An approach to quantitative analysis of resistance of equivalent transformations of algebraic circuits

A. V. Shokurov

Abstract. A system of computations on encrypted data such that

- transformation of encryption is effective, i.e. can be performed in polynomial time on the size of circuit $\mathcal{C}$;
- the size of scheme A' differs not essentially from the size of initial scheme A;
- lower bounds on resistance of circuit is exponential

is constructed.

1. Introduction

Tamper-resistant software technologies are intended for protecting software programs from intelligent tampering attacks aimed at obtaining some extra-knowledge about program structure and behavior (the key ideas of algorithms used in a program and some specific data — passwords, constants, parameters, etc.). To achieve these goals some specific semantic-preserving transformations are applied to a source computer program that perform deep and sophisticated changes in its control flow and data flow [9, 8, 5].

Tamper resistant software is one of the forms of software protection against reverse engineering. Data protection is an important part of tamper resistant software. This protection can be done in different ways and one of them which uses *data encodings* [7, 5] is considered in this paper.

Formally model with transformed data can be described as follows. Let Alice and Bob are two participants of a computation. Alice is the owner of some data but has not enough computational resources. Bob possesses enough amount of computational resources but Alice does fully not entrust Bob. Alice needs to transform data, constants and computational circuit in such a way that the needed by Alice result could be easily obtained from the result of transformed computation over transformed data. Then Alice needs to send all transformed data, constants and computational circuit to Bob. These transformations need to be such that the second participant could get minimum information (in ideal case nothing) from received data and circuit.

Now we can state the following problem.

Problem. Let $\mathcal{C}'$ be some transformed computational circuit. Find initial computational circuit $\mathcal{C}$ corresponding to $\mathcal{C}'$. How much computational circuits $\mathcal{C}$ correspond to transformed circuit $\mathcal{C}'$? It is suggested that the parameters of given transform are not given.

To explain what are data encodings consider the problem of computation of some arithmetical expression

$$y = F(x, c),$$

where $x = (x_1, \ldots, x_n)$ are input data and $c = (c_1, \ldots, c_m)$ are some internal parameters which we need to protect against an adversary. Let $y = (y_1, \ldots, y_t)$ be the result of this computation.

Encoding in general is a parametric collection of functions that map each integer into some tuples of integers. Thus, any integer x will be converted to $x' = (x'_1, x'_2, \ldots, x'_k)$. A simple example of it is so called linear encoding $x' = a \cdot x + b$. Integers a, b are parameters of this encoding [7]. Formal definition of data encoding will be given in Section 2.

The key idea of using such encodings is the following. Instead of computing $y = F(x, c)$ we perform some distorted computation $\tilde{y} = \tilde{F}(x', c')$, which is obtained from original computation F by some rules and then apply **decoding** (the inverse function to encoding) to obtain "real" results of the computation.

One of possible approaches to obtain $\tilde{F}$ is constructing for every basic operation ($+$,$\times$, etc.) used in F a corresponding sequence of operations over encoded data. This is possible if any arithmetic operation on data can be expressed in terms of encoded data. Such encodings will be called *homomorphic* (for formal definition see Section 2). To clarify this notion consider the following example. Let integers x_1 and x_2 be represented using linear encoding as

$$\begin{aligned} x'_1 &= a_1 \cdot x_1 + b_1 \\ x'_2 &= a_2 \cdot x_2 + b_2 \end{aligned} \tag{1}$$

and

$$\begin{aligned} A_1 &= a_1/m \\ A_2 &= a_2/m \end{aligned} \tag{2}$$

where

$$m = \mathrm{GCD}(a_1, a_2). \tag{3}$$

To calculate $y = x_1 + x_2$ in terms of linear encoding one can find its encoded value by the formula

$$\tilde{y} = A_2 \cdot x'_1 + A_1 \cdot x'_2. \tag{4}$$

Then

$$\tilde{y} = B_2 y + B_1 \tag{5}$$

where $B_1 = (b_1 a_2 + b_2 a_1)/m$ and $B_2 = a_1 a_2/m$ and the value y can be decoded from $\tilde{y}$ as follows

$$y = (\tilde{y} - B_1)/B_2. \tag{6}$$

Hence the result of addition is represented in terms of encoded data by the formula (4). In this example the arithmetic expression is $y = F(x_1, x_2) = x_1 + x_2$ and the distorted expression $\tilde{y} = \tilde{F}(x'_1, x'_2)$ is given by the formula (4).

Note that an adversary observes the values of parameters A_1 and A_2 which are related with encoding parameters by formulas (2–3), i.e. an adversary knowing the values A_1 and A_2 may try to guess the values of encoding parameters. The main question is the following. Whether such information is enough to find the values of all encoding parameters? Moreover it is clear that the more operations on encoded data an adversary should observe the more information he gets on encoding parameters. The aim of "good" encoding is to minimize such information.

Local analysis of encoded procedures may give to adversary some information about parameters of encoding. An example of such information is given above where addition of two linearly encoded integers is discussed. Therefore the problems of proper choice of encoding and of comparison of encodings are important. The solution of such problems should be based on a notion of measure of resistance of data encodings against local analysis (when each encoded procedure is treated separately). These problems are considered in this paper.

We propose a notion of quantitative measure of *resistance* of encoded circuit which we consider as our main contribution. This measure gives the opportunity to compare different encoded circuits. Estimates for lower bounds of resistance are obtained for relatively wide class of algebraic circuits. These estimates show high security of computations performed in such encodings. Typically such bounds are at least of order 2^{100}.

The structure of the paper is the following. In Section 2 formal definitions of encoding and homomorphic encoding are given. In Section 3 some examples of homomorphic encodings are considered. A measure of resistance of computation is introduced in Section 4. In Section 5 we present lower bounds of resistance of different types of encodings for some algebraic circuits. The proofs are given in the Appendix.

2. Notion of data encoding

Now we give a formal definition of encoding.

