05/15/2005

Geometric Cryptosystem

Arkady Berenstein[1] & Leon Chernyak[2]

Abstract

In this paper we propose a new class of cryptosystems that utilizes metric continuity. The geometric cryptosystem considered in this paper as the main example of metric cryptosystems has a number of interesting properties such as resistance to several basic cryptographic attacks, efficiency and detection of transmission errors.

Table of Content

[1] Affiliation: University of Oregon, Department of Mathematics
[2] Affiliation: Institute for Genometric Systems

Introduction

Many existing cryptosystems are based on mathematical concepts such as Boolean functions, finite fields, discrete groups, etc. AES, for example, utilizes Galois theory of finite fields. In more recent works such as [7], [8], and [10], more advanced mathematical concepts of arithmetical and algebraic geometry were employed.

In this paper we propose a class of cryptographic systems based on mathematical concept of geometric continuity. The main feature of such continuous cryptosystems is that plaintexts and ciphertexts are elements of such domains as real numbers or real vector spaces, and encryption/decryption procedures involve continuous transformations of these domains. Apparently, this attempt to utilize the geometric continuity in cryptography is new. So far we have found only one prior work [4] that somehow employs continuity: the Ajtai-Dwork cryptosystem introduced in this paper deals with a collection of parallel hyperplanes in *n*-dimensional vector space.

Informally speaking, continuity is a natural way to introduce randomness, which is important for security of cryptosystems (see e.g., [6]). Randomization of keys deflects brute force attacks against the keys and randomization of ciphertexts deflects cryptanalysis based on intercepted ciphertexts.

Speaking mathematically, a *continuous cryptographic system* (CCS) is organized as follows. Each plaintext is a point of a topological space X; each ciphertext is a point of a topological space Y; and each encryption/decryption procedure is a continuous isomorphism between X and Y. We will rigorously introduce continuous cryptosystems and discuss their properties in Section 1. (To enhance GCS security, it is reasonable to assume that prior to the moment GCS begins its work nearby texts in a natural language are transformed into mutually distant vectors)

Since a typical topological space is uncountable, like *n*-dimensional space $\mathbb{R}^n$, CCS is not a "real," but rather an "ideal" cryptosystem.

In order to make CCS a working cryptosystem, the continuous objects are to be "discretized." In Section 2 we introduce a concept of *metric cryptosystem* (MCS) in which such "discretization" is performed as a certain rounding procedure. Informally speaking, a metric cryptosystem is a continuous cryptosystem enhanced with the following data:

1. a certain finite subset $\mathbb{P}$ of X is designated as the space of plaintexts;
2. a certain finite subset $\mathbb{C}$ of Y is designated as the space of ciphertexts;
3. each encryption/decryption procedure is followed by *rounding* (which guarantees the transformation $\mathbb{P} \rightarrow \mathbb{C}$).

We will rigorously define metric cryptosystems and discuss their properties in Section 2.

We expect that the discrete cryptosystem derived from each MCS, which is built upon any strong topology, will posses a remarkable property of *natural randomization* of ciphertexts. Unlike, for example, in Blowfish, the discrete cryptosystem derived from MCS does not pursue any special randomization strategy. MCS-induced randomization is natural in that sense that it is generated by the very computation of a continuous function followed by customary procedures of rounding of results of the computations.

Our expectation was confirmed by what we consider to be the most interesting example of MCS – a cryptosystem that we call Geometric Cryptosystem (GCS).

Another benefit of each MCS is its efficiency in detecting errors during transmissions of ciphertexts. Consider, for example, the situation when a plaintext is presented by a *discrete* point in $\mathbb{R}^n$ (i.e., by a point that belongs to the integer lattice $\mathbb{Z}^n$ in $\mathbb{R}^n$, where $\mathbb{Z}$ denotes the set of all integers). The corresponding ciphertext produced by a continuous transformation is not discrete anymore. Therefore, if an error has occurred during the transmission of the ciphertext, it is highly unlikely that the decryption procedure will result in a discrete point. Such a non-discrete result of decryption would indicate a transmission error.

Below we propose an implementation of MCS, which we call Geometric Cryptosystem (GCS). Each encryption/decryption key of GCS consists of two parts: a *outer component* and a *inner component*. The role of the outer component is played by a set of discrete data that is a finite sequence of positive integers. The the inner component is represented by continuous data, which is an m-tuple of vectors $(v_1,v_2,\dots, v_m)$ of the vector space $\mathbb{Z}^n$.

The encryption and decryption techniques of GCS are mutually symmetric and require the same time, amount of memory, and computational power. The same device can work both as the encryption and the decryption device. Only the outer component of the key determines in which mode, i.e., encryption or decryption, the device is currently working. Namely, the outer component of the key used for encryption of a message can be transmitted along with the encrypted message so that the receiving device uses this outer component for decryption of this message.

The creation of an encryption transformation (from the space of plaintexts to the space of ciphertexts) requires a choice of both an outer component and an inner component. Once the inner component is chosen, the encryption/decryption process depends solely on the choice of outer component.

However, the decryption transformation (from the space of ciphertexts to the space of plaintexts) cannot be reconstructed based solely on the outer component. The inner component of the key can be chosen at random and is kept in secret. The inner component is hard to reconstruct because of its randomness. Attacks (considered in Appendix C) aiming at reconstruction of the inner component are not efficient.

Therefore, GCS allows for a dynamic key update without changing inner components. Instead of such change, the outer component of the encryption key, as embedded into a transmitted message, determines a new decryption key. The key, in its turn, triggers generation of a new decryption transformation. Actually, any transmitted message may trigger a new decryption key generation.

GCS has recently been implemented in C++ with $n = m = l = 4$. The results are very encouraging: the speed of (not yet optimized) GCS on Pentium-IV system resulted in the encryption/decryption speed of 215Mb per sec with the used memory approximately equal to the size of the plaintext messages. It is at least two times faster than the best results for AES 256. For comparison of GCS versus AES performance see *Appendix A*.

Acknowledgements. The authors express their gratitude to Igor Mendelev for invaluable help in implementation of the first prototype of GCS and for performing the comparative analysis of GCS prototype with the Advanced Encryption Standard (AES).

Our thanks are due to Professor Itkis of Boston University for his comments and remarks on this manuscript.

Section 1. Continuous and Metric Cryptosystems

In this section we propose a general concept of ***continuous cryptosystem***. Our discussion of possible implementations of continuous cryptosystems in a discrete medium, such as a computer, will result in concept of ***metric cryptosystem***.

Let us start with a standard (textbook-like) definition of a cryptosystem.

Definition 1.1 (discrete). A cryptosystem is a 5-tuple ($\mathbb{P}, \mathbb{C}, \mathbb{K}, E, D$) where:

- $\mathbb{P}$ is the set of possible plaintexts (i.e., texts that are not encrypted),
- $\mathbb{C}$ is the set of possible ciphertexts (i.e., texts that are encrypted)
- $\mathbb{K}$ is the set of possible keys
- E is the set of encryption rules depending on a key: if $\boldsymbol{k}$ is an element of $\mathbb{K}$ then $e_{\boldsymbol{k}}$: $\mathbb{P} \rightarrow \mathbb{C}$ is the corresponding encryption rule
- D is the set of decryption rules depending on a key: $d_{\boldsymbol{k}}$: $\mathbb{C} \rightarrow \mathbb{P}$ which verifies that the composition $d_{\boldsymbol{k}} \circ e_{\boldsymbol{k}}$ is the identity map $\mathbb{P} \rightarrow \mathbb{P}$.

