1]_5,$$
$$f([4]_{10}) = [3]_5,\ f([5]_{10}) = [0]_5,\ f([6]_{10}) = [2]_5,\ f([7]_{10}) = [4]_5,$$
$$f([9]_{10}) = [1]_5,\ f([9]) = [3]_5$$

5. For any Z/pZ and Z/qZ in which p, q are two distinct prime numbers, there is only the trivial homomorphic obfuscation from Z/pZ to Z/qZ.

3.3 Representation of homomorphic obfuscations

Theorem 1 *(The first representation theorem for homomorphic obfuscations) For any Z/nZ and $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, if $l_1, l_2, \ldots, l_k$ are integers such that $m_i|nl_i$, for $i = 1, 2, \ldots, k$, then the function*

$$f([x]_n) = ([l_1x]_{m_1}, [l_2x]_{m_2}, \ldots, [l_kx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

is a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$.
On the other hand, if f is a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, then
$f([x]_n) = ([l_1x]_{m_1}, [l_2x]_{m_2}, \ldots, [l_kx]_{m_k})$, for any $[x]_n \in Z/nZ$ and
$m_i|nl_i$, for $i = 1, 2, \ldots, k$ where $([l_1]_{m_1}, [l_2]_{m_2}, \ldots, [l_k]_{m_k}) = f([1])$.
Furthermore, we can choose these integers such that $0 \leq l_i < m_i$ for $i = 1, 2, \ldots, k$ and we say the homomorphic obfuscation f has the representation $(l_1, l_2, \ldots, l_k)$.

Theorem 2 *(The second representation theorem for homomorphic obfuscations) Assume that Z/nZ and that $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ satisfy that $m_1, m_2, \ldots, m_k \in Z$ are pairwise relatively prime and $n = m_1 \times m_2 \times \ldots \times m_k$, we have the following results:*

For any integer l, then the function

$$f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

is a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$.

On the other hand, if f is a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, then there exists an integer l such that

$$f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

Proof. We only need to prove the second part of our result. By Theorem 1, for a homomorphic obfuscation f from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, there integers $l_1, l_2, \ldots, l_k$ such that

$$f([x]_n) = ([l_1x]_{m_1}, [l_2x]_{m_2}, \ldots, [l_kx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

By Theorem A.28 in [12, page 255], there is an integer l such that $[l]_{m_i} = [l_i]_{m_i}$, for $i = 1, 2, \ldots, k$. Therefore, for any x, $[lx]_{m_i} = [l_ix]_{m_i}$, for $i = 1, 2, \ldots, k$. We get $f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k})$, for any $[x]_n \in Z/nZ$

Theorem 3 *(The representation theorem for isomorphic obfuscations) Assuming that Z/nZ and $that Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ satisfy $n = m_1 \times m_2 \times \ldots \times m_k$ and $m_1, m_2, \ldots, m_k \in Z$ are pairwise relatively prime integers, we have the following results:*

For any integer l such that l and n are relatively prime, the function

$$f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

is an isomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$.

On the other hand, if f is an isomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$, then there exists an integer l such that

$$f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k}), \text{ for any } [x]_n \in Z/nZ$$

Furthermore, $([l]_{m_1}, [l]_{m_2}, \ldots, [l]_{m_k}) = f([1])$ and l and n are relatively prime.

Proof. By Theorem 2, for any integer l satisfying that l and n are relatively prime, the function $f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k})$, for any $[x]_n \in Z/nZ$ is a homomorphic obfuscation from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$. We prove this function is also a bijection. Firstly, we prove that

if $f([x]_n)=([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k})$, then $[x]_n = [0]_n$.

In fact, if $f([x]_n) = ([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k})$, then $([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k}) = ([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k})$, that means $m_i|lx$ for $i = 1, 2, \ldots, k$. Because m_1, m_2, …, $m_k \in Z$ are pairwise relatively prime integers, we have $m_1m_2 \ldots m_k|lx$. By $n = m_1 \times m_2 \times \ldots \times m_k$ and l and n are relatively prime, we have $n|x$, that means $[x]_n = [0]_n$. Secondly, by the property 2 of homomorphic obfuscations, $f([x] - [y]) = f([x]) - f([y]))$, we have if $[x]_n = [y]_n$, then $f([x]_n) = f([y]_n)$. Because Z/nZ and $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ are finite sets with the same cardinality, the above function f must be a bijection, so it is an isomorphic obfuscation.