Definition 1. *Encoding* is a pair (φ, ψ) of transformations of integer data $(x_1, \ldots, x_n)$ and constants (parameters of encoding) $(c_1, \ldots, c_m)$ into integer data $(x'_1, \ldots, x'_k)$ such that

$$\begin{array}{l} \varphi(x_1, \ldots, x_n, c_1, \ldots, c_m) = (x'_1, \ldots, x'_k) \\ \psi(x'_1, \ldots, x'_k, c_1, \ldots, c_m) = (x_1, \ldots, x_n), \end{array} \tag{7}$$

where decoding ψ is left inverse of φ, i.e.

$$\psi(\varphi(x_1, \ldots, x_n, c_1, \ldots, c_m), c_1, \ldots, c_m) = (x_1, \ldots, x_n).$$

The data $(x'_1, \ldots, x'_k)$ are called encrypted data (cryptogram), $(x_1, \ldots, x_n$ is a plaintext and $c_1, \ldots, c_n)$ is a key.

Now the problem is to find a proper circuit $\tilde{F}$ on encoded data for arbitrary algebraic circuit F without multiplicity [1]. One of possible approaches is to construct for every basic arithmetic operation ($+$,$\times$, etc.) used in F a corresponding sequence of operations on encoded data. This is possible if any arithmetic operation over original data can be expressed in terms of encoded data. Such encodings will be called *homomorphic*.

Now we give a formal definition for such encodings.

Consider encoding (φ, ψ) of two integers x_1 and x_2 assuming $n = 1$ in formula (7) and

$$\varphi(x_1, c_{11}, \ldots, c_{m1}) = (x'_{11}, \ldots, x'_{k1}).$$

Let also

$$\varphi(x_2, c_{12}, \ldots, c_{m2}) = (x'_{12}, \ldots, x'_{k2}).$$

Integer numbers $c_{11}, \ldots, c_{m1}$ and $c_{12}, \ldots, c_{m2}$ are parameters of encoding of x_1 and x_2 respectively. Let $x_3 = h(x_1, x_2)$ be some binary operation on integer inputs with integer outcome (say, addition, multiplication, etc.).

Definition 2. Encoding (φ, ψ) is called *homomorphic* with respect to h if there exist functions A, C

$$\begin{array}{rcl} (\alpha_1, \ldots, \alpha_s) & = & A(c_{11}, \ldots, c_{m1}, c_{12}, \ldots, c_{m2}) \\ (c_{13}, \ldots, c_{m3}) & = & C(c_{11}, \ldots, c_{m1}, c_{12}, \ldots, c_{m2}) \end{array} \tag{8}$$

and an arithmetic procedure (a sequence of arithmetic operations)

$$(y_1, \ldots, y_k) = \tilde{h}(x'_{11}, \ldots, x'_{k1}, x'_{12}, \ldots, x'_{k2}, \alpha_1, \ldots, \alpha_s) \tag{9}$$

with integer input data $x'_{11}, \ldots, x'_{k1}, x'_{12}, \ldots, x'_{k2}, \alpha_1, \ldots, \alpha_s$ such that

$$h(x_1, x_2) = x_3 = \psi(y_1, \ldots, y_k, c_{13}, \ldots, c_{m3}).$$

It follows from the definition that function C computes the parameters of encoding of the outcome of operation whereas function A creates the parameters of arithmetic procedure $\tilde{h}$. It is clear from definition that one can choose as function A the identity function

$$A(c_{11}, \ldots, c_{m1}, c_{12}, \ldots, c_{m2}) = (c_{11}, \ldots, c_{m1}, c_{12}, \ldots, c_{m2}),$$

but in this case the aim of encoding should not be achieved, because all the parameters of encoding should take part in computation. The aim of function A is to decrease the information on encoding parameters and to show the possible minimum. In ideal case function A minimizes information on these parameters. It is also clear from the definition that $\tilde{h}$ represents algebraic circuit. Note that functions C and A may be arbitrary but for practical use need to be efficiently computable. To find encoded value of the result of operation it is not necessary to know all the parameters of encodings but their transform $\alpha_1, \ldots, \alpha_s$.

Linear encoding (1) gives an example of homomorphic encoding with respect to operations of addition and multiplication of integer numbers. For addition it follows from formulas (2–5) because

$$x'_1 = \varphi(x_1, a_1, b_1) = a_1 x_1 + b_1,$$

$$x'_2 = \varphi(x_2, a_2, b_2) = a_2 x_2 + b_2,$$

$$(\alpha_1, \alpha_2) = A(a_1, b_1, a_2, b_2) = (A_1, A_2),$$

$$(a_3, b_3) = C(a_1, b_1, a_2, b_2) = (B_2, B_1),$$

where A_1, A_2, B_1 and B_2 are given by formulas (2–3) and arithmetic procedure $\tilde{h}$ is given by formula (4).

3. Examples of homomorphic encodings

In this section we present some examples of homomorphic encodings with respect to addition and multiplication.

- Linear encoding (see, Section 2).

- Residue encoding of integer x is a tuple

$$x'_i = x \pmod{p_i},$$

where p_i, $i = 1, \ldots, m$ are coprime integers. The parameters of encoding for all data are the same and are equal to

$$(c_1, \ldots, c_m) = (p_1, \ldots, p_m).$$

The formulas for decoding function ψ are given in [2].

Addition and multiplication of integers in this encoding are performed as follows. Let $y'_i = y \pmod{p_i}$, $i = 1, \ldots, k$ then $x'_i + y'_i \pmod{p_i}$ is the encoded sum $x + y$ and $x'_i \cdot y'_i \pmod{p_i}$ is the encoded product $x \cdot y$ [1, 2]. The function A from formula (8) is a constant function

$$A(p_1, \ldots, p_m, p_1, \ldots, p_m) = 0$$

because the formulas for addition and multiplication do not depend on parameters of encoding $p_1, \ldots, p_m$.