Next, we define a cryptographic system that is a continuous analogue of the cryptosystem of Definition 1.1.

Definition 1.2 (continuous). A *continuous cryptographic system* (CCS) is a 5-tuple (X, Y, K, e, d), where

- X is a topological space of plaintexts,
- Y is a topological space of ciphertexts (Y is isomorphic to X),
- K is a topological space of keys,
- e: K×X$\rightarrow$ Y is a continuous map – the set of encryption rules, and
- d: K×Y $\rightarrow$ X is a continuous map – the set of decryption rules,
- the encryption and decryption rules are mutually inverse: for each $\boldsymbol{k}$ in K and each $\boldsymbol{x}$ in X, and each $\boldsymbol{y}$ in Y one has:

$$d(\boldsymbol{k}, e(\boldsymbol{k}, \boldsymbol{x})) = \boldsymbol{x}, \quad e(\boldsymbol{k}, d(\boldsymbol{k}, \boldsymbol{y})) = \boldsymbol{y}. \tag{1.1}$$

The encrypted message is the pair (k, $e(\boldsymbol{k}, \boldsymbol{x})$), or simply $e(\boldsymbol{k}, \boldsymbol{x})$ if the encryption key $\boldsymbol{k}$ is known to the recipient.

It follows from Definition 1.2 that the *encryption transformation* $\mathrm{T}_{\boldsymbol{k}}$: X $\rightarrow$ Y, defined by $\mathrm{T}_{\boldsymbol{k}}(\boldsymbol{x}) = e(\boldsymbol{k}, \boldsymbol{x})$, and the *decryption transformation* $\mathrm{T}_{\boldsymbol{k}}'$: Y $\rightarrow$ X, defined by $\mathrm{T}_{\boldsymbol{k}}'(\boldsymbol{y}) = d(\boldsymbol{k}, \boldsymbol{y})$, are mutually inverse (in particular, the spaces X and Y are isomorphic).

Remark. In GCS (see Section 3) only a part of key $\boldsymbol{k}$ (so called ‘inner component’) is known to the recipient and only a part of the key $\boldsymbol{k}$ (so called ‘outer component’) is being transmitted along with the ciphertext $e(\boldsymbol{k}, \boldsymbol{x})$.

It is easy to see that our Definition 1.2 of CCS is a continuous analog of the standard (discrete) Definition 1.1. Moreover, since any set S is a topological space with respect to the *discrete* topology (i.e., when each subset of S is declared open) and any map between sets is, therefore, continuous, any cryptosystem satisfying the Definition 1.1 is a particular case of CCS.

However, we are not interested here in this trivial discrete topology. The most interesting examples of CCS are built upon a topology as strong as the topologies of real line $\mathbb{R}$ or that of the n-dimensional space $\mathbb{R}^n$. Another example of a strong enough topology is that of p-adic numbers for each prime p (see, e.g., [1]).

However, CCS built upon such strong topologies as $\mathbb{R}^n$ (or that of p-adic numbers) has obviously uncountably many plaintexts and ciphertexts. In order to "materialize" such a strong CCS in a discrete medium, one has to address the following challenges.

1. One has to deal with actual (i.e., discrete) plaintexts and ciphertexts rather than with continuous ones.
2. One has to make sure that, despite of the continuous nature of each encryption transformation, encryption of an actual (i.e., discrete) plaintext will always result in an actual (i.e., discrete) ciphertext. The same for decryption.

As a response to these challenges, i.e., as an implementation of CCS in a discrete medium, we propose a *metric cryptosystem*.

Definition 1.3 (metric). A metric cryptosystem (MCS) is a 10-tuple

$$(X, Y, K, e, d; \mathbb{P}, \mathbb{C}, \mathbb{K}; \mathrm{Round}_{\mathbb{P}}, \mathrm{Round}_{\mathbb{C}}),$$

where (X, Y, K, e, d) is a CCS, in which X is a metric space (with topology induced by the metric), Y is another metric space (with topology induced by the metric); K, e, d are as in CCS, and

- $\mathbb{P}$ is a discrete subset of X,
- $\mathbb{C}$ is a discrete subset of Y,
- $\mathbb{K}$ is a discrete subset of K,
- $\mathrm{Round}_{\mathbb{P}}$: X → $\mathbb{P}$ is a map such that for each point $x \in$ X the point $p = \mathrm{Round}_{\mathbb{P}}(\boldsymbol{x})$ is the closest to x among all point $p' \in \mathbb{P}$ (in particular, $\mathrm{Round}_{\mathbb{P}}(p) = p$ for any $p \in \mathbb{P}$);
- $\mathrm{Round}_{\mathbb{C}}$: Y → $\mathbb{C}$ is a map such that for each point $y \in$ Y the point $c = \mathrm{Round}_{\mathbb{C}}(\boldsymbol{y})$ is the closest to y among all point $c' \in \mathbb{C}$ (in particular, $\mathrm{Round}_{\mathbb{C}}(c) = c$ for any $c \in \mathbb{C}$);
- It is required that for each key $\boldsymbol{k}$ in $\mathbb{K}$ and for each $p \in \mathbb{P}$ one has

$$\mathrm{Round}_{\mathbb{P}}(d(\boldsymbol{k}, \mathrm{Round}_{\mathbb{C}}(e(\boldsymbol{k}, p)))) = p\,. \tag{1.2}$$

Note that each standard system of Definition 1.1 is tautologically a metric system of the Definition 1.3 with the discrete X = $\mathbb{P}$, discrete Y = $\mathbb{C}$ (and, therefore, and $\mathrm{Round}_{\mathbb{P}}$ and $\mathrm{Round}_{\mathbb{C}}$ are identity transformations). However, we introduced metric cryptosystems for more interesting "discretezations."

Indeed, the identity (1.2) implies that in MCS the continuous encryption and decryption rules e: K×X→ Y and d: K×Y → X of the underlying CCS are replaced by the following discrete encryption and decryption rules respectively: $e_{\text{discr}}: \mathbb{K} \times \mathbb{P} \to \mathbb{C}$ and $d_{\text{discr}}: \mathbb{K} \times \mathbb{C} \to \mathbb{P}$:

(1.3) $$e_{\text{discr}}(\boldsymbol{k}, p) = \text{Round}_{\mathbb{C}}(e(\boldsymbol{k}, p)), \quad d_{\text{discr}}(\boldsymbol{k}, c) = \text{Round}_{\mathbb{P}}(d(\boldsymbol{k}, c)).$$

The identity (1.2) can, therefore, be rewritten as

$$d_{\text{discr}}(\boldsymbol{k}, e_{\text{discr}}(\boldsymbol{k}, p)) = p.$$

This argument proves that the procedure of rounding turns our MCS into a conventional discrete cryptosystem - the 5-tuple ($\mathbb{P}, \mathbb{C}, \mathbb{K}, e_{\text{discr}}, d_{\text{discr}}$) as defined in Definition 1.1. We will refer to this ($\mathbb{P}, \mathbb{C}, \mathbb{K}, e_{\text{discr}}, d_{\text{discr}}$) as the discrete cryptosystem derived from MCS.