On the other hand, if a function f from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ is an isomorphic obfuscation, by Theorem 2, we have an integer l such that

$f([x]_n) = ([lx]_{m_1}, [lx]_{m_2}, \ldots, [lx]_{m_k})$, for any $[x]_n \in Z/nZ$

We prove l and n are relatively prime. Otherwise, l and n has a common divisor $d > 1$. Let $n = n'd$ and $l = l'd$, then $[n']_n \neq [0]_n$, but $[ln']_{m_i} = [l'd \times n']_{m_i} = [l' \times n'd]_{m_i} = [l' \times n]_{m_i} = [0]_{m_i}$, for $i = 1, 2, \ldots, k$, that is $f([n']_n) = ([0]_{m_1}, [0]_{m_2}, \ldots, [0]_{m_k})$. This concludes our result.

4 A solution to division of integers by constants

4.1 Division of integers by a constant

The division of residue numbers is a complicated problem. In line 38 and 39 of a patent [2, page 17], Chow et al. wrote "Most texts like Knuth also indicate that division is impossible." However, to our knowledge, Knuth has never stated that residue division is impossible. Knuth, in [1], discusses the division of residue numbers: "It is even more difficult to perform division" in line 21 of pp. 285; see also Exercise 4.3.2-11 at pp. 293. Chow et al's patent [2, page 17] describes a method for division of residue numbers, at line 39 and 40, "However, the invention provides a manner of division by a constant." We assert, and will prove, that this method is invalid. The main problem is that the formula (19) and (20) in the patent [2, page 17] are incorrect. Because these two equations are the basis for the techniques of the division of residue numbers, the algorithm as described in [2, page 17] is incorrect. The solution to division by a constant d in the patent [2] has another problem: an overly restrictive condition that such d can only be one of its bases.

In Section 2 through 3 on this paper, we have laid a sound grounding for constructing a mechanism for division by constants. Now we give a method for division by a constant d. Assume that Z/nZ and $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ satisfy $n = m_1 \times m_2 \times \ldots \times m_k$ and $m_1, m_2, \ldots, m_k$ are pairwise relatively prime integers and l and n are relatively prime. We define an isomorphic obfuscation f from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ as follows:

$$f([y]_n) = ([dy]_{m_1}, [dy]_{m_2}, \ldots, [dy]_{m_k}), \text{ for any } [y]_n \in Z/nZ$$

then, for any $0 \leq x < n$ such that $d|x$, we have

$$f([\frac{x}{d}]) = ([x]_{m_1}, [x]_{m_2}, \ldots, [x]_{m_k}).$$

This solution is simple and easy to implement, and it is better than that in Patent [2]. As for the following restrictions to our solution, here are some comments.

1. Our first constraint is that $d|x$. This is unsurprising, for if this constraint is not met, then
$$\frac{x}{d}$$
is not a linear function of x.
2. Our second constraint is that d and n are relatively prime. This is not a big problem in some application, for when obfuscating a program containing a single constant d, we have the freedom to choose n so that this condition is satisfied.

In the case that d is one of $m_1, m_2, \ldots, m_k$, as in Chow et al's patent [2], we see some fundamental difficulties, for we can prove the following result.

Theorem 4

$$[\frac{y}{m_i}]_{m_i}$$

is not a linear function of $[y]_{m_1}, [y]_{m_2}, \ldots, [y]_{m_k}$*, that is, there are no constants with respect to y,* $c_1, c_2, \ldots, c_k$*, such that*

$$[\frac{y}{m_i}]_{m_i} = c_1[y]_{m_1} + c_2[y]_{m_2} + \ldots + c_k[y]_{m_k} \quad (1)$$

where $m_i|y$.

Let's first consider the case $k = 2$ and $i = 1$ and $m_1 < m_2$. If the above linear function exists, that is

$$[\frac{y}{m_1}]_{m_1} = c_1[y]_{m_1} + c_2[y]_{m_2} \quad (2)$$

holds for all $m_1|y$. Let $y = m_1^2$, we have

$$[\frac{m_1^2}{m_1}]_{m_1} = c_1[m_1^2]_{m_1} + c_2[m_1^2]_{m_2} \quad (3)$$

So, $c_2[m_1^2]_{m_2} = 0$. For m_1 and m_2 are relatively prime, $c_2 = 0$.