- Mixed encoding is a generalization of linear and residue encodings. Fix coprime integers $p_1, \ldots, p_m$. Integer x is represented in mixed encoding as

$$\begin{aligned} x'_1 &\equiv a_1 \cdot x_1 + b_1 \mod p_1 \\ &\ldots \\ x'_m &\equiv a_m \cdot x_m + b_m \mod p_m \end{aligned} \tag{10}$$

where $\mathrm{GCD}(a_k, p_k) = 1$ for all $k = 1, \ldots, m$. The parameters of encodings in this case are $(a_1, b_1, \ldots, a_m, b_m)$. Now take two integers encoded by parameters $(a_{11}, b_{11}, \ldots, a_{m1}, b_{m1})$ and $(a_{12}, b_{12}, \ldots, a_{m2}, b_{m2})$ correspondingly. Then addition $y = x_1 + x_2$ in terms of encoded data can be made by formula

$$y'_i = \gamma_i a_{i1}^{-1} x'_{i1} + \gamma_i a_{i2}^{-1} x'_{i2} \mod p_i$$

for arbitrary γ_i such that $\mathrm{GCD}(\gamma_i, p_i) = 1$. So the function A from formula (8) is

$$\begin{aligned} &A(a_{11}, b_{11}, \ldots, a_{m1}, b_{m1}, a_{12}, b_{12}, \ldots, a_{m2}, b_{m2}) = \\ &(\alpha_{11}, \ldots, \alpha_{m1}, \alpha_{12}, \ldots, \alpha_{m2}), \end{aligned}$$

where $\alpha_{ij} \equiv \gamma_i a_{ij}^{-1} \mod p_i$. Function C in this case is

$$\begin{aligned} &C(a_{11}, b_{11}, \ldots, a_{m1}, b_{m1}, a_{12}, b_{12}, \ldots, a_{m2}, b_{m2}) = \\ &(\gamma_1, -(\alpha_{11} b_1 + \alpha_{12} b_2), \ldots, \gamma_m, -(\alpha_{m1} b_1 + \alpha_{m2} b_2)). \end{aligned}$$

The formula for performing multiplication in mixed encoding is given in section 5 and multiplication also is homomorphic for mixed encoding.

- p-base representation gives an homomorphic encoding [11].
- Encoding based of Discrete Fourier Transform [11, 2].

One can try to protect data in a program using well-known cryptographic encodings. Let us consider the following example: RSA function $x' = x^e \bmod m$, where $m = pq$, p and q are prime numbers [4].

Given encoded data x' and y' it is easy to implement multiplication $z = xy$ as follows: $z' = x'y'$. So this encoding is homomorphic with respect to multiplication.

But it is difficult to implement in the same manner the sum of RSA encoded data. This is the answer to the question: "why we do not use well-known cryptographic functions for encodings of data"?

4. Resistance of data encodings

Firstly introduce the notion of the "observable" and the "real" worlds. The "observable" world is a set of encrypted values which Alice sends to Bob. The "real" world is a set of non-encrypted values of inputs and constants. Illustrate this by the following example. Let x be encoded in linear encoding as $x' = a{\cdot}x{+}b$ and assume that an adversary Bob observes only x'. The "observable" world is x' and a "real" world is a set (a, x, b) for which encoded x corresponds to the observable value x'.

It is obvious that the same "observable" world can correspond to several "real" worlds. And any of these "real" worlds can be the real world which we encode. In the example mentioned above the number of the "real" worlds, which we denote as R_w, can be estimated as $R_w \geq K^2$, where K is the **range** of integers we use.

Note that operations with encoded data can reduce the resistance because additional relations between parameters occur. It can be illustrated by the example of the sum of two integers in linear encoding given in the previous section.

Let integers x and y be represented as x' and y' in linear encodings by formulas (1–2). The sum $z = x + y$ is given in terms of encoded data by formula (4).

The observable world is determined by the following parameters: x', y', A_1, A_2 and the number of "real" worlds is the number of solutions of the corresponding system of equations (1-4). Additional relations which reduce R_w are

equations (2-3). The solution (i.e., one of the possible "real" worlds R_w) is a set of values for x, y, a_1, a_2, b_1, b_2. Let us denote the range of possible values as K.

We estimate now the number of "real" worlds in some cases.

Proposition 1. For fixed x', y', A_1, A_2 and a_1, a_2 the number of possible solutions $R_w \geq K^2$.

To prove it note that arbitrary values of x or y are solutions of our system as for any x (y, respectively) one can choose such b_1 (b_2, respectively) that the value of x' (y', respectively) does not change.

Proposition 2. For fixed x', y', A_1, A_2 and x, y the number of possible solutions can be estimated as $R_w \geq K/\max(a_1, a_2)$.

Note that for some solutions of our system a_1, a_2 and for any q the values $\tilde{a}_1 = q \cdot a_1$ and $\tilde{a}_2 = q \cdot a_2$ also give a solution because A_1, A_2 are the same and there exist $\tilde{b}_1$ and $\tilde{b}_2$ such that x' and y' do not change.

Proposition 3. The number of real worlds is $R_w \geq K^3/A$, where $A = \max(a_1, a_2)$.

This follows immediately from **Proposition 1** and **Proposition 2**.

As we can see in this example the procedure of estimating R_w (the number of "real" worlds) can be rather difficult and greatly depends on the sequence of the operations with encoded data.

The number of such "real" worlds which correspond to the same observable world can be used for estimating the resistance of the encoding. We introduce a measure of encodings resistance as a measure of uncertainty that is the number of "real" worlds R_w which can correspond to the observable encoded world. An adversary observing only operations in encoded world and inputs to encoded world (i.e., all encoded input data) can not distinguish between any of "real" worlds. Thus the more there is the number of corresponding "real" worlds the more there are uncertainty and resistance of encoding.

Let $y = F(x, c)$ be some algebraic circuit [1] and $\tilde{y} = \tilde{F}(x', c')$ be its corresponding encoded circuit.

Definition 3. A measure of resistance of *observed encoded circuit* $\tilde{y} = \tilde{F}(x', c')$ is the number of different "real" worlds (c, x) which correspond to the same encoded world $(x', c', \tilde{F})$.

We will denote this resistance by $R_w(x', c')$. Now take the maximum of such circuits over all encoded data x'.

Definition 4. A measure of resistance of *encoded circuit* $\tilde{y} = \tilde{F}(x', c')$ is the maximum of all observed resistances. We denote the resistance by $R_w(F)$ or simply by R_w.

It is important to note that such measure characterize the resistance of encoding to arbitrary attack which uses only information from encoded world. It means that this measure characterizes absolute resistance.

Protection of data in encoded world will be guaranteed by lower bounds of resistance of encodings which one can obtain. Typically the bounds we present below are at least 2^{100} when the range of integers is 2^{64}.