Remark. In terms of the encryption transformation $T_{\boldsymbol{k}}$:X→Y, defined by $T_{\boldsymbol{k}}(\boldsymbol{x}) = e(\boldsymbol{k}, \boldsymbol{x})$, and the decryption transformation $T_{\boldsymbol{k}}'$: Y → X, the formula (1.2) takes the form:

(1.4) $$\text{Round}_{\mathbb{P}}(T_{\boldsymbol{k}}'(\text{Round}_{\mathbb{C}}(T_{\boldsymbol{k}}(p)))) = p,$$

for each key $\boldsymbol{k}$ in $\mathbb{K}$ and for each $p \in \mathbb{P}$.

Section 2. GCS as a continuous cryptosystem (informal description)

In this section we informally define Geometric Cryptosystem (GCS) as a continuous cryptosystem in which X = Y = $\mathbb{R}^n$, where $\mathbb{R}^n$ is the n-dimensional Euclidean vector space. Therefore, both the plaintexts and the ciphertexts are (continuous) points of an n-dimensional Euclidean vector space $\mathbb{R}^n$. In what follows we will construct the encryption rule in the form of collection of encryption transformations $T_{\boldsymbol{k}}: \mathbb{R}^n \to \mathbb{R}^n$ for each key $\boldsymbol{k}$. Respectively, the decryption transformation $T_{\boldsymbol{k}}': \mathbb{R}^n \to \mathbb{R}^n$ will always be the inverse of the encryption transformations.

Here is the construction of $T_{\boldsymbol{k}}$ (for relevant definitions on reflection groups see, e.g., [2], or Section 3 below).

1. In the vector space $\mathbb{R}^n$ a collection of m mirrors, i.e., of $(n-1)$-dimensional subspaces, is chosen. These mirrors are enumerated by numbers 1, 2, …, m.
2. The collection of m mirrors is parametrized by a sequence $\boldsymbol{v}=(v_1, v_2, \ldots, v_m)$, where v_1 is a non-zero vector orthogonal to the mirror 1, v_2 is a non-zero vector orthogonal to the mirror 2, etc.
3. The encryption transformation $T_{\boldsymbol{k}}$ is defined as a sequence of reflections performed in the order $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$, where each of these indices i_j is a number of an involved mirror, that is, the mirror i_l is reflected about first, then i_{l-1}, etc., and finally the mirror i_1 is applied. The same mirror can be used more than one time in the

sequence $\boldsymbol{i}$ (e.g., if $\boldsymbol{i} = (1, 2, 1)$, i.e., the mirror 1 is used twice – the first time at the beginning, and second time at the end).

4. Therefore, the encryption key $\boldsymbol{k}$ is comprised of two components: the *inner* component $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ and the *outer* component $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$, i.e., $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$.
5. For each (continuous) plaintext $\boldsymbol{x}$ of $\mathbb{R}^n$ its image $T_{\boldsymbol{k}}(\boldsymbol{x})$ is a corresponding (continuous) ciphertext. The pair $(\boldsymbol{i}, T_{\boldsymbol{k}}(\boldsymbol{x}))$ is the encrypted message, in which only the outer component $\boldsymbol{i}$ of the key $\boldsymbol{k}$ is transmitted along with the ciphertext $T_{\boldsymbol{k}}(\boldsymbol{x})$. That is, the mirrors are not revealed during transmission of the encrypted message.
6. Decryption transformation $T_{\boldsymbol{k}}' = T_{\boldsymbol{k}}^{-1}$ is defined as "undoing" the reflections of the sequence $\boldsymbol{i}$, that is, as applying the same reflections in the opposite order: the mirror i_1 is reflected about first, then i_2, etc., and finally the mirror i_l is applied (e.g., if $\boldsymbol{i} = (1, 2, 1, 3)$, then $T_{\boldsymbol{k}}'$ is obtained by applying first the mirror 1, second – the mirror 2, third – the mirror 1 again, and forth – the mirror 3).
7. This definition of the decryption transformation $T_{\boldsymbol{k}}' = T_{\boldsymbol{k}}^{-1}$ implies that $T_{\boldsymbol{k}}'$ has the same structure as $T_{\boldsymbol{k}}$, namely, $T_{\boldsymbol{k}}' = T_{\boldsymbol{k}'}$, where $\boldsymbol{k}'$ is the key comprised of the same inner component $\boldsymbol{v}$ and a new outer component $\boldsymbol{i}' = (i_l, i_{l-1}, \ldots, i_1)$, that is, is $\boldsymbol{i}'$ is the reversed sequence $\boldsymbol{i}$.
8. We will refer to this new $\boldsymbol{k}' = (\boldsymbol{v}, \boldsymbol{i}')$ as to *decryption* key.
9. In a general GCS (see Section 3 below), the mirrors are "curved," i.e., the reflections about the mirrors are twisted with some invertible transformations g_i of the space $\mathbb{R}^n$.

Section 3. GCS as a continuous cryptosystem (formal description)

In this section we provide a formal definition of Geometric Cryptosystem (GCS) within the framework of continuous cryptosystems introduced in Section 1 (GCS as metric cryptosystem will be addressed in Section 4 and Appendix B).

Geometric Cryptosystem (GCS) is a continuous cryptosystem (X, Y, K, *e*, *d*) (as in Section 1, Definition 1.2) in which:

- $X = Y = \mathbb{R}^n$, which is the real n-dimensional vector space, $n \geq 4$.

- K is a space of keys: $K = K_{\text{inner}} \times K_{\text{outer}}$, where each element of the set K_{inner} is a m-tuple $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ of non-zero vectors $v_1, v_2, \ldots, v_m$ in $\mathbb{R}^n$, $m \geq n$; and each element of K_{outer} is a *repetition-free* sequence $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ of indices such that $n \leq l \leq m$, where each of these indices i_j belongs to the set $\{1, 2, \ldots, m\}$; thus each element of K is a pair $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$, where $\boldsymbol{v}$ and $\boldsymbol{i}$ are as above.

- e: $K \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a continuous map defined for each $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$ and $\boldsymbol{x}$ in $\mathbb{R}^n$ by the formula:

$$e((\boldsymbol{v}, \boldsymbol{i}), \boldsymbol{x}) = S_{i_1} \circ S_{i_2} \circ \ldots \circ S_{i_l}(\boldsymbol{x}), \tag{3.1}$$

where each S_j: $\mathbb{R}^n \rightarrow \mathbb{R}^n$, $j = 1, 2, \ldots m$, is the reflection about the hyperplane (i.e., the mirror) H_j orthogonal to the vector v_j, that is,

(3.2) $$S_j(x) = \boldsymbol{x} - [2(\boldsymbol{x},v_j)/(v_j, v_j)]\cdot v_j$$

for all $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ in $\mathbb{R}^n$, where $(\boldsymbol{x}, \boldsymbol{y})$ stands for a given inner product in $\mathbb{R}^n$. In the most of cases we shall employ the standard Euclidean dot product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$, that is,

(3.3) $$(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \bullet \boldsymbol{y} = \textstyle\sum_j x_j y_j,$$

where summation is over $j = 1, 2, \ldots, n$.

- d: K× $\mathbb{R}^n \rightarrow \mathbb{R}^n$, is a continuous map defined for each $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$ and $\boldsymbol{x}$ in $\mathbb{R}^n$ by the formula:

(3.3) $$d((\boldsymbol{v}, \boldsymbol{i}), \boldsymbol{y}) = e((\boldsymbol{v}, \boldsymbol{i'}), \boldsymbol{y}),$$

where $\boldsymbol{i'} = (i_l, i_{l-1}, \ldots, i_1)$, that is, is $\boldsymbol{i'}$ is the reversed sequence $\boldsymbol{i}$.