Now Equation (2) is reduced to

$$[\frac{y}{m_1}]_{m_1} = c_1[y]_{m_1} \quad (4)$$

Let $y = m_1m_2$, we have $[m_2]_{m_1} = 0$. This is a contradiction.

For other cases, the proof is similar.

Compared with the method in Chow et al's patent [2], where d can only be one of $m_1, m_2, \ldots, m_k$, our solution has a more freedom for choosing d. When d is one of $m_1, m_2, \ldots, m_k$, our solution does not work, but we can choose another set of parameters $m_1, m_2, \ldots, m_k$ so that our solution works. In fact, the solution in Chow et al's patent [2] itself is incorrect in the case that d is one of $m_1, m_2, \ldots, m_k$.

4.2 Division of integers by several constants

Assume that Z/nZ and $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ satisfy $n = m_1 \times m_2 \times \ldots \times m_k$ and $m_1, m_2, \ldots, m_k$ are pairwise relatively prime integers, $d_1, d_2, \ldots, d_p$ are p integers, and d_i and n are relatively prime for any $i = 1, 2, \ldots, p$. Let $d = d_1d_2 \ldots d_p$, we define an isomorphic obfuscation f from Z/nZ to $Z/m_1Z \times Z/m_2Z \times \ldots \times Z/m_kZ$ as follows:

$$f([y]_n) = ([dy]_{m_1}, [dy]_{m_2}, \ldots, [dy]_{m_k}), \text{ for any } [y]_n \in Z/nZ$$

then, for any $1 \leq i \leq p$ and $0 \leq x < n$ such that $d_i|x$, denote $d/d_i = d_1 \cdots d_{i-1}d_{i+1} \cdots d_k$ as d'_i, we have

$$f([\frac{x}{d_i}]) = ([d'_i x]_{m_1}, [d'_i x]_{m_2}, \ldots, [d'_i x]_{m_k})$$

5 Conclusions

While the residue number coding can be used in RSA cryptography, it also has applications to software obfuscation to encode variables to hide the real meaning of these variables [2]. Unfortunately, the method for the division in residue number coding proposed in patent [2] is incorrect. The method for division of residue numbers proposed in this paper is based on a sound grounding in number theory and can be used in software obfuscation to hide integers. It is also novel in the software obfuscation literature.

There are some points in our methods requiring further exploration, such as the security of our methods, the combination of our methods with other software obfuscation techniques, extensions of the homomorphic obfuscation concepts to fields that are not integral numbers, etc. We will investigate these issues in our future research.

Acknowledgements: Thanks for Dr. F.-Y. Wang's stimulating comments on this paper.

References

[1] D. Knuth, The art of computer programming, vol. 2, seminumerical algorithms (1997) 14–524.

[2] Chow, et al, Tamper resistant software encoding, US patent 6594761 (2003) 1–32.

[3] D. Gollmann, Computer security, New York: Willey, 1999.

[4] W. Zhu, C. Thomborson, F.-Y. Wang, A survey of software watermarking, in: LNCS 3495, 2005, pp. 454–458.

[5] C. Collberg, C. Thomborson, Watermarking, tamper-proofing, and obfuscation - tools for software protection, in: IEEE Transactions on Software Engineering, Vol. 28, 2002, pp. 735–746.

[6] C. Collberg, C. Thomborson, D. Low, A taxonomy of obfuscating transformations, in: Tech. Report, No.148, Dept. of Computer Sciences, Univ. of Auckland, 1997.

[7] L. Ertaul, S. Venkatesh, Jhide - a tool kit for code obfuscation, in: 8th IASTED International Conference on Software Engineering and Applications (SEA 2004), 2004, pp. 133–138.

[8] L. Ertaul, S. Venkatesh, Novel obfuscation algorithms for software security, in: 2005 International Conference on Software Engineering Research and Practice, SERP'05, 2005, pp. 209–215.

[9] Y. Sakabe, M. Soshi, A. Miyaji, Java obfuscation approaches to construct tamper-resistant object-oriented programs, IPSJ Digital Courier 1 (2005) 349–361.

[10] J. Nicherson, S. Chow, H. Johnson, Tamper resistant software: extending trust into a hostile environment, in: Proceedings of ACM Multimedia '01, ACM Press, 2001.

[11] Chow, et al., An approach to the obfuscation of control-flow of sequential computer programs, in: LNCS 2200, 2001, pp. 144–155.

[12] H. Delfs, H. Knebl, Introduction to cryptography, principles and applications, Springer-Verlag, 2002.