5. Results

5.1. Mixed encoding

In this section some estimates of resistance of some circuits of computations are obtained. Two algebraic circuits with encoded data are considered — the first is a computation of second degree form and the second is some homogeneous circuit without multiplicity. It is shown that for both cases we have the following estimate of resistance

$$R_w \geq (\nu(p_1) \cdot \ldots \cdot \nu(p_m))^n,$$

where ν is the Euler function, $p_1, \ldots, p_m$ are mixed encoding parameters, and n is the number of input variables.

Mixed encoding for a vector of data $x = (x_1, \ldots, x_n)^t$ and coprime numbers $(p_1, \cdots, p_m)$ is given by equations

$$\begin{array}{lcll} x'_{11} & = & a_{11}x_1 + b_{11} & \mod p_1 \\ \cdots & \cdots & \cdots & \cdots \\ x'_{1m} & = & a_{1m}x_1 + b_{1m} & \mod p_m \\ \cdots & \cdots & \cdots & \cdots \\ x'_{n1} & = & a_{n1}x_1 + b_{n1} & \mod p_1 \\ \cdots & \cdots & \cdots & \cdots \\ x'_{nm} & = & a_{nm}x_1 + b_{nm} & \mod p_m. \end{array} \tag{11}$$

Then the encoded data are given by the matrix

$$\begin{pmatrix} x'_{11} & \cdots & x'_{1m} \\ \vdots & \ldots & \vdots \\ x'_{n1} & \cdots & x'_{nm} \end{pmatrix}. \tag{12}$$

The original data can be decoded from encoded data by the following procedure. Let

$$\begin{array}{lcll} x_{11} & = & c_{11}x'_{11} + d_{11} & \mod p_1 \\ \cdots & \cdots & \cdots & \cdots \\ x_{1m} & = & c_{1m}x'_{1m} + d_{1m} & \mod p_m \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & = & c_{n1}x'_{n1} + d_{n1} & \mod p_1 \\ \cdots & \cdots & \cdots & \cdots \\ x_{nm} & = & c_{nm}x'_{nm} + d_{nm} & \mod p_m. \end{array} \tag{13}$$

Then

$$\begin{array}{lcll} x_1 & = & \lambda_1 x_{11} + \cdots + \lambda_m x_{1m} & \mod p_1 \cdot \ldots \cdot p_m \\ \cdots & \cdots & \cdots & \cdots \\ x_n & = & \lambda_1 x_{n1} + \cdots + \lambda_m x_{nm} & \mod p_1 \cdot \ldots \cdot p_m \end{array} \tag{14}$$

for some integer numbers $\lambda_1, \ldots, \lambda_m$.

How to perform operations over data in terms of encoded data?

To **add** two integer numbers $x_1 + x_2$ in terms of encoded data one may use the following procedure

$$\begin{array}{lll} x_{11} + x_{21} & = c_{11}x'_{11} + c_{21}x'_{21} + d_{11} + d_{21} & \mod p_1 \\ \cdots & \cdots & \cdots \quad \cdots \\ x_{1m} + x_{2m} & = c_{1m}x'_{1m} + c_{2m}x'_{2m} + d_{1m} + d_{2m} & \mod p_m. \end{array} \tag{15}$$

For calculation of linear **combination** of two integer numbers $\lambda_1 x_1 + \lambda_2 x_2$ the following formula can be used

$$\begin{array}{lll} \lambda_1 x_{11} + \lambda_2 x_{21} & = \lambda_1 c_{11}x'_{11} + \lambda_2 c_{21}x'_{21} + \lambda_1 d_{11} + \lambda_2 d_{21} & \mod p_1 \\ \cdots & \cdots & \cdots \quad \cdots \\ \lambda_1 x_{1m} + \lambda_2 x_{2m} & = \lambda_1 c_{1m}x'_{1m} + \lambda_2 c_{2m}x'_{2m} + \lambda_1 d_{1m} + \lambda_2 d_{2m} & \mod p_m. \end{array}$$

The products $\lambda_k d_{ki}$ and linear combinations $\lambda_1 b_{1i} + \lambda_2 b_{2i}$ can be calculated during the compilation.

Now consider **multiplication**. To calculate the product of two integers $x_1 x_2$ one may use the following formula

$$x_{1i} \cdot x_{2i} \quad = (c_{1i}x'_{1i} + d_{1i}) \cdot (c_{2i}x'_{2i} + d_{2i}) \mod p_i. \tag{16}$$

Hence

$$x_{1i} \cdot x_{2i} \quad = (c_{1i}c_{2i}x'_{1_i}x'_{2i} + d_{1i}c_{2i}x'_{2i} + d_{2i}c_{1i}x'_{2i}) + d_{1i}d_{2i} \mod p_i. \tag{17}$$

In observable world only the products $c_{1i}c_{2j}, d_{1i}c_{2i}, d_{2i}c_{1i}$ and $d_{1i}d_{2i}$ are given in evaluation.

Now we show how to estimate the resistance of encoding computations considering (as an important example) an evaluation of the second degree form

$$f(x_1,\dots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij}x_ix_j+\sum_{i=1}^{n}\beta_ix_i+\gamma \tag{18}$$

using encoded data x'. How to find it in terms of encoded data? To do this let us calculate its three summands using only encoded data.

Claim 1. Let

$$\xi_{ijk}=\alpha_{ijk}c_{ik}c_{jk} \quad \text{for} \quad i,j=1,\dots,n \tag{19}$$

$$\zeta_{ik}=\sum_{j=1}^{n}(\alpha_{ijk}+\alpha_{jik})c_{ik}d_{jk}+\beta_{ik}c_{ik} \quad \text{for} \quad k=1,\dots,m \tag{20}$$

$$\eta_k=\sum_{i,j=1}^{n}\alpha_{ijk}d_{ik}d_{jk}+\sum_{i=1}^{n}\beta_{ik}d_{ik}+\gamma_k \tag{21}$$

where coefficients c_{ik} are given by formulas (13). Then

$$f_k(x_1,\dots,x_n)=\sum_{i,j=1}^{n}\xi_{ijk}x'_{ik}x'_{jk}+\sum_{i=1}^{n}\zeta_{ik}x'_{ik}+\eta_k. \tag{22}$$

The proof is given in Appendix.

Claim 2. The resistance of formula (22) in mixed encoding is

$$R_w\geq\left(\nu(p_1)\cdot\ldots\cdot\nu(p_m)\right)^n,$$

where ν is the Euler function and n is a number of input variables.