Note that the formula (3.1) for the encryption rule $\boldsymbol{e}$ can be rewritten as

(3.5) $$e((\boldsymbol{v}, \boldsymbol{i}), \boldsymbol{x}) = \mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i})}(\boldsymbol{x}),$$

where $\mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i})}$ is a transformation of $\mathbb{R}^n$ given by the formula:

(3.6) $$\mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i})} = \mathrm{S}_{i_1} \circ \mathrm{S}_{i_2} \circ \ldots \circ \mathrm{S}_{i_l},$$

and where each S_j is a transformation of $\mathbb{R}^n$ given by (3.2).

Following the discussion after Definition 1.2, we will refer to this $\mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i})}$ as *encryption transformation* of GCS.

Therefore, the formula (3.4) for the decryption rule $\boldsymbol{d}$ can be rewritten as

(3.7) $$d((\boldsymbol{v}, \boldsymbol{i}), \boldsymbol{y}) = \mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i'})}(\boldsymbol{y}),$$

where $\boldsymbol{i'} = (i_l, i_{l-1}, \ldots, i_1)$, that is, is $\boldsymbol{i'}$ is the reversed sequence $\boldsymbol{i}$ (as in (3.4)).

Following the discussion after Definition 1.2, we will refer to this $\mathrm{T}_{(\boldsymbol{v}, \boldsymbol{i'})}$ as *decryption transformation* of GCS.

For each encryption key $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$, we refer to $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ as the *inner component* of $\boldsymbol{k}$ and to $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ as the *outer component* of $\boldsymbol{k}$.

Remark. We choose the standard dot product in (3.3) only for the sake of simplicity. Any other inner product in Rn would work here as well.

Remark. The set K_{outer} of repetition-free sequences is finite: the number of repetition-free sequences $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ of length l is

$$m \cdot (m-1) \cdot \ldots \cdot (m-l+1) = m!/(m-l)!, \tag{3.8}$$

(the length l of any repetition-free sequences on the set $\{1, 2, \ldots, m\}$ never exceeds m).

Remark. We can consider a larger class of outer components by allowing any sequence $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ of indices, where each of these indices i_j belongs to the set $\{1, 2, \ldots, m\}$, with the only restriction that $i_j \neq i_{j+1}$, for $j = 1, 2, \ldots, l-1$. Unlike repetition-free sequences, this larger class of possible outer components is infinite, and the length l of $\boldsymbol{i}$ is arbitrary. For this larger class of the outer components the numbers m and l may be chosen arbitrarily (i.e., there do not have to satisfy the above inequalities $n \leq 1 \leq m$).

Remark. If the vector v_j is of the unit length, i.e., if $(v_j, v_j) = 1$, the above formula (3.2) for S_j gets simplified as follows.

$$S_j(x) = \boldsymbol{x} - 2(\boldsymbol{x}, v_j)\cdot v_j\,. \tag{3.9}$$

Remark. The decryption transformation $T_{(\boldsymbol{v},\, \boldsymbol{i}')}$ is equal to $T_{(\boldsymbol{v},\, \boldsymbol{i})}^{-1}$ because each reflection S_j is self-invertible, i.e., $S_j^{-1} = S_j$. More precisely,

$$T_{(\boldsymbol{v},\boldsymbol{i})}^{-1} = (S_{i_1} \circ S_{i_2} \circ \ldots \circ S_{i_l})^{-1} = S_{i_l}^{-1} \circ S_{i_{l-1}}^{-1} \circ \ldots \circ S_{i_1}^{-1} = S_{i_l} \circ S_{i_{l-1}} \circ \ldots \circ S_{i_1} = T_{(\boldsymbol{v},\, \boldsymbol{i}')}, \tag{3.10}$$

where $\boldsymbol{i}' = (i_l, i_{l-1}, \ldots, i_1)$.

Remark. By definition, each S_j is an orthogonal transformation of $\mathbb{R}^n$ (recall that a transformation f of $\mathbb{R}^n$ is orthogonal if $(f(\boldsymbol{x}), f(\boldsymbol{y})) = (\boldsymbol{x}, \boldsymbol{y})$ for any $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$). Therefore, both the encryption transformation $T_{(\boldsymbol{v},\, \boldsymbol{i})}$ and the decryption transformation $T_{(\boldsymbol{v},\boldsymbol{i}')}$ are orthogonal. This happens because each of them is a composition of orthogonal transformations.

Remark. It is well known that the group $O_n(\mathbb{R})$ of the orthogonal transformation of $\mathbb{R}^n$ is a compact topological group. Thus, each encryption transformation $T_{(\boldsymbol{v},\, \boldsymbol{i})}$ and each decryption transformation $T_{(\boldsymbol{v},\boldsymbol{i}')}$ belongs to this compact group $O_n(\mathbb{R})$. (Below, in Appendix C, where we consider possible cryptographic attacks against GCS, we shall discuss this compact group in more details).

Remark. We can propose a generalization of the encryption transformation (3.6) by defining transformations $T_{(\boldsymbol{g},\, \boldsymbol{v},\, \boldsymbol{i})}$ given by the formula:

$$T_{(\boldsymbol{g},\, \boldsymbol{v},\, \boldsymbol{i})} = g_{i_1} \circ S_{i_1} \circ g_{i_1}^{-1} \circ g_{i_2} \circ S_{i_2} \circ g_{i_2}^{-1} \circ \ldots \circ g_{i_l} \circ S_{i_l} \circ g_{i_l}^{-1}\,. \tag{3.11}$$

where $g_1, g_2, \ldots, g_m$ are the invertible transformations of $\mathbb{R}^{\boldsymbol{n}}$. It is easy to see that each $T_{(\boldsymbol{g},\, \boldsymbol{v},\, \boldsymbol{i})}$ is an invertible transformation of $\mathbb{R}^n$ and the inverse of $T_{(\boldsymbol{g},\, \boldsymbol{v},\, \boldsymbol{i})}$ is $T_{(\boldsymbol{g},\, \boldsymbol{v},\, \boldsymbol{i}')}$ which agrees with (3.10). Clearly the encryption transformation (3.6) is a particular case of (3.11) where each g_j is the identity transformation of $\mathbb{R}^{\boldsymbol{n}}$. Therefore, the continuous cryptosystem with the keys of the form $\boldsymbol{k} = (\boldsymbol{g}, \boldsymbol{v}, \boldsymbol{i})$ and the encryption transformations $T_{\boldsymbol{k}} = T_{(\boldsymbol{g}, \boldsymbol{v}, \boldsymbol{i})}$ is a generalization of the continuous cryptosystem GCS.

However, the metric property (1.4) of this more general cryptosystem is unknown.