Now consider algebraic circuits without multiplicity using operations of addition and multiplication of integers and define corresponding homogeneous algebraic circuits. We shall present such circuits as graphs of computation with operations of addition and multiplications as nodes and data and variables as leaves (see [3, 1]). Let input variables $x_1,\dots,x_n$ and the constant data $c_1,\dots,c_m$ of this circuit have some weights such that $\deg(x_i)=1$ and $\deg(c_i)<0$. The degree of the product is equal to the sum of degrees of multipliers. As usually the degree of the sum is not greater then the maximal degree of the summands.

Definition 5. A circuit is called homogeneous without multiplicity if:

(1) Addition is performed only for summands of the same nonpositive degree and the result has the same degree as its summands.

(2) Each coefficient c_i is used in the circuit only once.

(3) Multiple use of input variables is allowed.

Condition (1) means homogeneousity of the circuit. Condition (2) means that each coefficient is used only once.

Example 1. Consider Horner's scheme of computation of polynomial

$$P_l(x)=c_lx^l+\cdots+c_1x+c_0=(\dots(c_lx+c_{l-1})x+\cdots+c_1)x+c_0.$$

Let $\deg(c_k)=-k$ and $\deg(x)=1$. It is not difficult to see that corresponding circuit is homogeneous without multiplicity.

Example 2. Sparse multivariate polynomial gives another example of homogeneous circuit without multiplicity.

Now encode a homogeneous circuit step by step using the encodings of operations in mixed encoding.

Claim 3. The resistance of any encoded homogeneous circuit without multiplicity in mixed encoding is

$$R_w\geq\left(\nu(p_1)\cdot\ldots\cdot\nu(p_m)\right)^n,$$

where ν is the Euler totient function, n is the number of variables, and m is the parameter of encoding which is a number of modules p_k.

Now we add to the algebraic graph of computations on integers the nodes with one input and one output that correspond to exponents, i.e. if input is k, then output is k^m for some integer m. Definition of homogeneousity is the same as for ordinary homogeneous algebraic circuits. Such circuits will be called *general homogeneous circuits*. As for homogeneous circuits without multiplicity for general homogeneous circuits the following Claim holds.

Claim 4. The resistance of any encoded general homogeneous circuit without multiplicity in mixed encoding is

$$R_w\geq\left(\nu(p_1)\cdot\ldots\cdot\nu(p_m)\right)^n,$$

where ν is the Euler function, n is the number of variables, and m is the parameter of encoding which is a number of modules p_k.

Consider a circuit

$$\mathcal{C}(c_1,\dots,c_n,x_1,\dots,x_m),$$

which depends on parameters $c_1, \ldots, c_n$ and inputs $x_1, \ldots, x_m$. Some circuit

$$\mathcal{C}'(c_1, \ldots, c_n, c_{n+1}, \ldots, c_{n+k}, x_1, \ldots, x_m)$$

will be called a generalization of $\mathcal{C}$ if for some values of parameters $c_{n+1}, \ldots, c_{n+k}$ circuit $\mathcal{C}'$ computes the same result as circuit $\mathcal{C}$.

Theorem 1. For any circuit

$$\mathcal{C}(c_1, \ldots, c_n, x_1, \ldots, x_m)$$

there exists some homogeneous circuit without multiplicity

$$\mathcal{C}'(c_1, \ldots, c_n, c_{n+1}, \ldots, c_{n+k}, x_1, \ldots, x_m)$$

which generalizes the first circuit and contains extra multiplications that does not exceed the doubled number of additions and k is not greater then the number of extra multiplication.

Then from **Theorem 1** and **Claim 4** follows the next **Theorem**.

Theorem 2. For any algebraic circuit without multiplicity there exists encoded homogeneous circuit without multiplicity in mixed encoding which resistance is

$$R_w \geq \left(\nu(p_1) \cdot \ldots \cdot \nu(p_m)\right)^n,$$

where ν is the Euler function, n is the number of variables, and m is the parameter of encoding which is a number of modules p_k.

5.2. Multi-linear encoding

In this section for this type of encoding we propose some circuit of computation of second degree form with n variables and show that its resistance satisfies the inequality

$$R_w \geq \nu(K)^{mn+n}$$

where K is the range of integers used, m is the parameter of the encoding, and ν is the Euler function. This means that all computations are modulo K.

What is multi-linear encoding? Let $x = (x_1, \ldots, x_n)^t$ be a vector of data and

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ldots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \tag{23}$$

and

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \tag{24}$$

Then the encoded vector is

$$x' = \begin{pmatrix} x'_1 \\ \vdots \\ x'_m \end{pmatrix} = Ax + b.$$

For matrix A there must exist left inverse

$$B = \begin{pmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ldots & \vdots \\ b_{n1} & \cdots & b_{nm} \end{pmatrix}. \tag{25}$$

Then

$$x = BAx = B(x' - b) = Bx' - Bb = Bx' - b'. \tag{26}$$

The elements of matrix B may no longer be integers. Then there exists such integer m that matrix $mB = B_0$ is integer. Then vector $mb' = b_0$ is also integer and the following equation holds

$$mx = (mB)x' - mb' = B_0x' - b_0. \tag{27}$$

How to perform operations over data in terms of encoded data?

To **add** two integer numbers $x_1 + x_2$ in terms of encoded data one may use the following procedure

$$\begin{aligned} x_1 + x_2 &= (b_{11}x'_1 + \cdots + b_{1m}x'_m) + (b_{21}x'_1 + \cdots + b_{2m}x'_m) - b'_1 - b'_2 \\ &= (b_{11} + b_{21})x'_1 + \cdots + (b_{1m} + b_{2m})x'_m - (b'_1 + b'_2). \end{aligned} \tag{28}$$

For calculation of **linear combination** of two integer numbers $\lambda_1 x_1 + \lambda_2 x_2$ the following formula holds

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 &= \lambda_1(b_{11}x'_1 + \cdots + b_{1m}x'_m) + \lambda_2(b_{21}x'_1 + \cdots + b_{2m}x'_m) - \lambda_1 b'_1 - \lambda_2 b'_2 \\ &= (\lambda_1 b_{11} + \lambda_2 b_{21})x'_1 + \cdots + (\lambda_1 b_{1m} + \lambda_2 b_{2m})x'_m - (\lambda_1 b'_1 + \lambda_2 b'_2). \end{aligned}$$

The linear combinations of coefficients $\lambda_1 b_{1i} + \lambda_2 b_{2j}$ can be calculated during compilation.

Now consider **multiplication**. To calculate the product of two integers $x_1 x_2$ one may use the following formula

$$\begin{aligned} x_1 \cdot x_2 &= \left(\sum_{i=1}^{m} b_{1i} x'_i - b'_1\right) \cdot \left(\sum_{j=1}^{m} b_{2j} x'_j - b'_2\right) \\ &= \sum_{i,j=1}^{m} b_{1i} b_{2j} x'_i x'_j - \sum_{i=1}^{m} (b'_1 b_{2i} + b'_2 b_{1i}) x'_i + b'_1 b'_2. \end{aligned} \tag{29}$$

The products $b_{1i} b_{2j}$ and combinations $b'_1 b_{2i} + b'_2 b_{1i}$ can be calculated during compilation.

Now consider calculation of the second degree form

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij} x_i x_j + \sum_{i=1}^{n} \beta_i x_i + \gamma \tag{30}$$

using encoded data x'.

Claim 5. Let

$$\xi_{kl} = \sum_{i,j=1}^{n} \alpha_{ij} b_{ik} b_{jl} \quad \text{for} \quad k, l = 1, \ldots, m \tag{31}$$

$$\zeta_k = \sum_{i=1}^{n} \beta_i b_{ik} - \sum_{i,j=1}^{n} \alpha_{ij} (b'_i b_{jk} + b'_j b_{ik}) \quad \text{for} \quad k = 1, \ldots, m \tag{32}$$

$$\eta = \sum_{i,j=1}^{n} \alpha_{ij} b'_i b'_j - \sum_{i=1}^{n} \beta_i b'_i. \tag{33}$$

Then the following formula for computing function f defined by equation (30) holds

$$f(x_1, \ldots, x_n) = \sum_{k,l=1}^{m} \xi_{kl} x'_k x'_l + \sum_{k=1}^{m} \zeta_k x'_k + \eta. \tag{34}$$

Claim 6. The resistance of expression (34) in multi-linear encoding satisfies the inequality

$$R_w \geq (\nu(K))^{mn+n},$$

where K is the range of integers used and m is the parameter of the encoding.

For $K = 2^{64}$ (usual range for representing integers) the value $\nu(K) = 2^{63}$ and so lower bounds of suggested circuit of computation of second degree form is greater then 2^{100} which seems large enough from the point of computational complexity (enumerating all possible solutions is impossible) and from probabilistic point of view (the probability to guess right parameters is less than 2^{-100}).

6. On "good" and "bad" parameters of encodings

The measure of resistance of encodings we introduced gives the possibility to analyze encodings on a quantitative base and choose those parameters which provide greater resistance. Moreover, such measure of resistance allows to compare different algebraic circuits and choose those that have greater resistance.

6.1. An example of "bad" parameters in linear encodings

Let p_1, p_2, p_3, p_4 — be four different prime numbers. Let data $x_1 = p_3$ and $x_2 = p_4$ are linearly encoded by formulas

$$\begin{aligned} x'_1 &= p_1 x_1 + b_1 \\ x'_2 &= p_2 x_2 + b_2. \end{aligned}$$

Let adversary observes computation of their product $x_3 = x_1 x_2$ in terms of encoded data made by the following formula

$$x'_3 = x'_1 x'_2 - b_1 x'_2 - b_2 x'_1 + b_1 b_2.$$

The result of encoded computation is connected with real product by relation $x'_3 = a_1 a_2 x_3$. Therefore the adversary knows the values b_1, b_2, x'_1 and x'_2. So he can find that

$$\begin{aligned} x'_1 - b_1 &= a_1 x_1 \\ x'_2 - b_2 &= a_2 x_2. \end{aligned}$$

In our case the adversary finds that

$$\begin{aligned} a_1 x_1 &= p_1 p_3 \\ a_2 x_2 &= p_2 p_4. \end{aligned}$$

Hence he obtains that there are 4 cases for a_1 and x_1

- $a_1 = 1$, $x_1 = p_1 p_3$
- $a_1 = p_1$, $x_1 = p_3$

- $a_1 = p_3$, $x_1 = p_1$
- $a_1 = p_1 p_3$, $x_1 = 1$,

and 4 independent cases for a_2 and x_2

- $a_2 = 1$, $x_2 = p_2 p_4$
- $a_2 = p_2$, $x_2 = p_4$
- $a_2 = p_4$, $x_2 = p_2$
- $a_2 = p_2 p_4$, $x_2 = 1$.

So there are only 16 possible cases in this example.

6.2. Resistance and comparison of algebraic circuits for encoded computations

Now consider multiplication of two integers and estimate its resistance. At first consider **linear encoding**. Let encodings for x_1 and x_2 be given by formulas

$$x'_1 = a_1 x_1 + b_1$$
$$x'_2 = a_2 x_2 + b_2$$

and $m = \mathrm{GCD}(a_1, a_2)$. Then the product $y = x_1 x_2$ can be encoded by formula (see [10])

$$\tilde{y} = x'_1 x'_2 - b_2 x'_1 - b_1 x'_2.$$

Then an adversary observes the parameters of encoding b_1 and b_2 and finds that

$$a_1 a_2 \mid (x'_1 x'_2 - b_2 x'_1 - b_1 x'_2 + b_1 b_2).$$

In the example of Subsection 6.1 we obtain only 16 possibilities for parameters a_1, a_2, x_1 and x_2.

So in the case of linear encoding the upper bound for resistance of multiplication is equal to 16. To increase resistance of multiplication it is necessary to make restrictions on the encoding parameters.