Section 4. GCS as a metric cryptosystem

GCS, as a continuous cryptosystem, has been described in the two previous sections. Here we will describe it as a metric cryptosystem. For this purpose we introduce:

1. The metric on $X = Y = \mathbb{R}^{\boldsymbol{n}}$: this metric is a standard Euclidean distance.
2. the set $\mathbb{P}$ of plaintexts: $\mathbb{P} = \mathbb{Z}^n$, i.e., $\mathbb{P}$ is the set of all vectors $p = [p_1, p_2, \ldots, p_n]$ with integer coordinates.
3. the set $\mathbb{C}$ of ciphertexts: $\mathbb{C} = \mathbb{C}_r = [1/2^r] \cdot \mathbb{Z}^n$ for a certain positive integer r, that is, $\mathbb{C}_r$ consists of binary rational vectors $c = [c_1, c_2, \ldots, c_n]$ such that $2^r \cdot c_i$ is an integer for $\boldsymbol{i} = 1, 2, \ldots, n$.
4. the set $\mathbb{K}$ of keys: $\mathbb{K} = \mathbb{K}_{\text{inner}} \times \mathbb{K}_{\text{outer}}$, where each element of the set $\mathbb{K}_{\text{inner}}$ is a m-tuple $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ of non-zero vectors $v_1, v_2, \ldots, v_m$ in $\mathbb{Z}^n$, $m \geq n$; and each element of $\mathbb{K}_{\text{outer}}$ is a *repetition-free* sequence $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ of indices, where each of these indices i_j belongs to the set $\{1, 2, \ldots, m\}$; thus each element of $\mathbb{K}$ is a pair $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$, where $\boldsymbol{v}$ and $\boldsymbol{i}$ are as above.
5. Rounding of plaintexts $\text{Round}_{\mathbb{P}}: X \to \mathbb{P}$: We define $\text{Round}_{\mathbb{P}}$ to be the ordinary rounding of vectors of $X = \mathbb{R}^{\boldsymbol{n}}$ to the vectors in $\mathbb{P} = \mathbb{Z}^n$. (For example, if $n = 4$ and $\boldsymbol{x}$ is a *binary* vector of the form $\boldsymbol{x} = [10.11, 11.1, -1011.0011, -1.101101]$, then $\text{Round}_{\mathbb{P}}(\boldsymbol{x}) = [11, 100, -1011, -10]$).
6. Rounding of ciphertexts $\text{Round}_{\mathbb{C}}: Y \to \mathbb{C}$: We define $\text{Round}_{\mathbb{C}}$ by the formula:

   $$\text{Round}_{\mathbb{C}}(\boldsymbol{y}) = [1/2^r] \cdot \text{Round}_{\mathbb{P}}(2^r \cdot \boldsymbol{y}).$$

   In other words, $\text{Round}_{\mathbb{C}}(\boldsymbol{y}) = [c_1, c_2, \ldots, c_n]$, where each c_i is the ordinary rounding of x_i to r correct binary places after the binary dot. (For example, if n=4, r=1, and $\boldsymbol{y}$ is a *binary* vector of the form $\boldsymbol{y} = [10.11, 11.1, -1011.0011, -1.101101]$, then $\text{Round}_{\mathbb{C}}(\boldsymbol{y}) = [11, 11.1, -1011, -1.1]$).

In what follows we assume that a device computing encryption/decryption transformations has an *unlimited precision*, i.e., if each coordinate of a vector $\boldsymbol{x}$ is between -2^N and $+2^N$, then the error in computing each (continuous) ciphertext $T_{(\boldsymbol{v},\boldsymbol{i})}(p)$ can be made smaller than, for instance, 2^{-2N}.

Theorem 4.1. Assume that r is a positive integer such that $2^{2^r} > n$ (that is, $r > ½\log_2(n)$). Then the requirement (1.2) holds, that is, for each key $\boldsymbol{k} = (\boldsymbol{v}, \boldsymbol{i})$ in $\mathbb{K}$ and for each $p \in \mathbb{P}$ one has

(4.1) $$\mathrm{Round}_{\mathbb{P}}\,(\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}{}^{-1}(\,\mathrm{Round}_{\mathbb{C}}\,(\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}(p)))) = p$$

(where $\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}$ is the encryption transformation given by (3.5)).

In what follows we prove Theorem 4.1 by producing a more general statement (Proposition 4.2) that defines the restrictions for the computational device guaranteeing the formula (4.1). To formulate Proposition 4.2 we need to introduce some additional notation.

Recall that each key $\boldsymbol{k}$ of GCS is comprised of its inner component $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ and its outer component $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$. In what follows we fix the inner component $\boldsymbol{v}$, and therefore, each encryption or decryption rule will depend only on the outer component $\boldsymbol{i}$. Then $e(\boldsymbol{k}, \boldsymbol{x}) = \mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}(x)$, where $\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}$ is a transformation of $\mathbb{R}^n$ given by the formula (3.5).

Let $\boldsymbol{e}_i$ be the unit length vector in the direction of v_i, that is,

(4.2) $$\boldsymbol{e}_i = v_i/||v_i||,$$

for $\boldsymbol{i} = 1, 2, \ldots, m$, where $||\boldsymbol{x}||$ stands for the length of $\boldsymbol{x}$, that is $||\boldsymbol{x}||^2 = (\boldsymbol{x}, \boldsymbol{x})$.

We will replace the inner component $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ by the normalized inner component $\boldsymbol{e} = (e_1, e_2, \ldots, e_m)$. Clearly

(4.3) $$\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})} = \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}\,.$$

We proceed with the assumption that, while each v_i is precise, its unit vector $\boldsymbol{e}_i$ is computed with the error

(4.4) $$\Delta\boldsymbol{e}_i$$

for $i = 1, 2, \ldots, m$ (that is, we do not assume here that the computational device is absolutely precise).

Next we need to define the scalar magnitude ε_i of the vector $\Delta\boldsymbol{e}_i$:

(4.5) $$\varepsilon_i = 4\cdot||\Delta\boldsymbol{e}_i|| + 2\cdot||\Delta\boldsymbol{e}_i||^2$$

for $i = 1, 2, \ldots, m$.

And the total scalar error ε in the direction $\boldsymbol{i}$:

(4.6) $$\varepsilon = \varepsilon_{i_1} + \varepsilon_{i_2} + \ldots + \varepsilon_{i_l}$$

(by this definition, ε is always non-negative, and ε becomes 0 if and only if there is no computational error at all).

In the *Proposition 4.2* we shall work with plaintext vectors $p = [p_1, p_2, \ldots, p_n]$ such that each coordinate p_i of p is an integer satisfying

(4.7) $$-2^{N} < p_i < 2^{N}$$

for $i = 1, 2, \ldots, n$, (we will view each p_i as a binary integer having at most N binary digits).

Finally, we shall proceed with the assumption that

(4.8) $$\varepsilon \cdot (2 + \varepsilon) \cdot (2^{N} - 1)) \cdot \sqrt{n} < ½ .$$

Remark. Formula (4.8) demonstrate that in a real device, that is, when ε > 0, the magnitude N of the plaintext cannot be arbitrarily big, because, if $N = +\infty$, then ε must be zero, i.e. the device must be absolutely precise.

Now we can state the main technical result of the section, which will immediately imply Theorem 4.1.

Proposition 4.2. With the notations and assumptions (4.2) – (4.8), let r be any non-negative integer such that the following inequality holds.

(4.9) $$((\varepsilon + 1) \cdot 2^{-r-1} + \varepsilon \cdot (2 + \varepsilon) \cdot (2^{N} - 1)) \cdot \sqrt{n} < ½$$

Then (4.1) holds for each vector p satisfying (4.7), that is,

$$\mathrm{Round}_{\mathbb{P}} (\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\mathrm{Round}_{\mathbb{C}} (\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(p)))) = p.$$

Remark. For practical purposes it is convenient to take the minimal possible r with the constraint (4.9).