Now consider a variant of **mixed encoding**. Let encodings for x_1 and x_2 be given by formulas

$$x'_{1i} = a_{1i} x_{1i} + b_{1i}$$
$$x'_{2i} = a_{2i} x_{2i} + b_{2i},$$

where

$$x_1 \equiv x_{1i} \mod p_i \quad \text{for} \quad 0 \le x_{1i} < p_i$$
$$x_2 \equiv x_{2i} \mod p_i \quad \text{for} \quad 0 \le x_{2i} < p_i.$$

Then multiplication $y = x_1 x_2$ can be expressed in terms of encoded data by formulas (similar to linear encoding) (see [10])

$$\tilde{y}_i = x'_{1i} x'_{2i} - b_{2i} x'_{1i} - b_{1i} x'_{2i}. \tag{35}$$

Than an adversary finds that

$$a_{1i} a_{2i} \mid (x'_{1i} x'_{2i} - b_{2i} x'_{1i} - b_{1i} x'_{2i} + b_{1i} b_{2i}). \tag{36}$$

and we have the same problem as in case of linear encoding. In this case the resistance is at most 16^m. In the case $m = 5$ it is not more then 2^{20}.

Now consider another variant **of mixed encoding**. In this case integers x_1 and x_2 are encoded by the formulas

$$a_{1i} x_{1i} + b_{1i} \equiv x'_{1i} \mod p_i$$
$$a_{2i} x_{2i} + b_{2i} \equiv x'_{2i} \mod p_i,$$

where $\mathrm{GCD}(a_{ki}, p_i) = 1$ and there are no other restrictions on x'_{1i} and x'_{2i} and coefficients a_{ki} and b_i. Then the product $y = x_1 x_2$ can be expressed in terms of encoded data by the formula (see [11])

$$\tilde{y} = \alpha_{i3} x'_{1i} x'_{2i} + \alpha_{1i} x'_{1i} + \alpha_{2i} x'_{2i}, \tag{37}$$

where

$$\begin{aligned} \alpha_{i3} &\equiv \gamma_i a_{1i} a_{2i} \mod p_i \\ \alpha_{i1} &\equiv \gamma_i a_{1i} a_{2i} b_2 \mod p_i \\ \alpha_{i3} &\equiv \gamma_i a_{1i} a_{2i} b_1 \mod p_i \end{aligned} \tag{38}$$

for some (arbitrary) γ_i such that $\mathrm{GCD}(\gamma_i, p_i) = 1$. Then adversary observes parameters

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \ldots, \alpha_{m1}, \alpha_{m2}, \alpha_{m3}.$$

In this case the function A of formula (8) is given by

$$A(a_{11}, b_{11}, \ldots, a_{1m}, b_{1m}, a_{21}, b_{21}, \ldots, a_{2m}, b_{2m}) =$$
$$(\alpha_{11}, \alpha_{12}, \alpha_{13}, \ldots, \alpha_{m1}, \alpha_{m2}, \alpha_{m3}).$$

It may be shown in this case that the resistance of such multiplication is at least $\nu(p_1) \cdot \cdots \cdot \nu(p_m)$ where ν is the Euler totient function.

7. Conclusions

The main contribution of this paper is the notion of measure of resistance of encoded computations we introduced. This gives the possibility to perform quantitative analysis of encoding schemes and to compare different data encodings. The results presented here show sufficiently high level of protection of data during the computations when one uses considered encoding schemes. Protection of data in encoded world is guaranteed by lower bounds of resistance we obtained. Typically the bounds are at least 2^{100} when operating with integers of range 2^{64}.

References

[1] A. A. Aho, J.E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company, 1976.

[2] D. E. Knuth, The Art of Computer Programming, vol.2, Seminumerical Algorithms, 1997.

[3] G. Birkhoff, T. C. Bartee, Modern Applied Algebra, McGrow-Hill Book Company, 1975.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to Algorithms, The MIT Press Cambridge, Massachusetts, London, England, 1997.

[5] S. T. Chow, H. J. Johnson, Yuan Gu, Tamper Resistant Software Encoding, O8-878835US, 1999.

[6] I. Niven, H. S. Zuckerman, An Introduction to the Theory of Numbers, Wiley, 1980.

[7] C. Collberg, C. Thomborson, D. Low, Manufacturing cheap, resilient and stealthy opaque constructs, *Symp. on Principles of Prog. Lang.*, 1998, p.184-196.

[8] M. Mambo, T. Murayama, E. Okamoto, A tentative approach to constructing tamper-resistant software, *Workshop on New Security Paradigms*, 1998, p.23-33.

[9] C. Wang, J. Hill, J. Knight, J. Davidson, Software tamper resistance: obstructing static analysis of programs, Tech. Report, N 12, Dep. of Comp. Sci., Univ. of Virginia, 2000.

[10] A.V. Shokurov, On encodings of integers and the division problem. Technical Report, Institute for Sistem Programming Russian Acad. of Sci., May 2000.

[11] A.V. Shokurov, On measures of resistance of data encodings. Intitute for Sistem Programming Russian Acad. of Sci., Technical Report, February 2001.

Appendix

Proof of Claim 1. We have (all computations are made modulo p_k)

$$\begin{aligned}
\sum_{i,j=1}^{n} \alpha_{ijk} x_{ik} x_{jk} &= \sum_{i,j=1}^{n} \alpha_{ijk} (c_{ik} x'_{ik} + d_{ik}) \cdot (c_{jk} x'_{jk} + d_{jk}) \\
&= \sum_{i,j=1}^{n} \alpha_{ijk} (c_{ik} c_{jk} x'_{ik} x'_{jk} + \\
&\qquad c_{ik} d_{jk} x'_{ik} + c_{jk} d_{ik} x'_{jk} + d_{ik} d_{jk}) \\
&= \sum_{i,j=1}^{n} \alpha_{ijk} c_{ik} c_{jk} x'_{ik} x'_{jk} + \\
&\qquad \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \alpha_{ijk} c_{ik} d_{jk} + \alpha_{jik} c_{ik} d_{jk} \right) x'_{ik} + \\
&\qquad \sum_{i,j=1}^{n} \alpha_{ijk} d_{ik} d_{jk} \\
&= \sum_{i,j=1}^{n} \alpha_{ijk} c_{ik} c_{jk} x'_{ik} x'_{jk} + \\
&\qquad \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (\alpha_{ijk} + \alpha_{jik}) c_{ik} d_{jk} \right) x'_{ik} + \sum_{i,j=1}^{n} \alpha_{ijk} d_{ik} d_{jk}.
\end{aligned}$$