Proof of Proposition 4.2 is presented in Appendix B. Here we will demonstrate how Proposition 4.2 implies Theorem 4.1. Indeed, since Theorem 4.1 assumes the unlimited precision, we may set $\Delta \boldsymbol{e}_i = 0$, that is, $\varepsilon_i = 0$ for all i, that is, $\varepsilon = 0$. Therefore, the formula (4.9) becomes

$$2^{-r-1} \cdot \sqrt{n} < ½$$

or, equivalently,

$$\sqrt{n} < ½ \cdot (2^{r+1}) = 2^{r}.$$

Finally, we obtain $n < 2^{2r}$.

This proves Theorem 4.1. ■

Appendix A – *Performance of GCS in comparison with AES*

GCS has been implemented in Pentium IV computer system. The table below represents comparison of GCS performance with AES (Advanced Encryption Standard) performance, where GCS has parameters $m = n = l = 4$, N = 40, and $\boldsymbol{r} = 2$

Algorithm	**CPU**	**Language**	**Encoding, MB/sec**	**Decoding, MB/sec**
AES/ Rijndael * Key length: 128 bits	Pentium4 3.06GHz w/hyperthreading	X86 assembler	179.6	181.7
AES/ Rijndael* Key length: 128 bits	Athlon 2.25GHz	C	107.6	99.8
GCS (prototype implementation); Parameters: $m = n = 4$; Key length: $L = 512$ bits	Pentium4 2.4GHz	C++	>215	>210

* AES/ Rijndael - the fastest known software implementation
Data concerning AES performance are taken from: http://www.tcs.hut.fi/~helger/aes/rijndael.html

Appendix B - *Proof of Proposition 4.2.*

The proof of Propositon 4.2 is based on four technical statements regarding errors of computing certain transformations of $\mathbb{R}^n$. The first of them, Lemma B.1, defines the error in the case of linear transformations, and Lemma B.2 estimates this error in the case of the linear orthogonal transformations. The third one, Lemma B.3, provides error estimation in the case of composition of orthogonal transformations; Lemma B.4 provides error estimation when the transformation is a single reflection. The last result, Lemma B.5, specializes the results of Lemma B.3 and Lemma B.4 for the composition of reflections.

Lemma B.1. Assume that computation of a given vector $\boldsymbol{x}$ in $\mathbb{R}^n$ is performed with the error $\Delta\boldsymbol{x}$. Assume that f is a linear transformation of $\mathbb{R}^n$ such that its computational error is Δf: $\mathbb{R}^n \rightarrow \mathbb{R}^n$ (that is, for any precise vector $\boldsymbol{y}$ the value $f(\boldsymbol{y})$ is computed with the error $\Delta f(\boldsymbol{y})$). Then the total error $\tau\Delta f(\boldsymbol{x})$ of computing $f(\boldsymbol{x})$ is given by the formula:

(B.1) $$\tau\Delta f(\boldsymbol{x}) = f(\Delta\boldsymbol{x}) + \Delta f(\boldsymbol{x} + \Delta\boldsymbol{x}).$$

In particular, if g is another transformation of $\mathbb{R}^n$, then the error $\Delta(f\circ g)$ of the composition $f\circ g$ is given by the formula

(B.2) $$\Delta(f\circ g)(\boldsymbol{x}) = \Delta f(g(\boldsymbol{x})) + f(\Delta g(\boldsymbol{x})).$$

for each $\boldsymbol{x}$ in $\mathbb{R}^n$. Therefore, the total error of computing $f\circ g$ is given by

$$\tau\Delta(f\circ g)(\boldsymbol{x}) = f(g(\Delta\boldsymbol{x})) + \Delta f(g(\boldsymbol{x} + \Delta\boldsymbol{x})) + f(\Delta g(\boldsymbol{x} + \Delta\boldsymbol{x})).$$

Proof Lemma B.1. Let $\boldsymbol{x}' = \boldsymbol{x} + \Delta\boldsymbol{x}$ be the computed (i.e., approximate) value of $\boldsymbol{x}$. Furthermore, the computed (i.e., approximate) value of $f(\boldsymbol{y})$ for a given (i.e., precise, rater than approximate) vector $\boldsymbol{y}$ in $\mathbb{R}^n$ is:

$$f(\boldsymbol{y}) + \Delta f(\boldsymbol{y}).$$

Therefore, substituting $\boldsymbol{y}=\boldsymbol{x}'= \boldsymbol{x} +\Delta\boldsymbol{x}$, we see that the computed (i.e., approximate) value of $f(\boldsymbol{x}')$ is equal to

$$f(\boldsymbol{x}') + \Delta f(\boldsymbol{x}') = f(\boldsymbol{x} + \Delta\boldsymbol{x}) + \Delta f(\boldsymbol{x} + \Delta\boldsymbol{x}) = f(\boldsymbol{x}) + f(\Delta\boldsymbol{x}) + \Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})$$

because f is linear.

This finishes the proof of Lemma B.1. ■

Recall that, for each vector $\boldsymbol{x}$ in $\mathbb{R}^n$, $||\boldsymbol{x}||$ is the Euclidean norm of $\boldsymbol{x}$, that is,

$$||\boldsymbol{x}||^2 = (\boldsymbol{x}, \boldsymbol{x}) = x_1^2 + x_2^2 + \ldots + x_n^2.$$

Now assume that f is an orthogonal transformation of $\mathbb{R}^n$, that is, $||f(\boldsymbol{x})|| = ||\boldsymbol{x}||$ for every vector $\boldsymbol{x}$ (in particular, this guarantees that f is linear and $(f(\boldsymbol{x}), f(\boldsymbol{y})) = (\boldsymbol{x}, \boldsymbol{y})$ for any $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$).

Lemma B.2. Assume that computation of a given vector $\boldsymbol{x}$ in $\mathbb{R}^n$ is performed with the error $\Delta\boldsymbol{x}$. Assume that f is an orthogonal transformation of $\mathbb{R}^n$ such that its computational error is $\Delta f\colon \mathbb{R}^n \rightarrow \mathbb{R}^n$. Then the total error $\tau\Delta f(\boldsymbol{x})$ of computing $f(\boldsymbol{x})$ is estimated as follows:

(B.3) $$||\,\tau\Delta f(\boldsymbol{x})\,|| \leq ||\Delta\boldsymbol{x}|| + ||\Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})||$$

Proof of Lemma B.2. By Lemma B.1, the triangle inequality $||\boldsymbol{y} + \boldsymbol{z}|| \leq ||\boldsymbol{y}|| + ||\boldsymbol{z}||$, and the property $||f(\boldsymbol{y})|| =||\boldsymbol{y}||$ for any $\boldsymbol{y}$ in $\mathbb{R}^n$, we have

$$||\,\tau\Delta f(\boldsymbol{x})\,|| = ||f(\Delta\boldsymbol{x}) + \Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})|| \leq ||f(\Delta\boldsymbol{x})|| + ||\Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})|| = ||\Delta\boldsymbol{x}|| + ||\Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})||$$

Lemma B.2 is proved. ■

Lemma B.3. Let $f_1, f_2, \ldots, f_l$ be orthogonal transformations of $\mathbb{R}^n$ and let Δf_i be the error of computing f_i for each $\boldsymbol{i} = 1, 2, \ldots, l$ (see e.g. Lemma B.1). Assume that for each $\boldsymbol{i} = 1, 2, \ldots, l$ there exists a number $\varepsilon_i \geq 0$ such that

(B.4) $$||\,\Delta f_i(\boldsymbol{x})\,|| \leq \varepsilon_i \cdot ||\boldsymbol{x}||.$$

for any $\boldsymbol{x}$ in $\mathbb{R}^n$ and $i = 1, 2, \ldots, l$. Then for $f = f_1\circ f_2\circ\ldots\circ f_l$ we have

(B.5) $$||\Delta f(\boldsymbol{x})|| \leq \varepsilon \cdot ||\boldsymbol{x}||$$

for any $\boldsymbol{x}$ in $\mathbb{R}^n$, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + ... + \varepsilon_l$. Therefore, for $f = f_1 \circ f_2 \circ ... \circ f_l$ we have

(B.6) $$||\tau\Delta f(\boldsymbol{x})|| \leq \varepsilon \cdot ||\boldsymbol{x}|| + (\varepsilon + 1) \cdot ||\Delta\boldsymbol{x}||$$

for any $\boldsymbol{x}$ in $\mathbb{R}^n$.