For linear part of this expression we can write

$$\sum_{i=1}^{n} \beta_i x_i = \sum_{i=1}^{n} \beta_{ik} (c_{ik} x'_{ik} + d_{ik}) = \sum_{k=1}^{m} (\beta_{ik} c_{ik} x'_{ik} + \beta_{ik} d_{ik}). \tag{39}$$

Now let

$$\xi_{ijk} = \alpha_{ijk} c_{ik} c_{jk} \quad \text{for} \quad i, j = 1, \ldots, n \tag{40}$$

$$\zeta_{ik} = \sum_{j=1}^{n} (\alpha_{ijk} + \alpha_{jik}) c_{ik} d_{jk} + \beta_{ik} c_{ik} \quad \text{for} \quad k = 1, \ldots, m \tag{41}$$

$$\eta_k = \sum_{i,j=1}^{n} \alpha_{ijk} d_{ik} d_{jk} + \sum_{i=1}^{n} \beta_{ik} d_{ik} + \gamma_k. \tag{42}$$

Then the formula (22) for computing function f defined by equation (18) holds. ■

Proof of Claim 2. The observed data are coefficients $\xi_{ijk}, \zeta_{ik}, \eta_k$ of formula (22) and encoded data x'_{ik}. Take arbitrary numbers $\delta_{ik} \bmod p_k$ that coprime with p_k and change coefficients of the form (18) by relations $\tilde{\alpha}_{ijk} = \delta_{ik}\delta_{jk}\alpha_{ijk}$ and $\tilde{\beta}_{ik} = \delta_{ik}\beta_{ik}$. Then change the parameters of encoding by relations $\tilde{c}_{ik} = \delta_{ik}^{-1} c_{ik}$ and $\tilde{d}_{ik} = \delta_{ik}^{-1} d_{ik}$ where δ_{ik}^{-1} are modulo p_k inverse of δ_{ik}. Then from formulas (19)– (21) for $\tilde{\alpha}_{ijk}$ and $\tilde{\beta}_{ik}$

$$\tilde{\xi}_{ijk} = \tilde{\alpha}_{ijk}\tilde{c}_{ik}\tilde{c}_{jl} = \delta_{ik}\delta_{jk}\alpha_{ijk}\delta_{ik}^{-1}c_{ik}\delta_{jk}^{-1}c_{jk} = \alpha_{ijk}c_{ik}c_{jk} = \xi_{ijk}$$

$$\begin{aligned} \tilde{\zeta}_{ik} &= \sum_{j=1}^{n} (\tilde{\alpha}_{ijk} + \tilde{\alpha}_{jik})\tilde{c}_{ik}\tilde{d}_{jk} + \tilde{\beta}_{ik}\tilde{c}_{ik} \\ &= \sum_{j=1}^{n} (\delta_{ik}\delta_{jk}\alpha_{ijk} + \delta_{ik}\delta_{jk}\alpha_{jik})\delta_{ik}^{-1}c_{ik}\delta_{jk}^{-1}d_{jk} + \delta_{ik}\beta_{ik}\delta_{ik}^{-1}c_{ik} = \zeta_{ik} \end{aligned}$$

$$\begin{aligned} \tilde{\eta}_k &= \sum_{i,j=1}^{n} \tilde{\alpha}_{ijk}\tilde{d}_{ik}\tilde{d}_{jk} + \sum_{i=1}^{n} \tilde{\beta}_{ik}\tilde{d}_{ik} + \gamma_k \\ &= \sum_{i,j=1}^{n} \delta_{ik}\delta_{jk}\alpha_{ijk}\delta_{ik}^{-1}d_{ik}\delta_{jk}^{-1}d_{jk} + \sum_{i=1}^{n} \delta_{ik}\beta_{ik}\delta_{ik}^{-1}d_{ik} + \gamma_k = \eta_k. \end{aligned}$$

the coefficients of encoded form do not change. Thus encoded data and circuit don't change.

Prof of Statements 3 and 4 is similar to the proof of Claim 2.

Prof of Claim 5. We have

$$\begin{aligned} \sum_{i,j=1}^{n} \alpha_{ij} x_i x_j &= \sum_{i,j=1}^{n} \alpha_{ij} \left(\sum_{k=1}^{m} b_{ik} x'_k - b'_i \right) \cdot \left(\sum_{l=1}^{m} b_{jl} x'_l - b'_j \right) \\ &= \sum_{i,j=1}^{n} \alpha_{ij} \left(\sum_{i,j=1}^{m} b_{ik} b_{jl} x'_k x'_l - \right. \\ &\quad \left. b'_i \sum_{l=1}^{m} b_j l x'_l - b_j \sum_{k=1}^{m} b_{ik} x' k + b'_i b'_j \right) \\ &= \sum_{k,l=1}^{m} \left(\sum_{i,j=1}^{n} \alpha_{ij} b_{ik} b_{jl} \right) x'_k x'_l - \\ &\quad \sum_{k=1}^{m} \left(\sum_{i,j=1}^{n} \alpha_{ij} (b'_i b_{jk} + b'_j b_{ik}) \right) x'_k + \sum_{i,j=1}^{n} \alpha_{ij} b'_i b'_j \end{aligned} \tag{43}$$

For linear part of this expression we can write

$$\sum_{i=1}^{n} \beta_i x_i = \sum_{i=1}^{n} \left(\sum_{k=1}^{m} b_{ik} x'_k - b'_i \right) = \sum_{k=1}^{m} \left(\sum_{i=1}^{n} \beta_i b_{ik} \right) x'_k - \sum_{i=1}^{n} \beta_i b'_i. \tag{44}$$

Now denote by

$$\xi_{kl} = \sum_{i,j=1}^{n} \alpha_{ij} b_{ik} b_{jl} \quad \text{for} \quad k, l = 1, \ldots, m \tag{45}$$

$$\zeta_k = \sum_{i=1}^{n} \beta_i b_{ik} - \sum_{i,j=1}^{n} \alpha_{ij} (b'_i b_{jk} + b' j b_{ik}) \quad \text{for} \quad k = 1, \ldots, m \tag{46}$$

$$\eta = \sum_{i,j=1}^{n} \alpha_{ij} b'_i b'_j - \sum_{i=1}^{n} \beta_i b'_i. \tag{47}$$

Then the formula (34) for computing function f defined by equation (30) holds. ■

Prof of Claim 6 is similar to the proof of Claim 2.