Remark. In other words, for $\boldsymbol{x} \neq 0$, formula (B.5) reads

$$||\tau\Delta f(\boldsymbol{x})||/||\boldsymbol{x}|| \leq \varepsilon + (\varepsilon + 1) \cdot ||\Delta\boldsymbol{x}||/||\boldsymbol{x}||,$$

that is, the relative error $||\tau\Delta f(\boldsymbol{x})||/||\boldsymbol{x}||$ of computing $f(\boldsymbol{x})$ never exceeds ε plus the product of $(\varepsilon + 1)$ and the relative error of computing $\boldsymbol{x}$.

Proof of Lemma B.3. We prove (B.5) first. We employ Mathematical induction in l:

1. (Base of induction): If $l = 0$, i.e., f is the identity transformation of $\mathbb{R}^n$, the assertion is obvious because $||\Delta f(\boldsymbol{x})|| = 0$ and $\varepsilon = 0$.

2. (Inductive hypothesis): Assume now that (B.5) is proved for $l - 1$ or less, e.g., for $f' = f_1 \circ f_2 \circ ... \circ f_{l-1}$. Then we have (B.5) in the form

$$||\ \Delta f'(\boldsymbol{y})\ || \leq \varepsilon' \cdot ||\boldsymbol{y}||,$$

for any $\boldsymbol{y}$ in $\mathbb{R}^n$, where $\varepsilon' = \varepsilon_1 + \varepsilon_2 + ... + \varepsilon_{l-1} = \varepsilon - \varepsilon_l$.

3. (Inductive step) Let $f' = f_1 \circ f_2 \circ ... \circ f_{l-1}$ and $\varepsilon' = \varepsilon_1 + \varepsilon_2 + ... + \varepsilon_{l-1}$. Then $f = f' \circ f_l$ and $\varepsilon = \varepsilon' + \varepsilon_l$. Therefore, by the formula (B.2)

$$\Delta f(\boldsymbol{x}) = \Delta(f' \circ f_l)(\boldsymbol{x}) = \Delta f'(\boldsymbol{y}) + f'(\boldsymbol{z}),$$

where $\boldsymbol{y} = f_l(\boldsymbol{x})$ and $\boldsymbol{z} = \Delta f_l(\boldsymbol{x})$. Then using the inductive hypothesis for $\boldsymbol{y}$ we obtain:

$$||\ \Delta f(\boldsymbol{x})\ || = ||\Delta f'(\boldsymbol{y}) + f'(\boldsymbol{z})|| \leq ||\Delta f'(\boldsymbol{y})|| + ||f'(\boldsymbol{z})|| \leq \varepsilon' \cdot ||\boldsymbol{y}|| + ||\ f'(\boldsymbol{z})\ ||.$$

Since f', as a composition of orthogonal transformations of $\mathbb{R}^n$, is also an orthogonal transformation of $\mathbb{R}^n$, we have $||f'(\boldsymbol{z})|| = ||\boldsymbol{z}||$. And, by the assumption (B.4), $||\boldsymbol{z}|| = ||\Delta f_l(\boldsymbol{x})|| \leq \varepsilon_l \cdot ||\boldsymbol{x}||$. Since $f_l(\boldsymbol{x})$ is orthogonal, $||\boldsymbol{y}|| = ||f_l(\boldsymbol{x})|| = ||\boldsymbol{x}||$.

Combining the above computations, we finally obtain:

$$||\ \Delta f(\boldsymbol{x})\ || \leq \varepsilon' \cdot ||\boldsymbol{y}|| + ||\ f'(\boldsymbol{z})\ || \leq \varepsilon' \cdot ||\boldsymbol{x}|| + \varepsilon_l \cdot ||\boldsymbol{x}|| = (\varepsilon' + \varepsilon_l) \cdot ||\boldsymbol{x}|| = \varepsilon \cdot ||\boldsymbol{x}||,$$

which is the inequality (B.5).

This finishes the inductive step. Thus, the inequality (B.5) is proved.

In order to prove (B.6), we will use (B.3) and (B.5).

$$||\ \tau\Delta f(\boldsymbol{x})\ || \leq ||\Delta\boldsymbol{x}|| + ||\Delta f(\boldsymbol{x} + \Delta\boldsymbol{x})|| \leq ||\Delta\boldsymbol{x}|| + \varepsilon \cdot ||\boldsymbol{x} + \Delta\boldsymbol{x}|| \leq ||\Delta\boldsymbol{x}|| + \varepsilon \cdot (||\boldsymbol{x}|| + ||\Delta\boldsymbol{x}||) =$$
$$= \varepsilon \cdot ||\boldsymbol{x}|| + (\varepsilon + 1) \cdot ||\Delta\boldsymbol{x}||.$$

This proves (B.6). Lemma B.3 is proved. ■

We need the following fact, which estimates the error of computing a single reflection in $\mathbb{R}^n$.

Lemma B.4. Let $\boldsymbol{e}$ be a unit vector in $\mathbb{R}^n$ computed with the error $\Delta\boldsymbol{e}$ and let S be the reflection of $\mathbb{R}^n$ relative to the vector $\boldsymbol{e}$. Then the error ΔS of computing the reflection S is estimated by the formula:

$$||\Delta \mathrm{S}(\boldsymbol{x})|| \leq \varepsilon_{\boldsymbol{e}} \cdot ||\boldsymbol{x}||$$

for each $\boldsymbol{x}$ in $\mathbb{R}^n$, where $\varepsilon_{\boldsymbol{e}} = 4\cdot||\Delta\boldsymbol{e}|| + 2\cdot||\Delta\boldsymbol{e}||^2$.

<u>Proof of *Lemma B.4.*</u> Recall that since $\boldsymbol{e}$ is the unit length,

$$\mathrm{S}(\boldsymbol{x}) = \boldsymbol{x} - [2\cdot(\boldsymbol{x}, \boldsymbol{e})]\cdot\boldsymbol{e}$$

for any $\boldsymbol{x}$ in $\mathbb{R}^n$. We assume that the error ΔS of this S comes solely from the error $\Delta\boldsymbol{e}$ (of computing the vector $\boldsymbol{e}$). Then an approximate value of S($\boldsymbol{x}$) is

$$\boldsymbol{x} - [2\cdot(\boldsymbol{x}, \boldsymbol{e} + \Delta\boldsymbol{e})]\cdot(\boldsymbol{e} + \Delta\boldsymbol{e}).$$

Therefore,

$$\Delta\mathrm{S}(\boldsymbol{x}) = \{\boldsymbol{x} - [2\cdot(\boldsymbol{x}, \boldsymbol{e} + \Delta\boldsymbol{e})]\cdot(\boldsymbol{e} + \Delta\boldsymbol{e})\} - \{\boldsymbol{x} - [2\cdot(\boldsymbol{x}, \boldsymbol{e})]\cdot e\}$$

$$= [2\cdot(\boldsymbol{x}, \boldsymbol{e})]\cdot\boldsymbol{e} - [2\cdot(\boldsymbol{x}, \boldsymbol{e} + \Delta\boldsymbol{e})]\cdot(\boldsymbol{e} + \Delta\boldsymbol{e}) = -[2\cdot(\boldsymbol{x}, \boldsymbol{e})]\cdot\Delta\boldsymbol{e} - [2\cdot(\boldsymbol{x}, \Delta\boldsymbol{e})]\cdot\boldsymbol{e} - [2\cdot(\boldsymbol{x}, \Delta\boldsymbol{e})]\cdot\Delta\boldsymbol{e}\,.$$

(In particular, this formula implies that ΔS is a linear transformation of $\mathbb{R}^n$).This formula, in conjunction with the triangle inequality, implies that

$$||\Delta\mathrm{S}(\boldsymbol{x})|| \leq ||2\cdot(\boldsymbol{x}, \boldsymbol{e})\cdot\Delta\boldsymbol{e}|| + ||2\cdot(\boldsymbol{x}, \Delta\boldsymbol{e})\cdot\boldsymbol{e}|| + ||2\cdot(\boldsymbol{x}, \Delta\boldsymbol{e})\cdot\Delta\boldsymbol{e}|| =$$

$$= 2\cdot|(\boldsymbol{x}, \boldsymbol{e})|\cdot||\Delta\boldsymbol{e}|| + 2\cdot|(\boldsymbol{x}, \Delta\boldsymbol{e})|\cdot||\boldsymbol{e}|| + 2\cdot|(\boldsymbol{x}, \Delta\boldsymbol{e})|\cdot||\Delta\boldsymbol{e}||$$

Then using the fact $||\boldsymbol{e}||=1$ and Cauchy-Schwarz inequality $|(\boldsymbol{x}, \boldsymbol{y})| \leq ||\boldsymbol{x}||\cdot||\boldsymbol{y}||$ we obtain:

$$||\Delta\mathrm{S}(\boldsymbol{x})|| \leq 2\cdot||\boldsymbol{x}||\cdot||\Delta\boldsymbol{e}|| + 2\cdot||\boldsymbol{x}||\cdot||\Delta\boldsymbol{e}|| + 2\cdot||\boldsymbol{x}||\cdot||\Delta\boldsymbol{e}||\cdot||\Delta\boldsymbol{e}|| = (4\cdot||\Delta\boldsymbol{e}|| + 2\cdot||\Delta\boldsymbol{e}||^2)\cdot||\boldsymbol{x}|| = \varepsilon_{\boldsymbol{e}}\cdot||\boldsymbol{x}||$$

This proves <u>*Lemma B.4*</u>. ■

The following result explicitly estimates the precision of encryption/decryption transformations of GCS.

Lemma B.5. Let $\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}$ be the transformation given by the formula (4.2) with respect to the sequence $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$. Then for each vector $\boldsymbol{x}$ in $\mathbb{R}^n$ computed with the error $\Delta\boldsymbol{x}$, we have

(B.7) $$||\tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})|| \leq \varepsilon \cdot ||\boldsymbol{x}|| + (\varepsilon + 1) \cdot ||\Delta\boldsymbol{x}||,$$

where

$$\varepsilon = \varepsilon_{i_1} + \varepsilon_{i_2} + \ldots + \varepsilon_{i_l}$$

as in (4.6), and where

$$\varepsilon_i = 4\cdot||\Delta\boldsymbol{e}_i|| + 2\cdot||\Delta\boldsymbol{e}_i||^2,$$

for $i = 1, 2, \ldots, m$, as in (4.5).

Let $\boldsymbol{y} = \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})$, and assume that $\boldsymbol{y}$ is first computed with an error and then approximated (e.g., rounded) to the vector $\boldsymbol{y}' = \boldsymbol{y} + \Delta\boldsymbol{y} + \Delta$, where Δ is the error of approximation procedure. Then

(B.8) $$\|\tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| \le (\varepsilon + 1)\cdot\|\Delta\| + \varepsilon\cdot(\varepsilon + 2)\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)^2\cdot\|\Delta\boldsymbol{x}\|.$$

Proof of Lemma B.5. Prove (B.7) first. Using Lemma B.4 and Lemma B.3 in the situation when $f = \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})} = \mathrm{S}_{i_1}\circ\mathrm{S}_{i_2}\circ\ldots\circ\mathrm{S}_{i_l}$, we obtain from (B.6):

$$\|\tau\Delta f(\boldsymbol{x})\| \le \varepsilon\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)\cdot\|\Delta\boldsymbol{x}\|.$$

This proves the inequality (B.7).

Now we are going to prove the inequality (B.8). We will use the inequality (B.7) in the situation when we $\boldsymbol{y} = \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})$ is taken instead of $\boldsymbol{x}$ and $\Delta + \Delta\boldsymbol{y}$ instead of $\Delta\boldsymbol{x}$. Here $\Delta\boldsymbol{y}$ is the total error of computing $\boldsymbol{y}$, that is, $\Delta\boldsymbol{y} = \tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})$. Recall from (3.10) that

$$\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1} = (\mathrm{S}_{i_1}\circ\mathrm{S}_{i_2}\circ\ldots\circ\mathrm{S}_{i_l})^{-1} = \mathrm{S}_{i_l}{}^{-1}\circ\mathrm{S}_{i_{l-1}}{}^{-1}\circ\ldots\circ\mathrm{S}_{i_1}{}^{-1} = \mathrm{S}_{i_l}\circ\mathrm{S}_{i_{l-1}}\circ\ldots\circ\mathrm{S}_{i_1} = \mathrm{T}_{(\boldsymbol{e},\,\boldsymbol{i}')},$$

where $\boldsymbol{i}' = (i_l, i_{l-1}, \ldots, i_1)$. Then the total error of computing $\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})$ is estimated by the following modification of (B.7):

$$\|\tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| = \|\tau\Delta\mathrm{T}_{(\boldsymbol{e},\,\boldsymbol{i}')}(\boldsymbol{y})\| \le \varepsilon\cdot\|\boldsymbol{y}\| + (\varepsilon + 1)\cdot\|\Delta + \Delta\boldsymbol{y}\|.$$

Note that $\|\boldsymbol{y}\| = \|\boldsymbol{x}\|$, and $\|\Delta\boldsymbol{y}\| = \|\tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})\| \le \varepsilon\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)\cdot\|\Delta\boldsymbol{x}\|$ by (B.7). Therefore, using the triangle inequality $\|\Delta + \Delta\boldsymbol{y}\| \le \|\Delta\| + \|\Delta\boldsymbol{y}\|$, we obtain

$$\begin{aligned}\|\tau\Delta\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| &\le \varepsilon\cdot\|\boldsymbol{y}\| + (\varepsilon + 1)\cdot(\|\Delta\| + \|\Delta\boldsymbol{y}\|) \le \\ &\le \varepsilon\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)\cdot(\|\Delta\| + \varepsilon\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)\cdot\|\Delta\boldsymbol{x}\|) = \\ &= (\varepsilon + 1)\cdot\|\Delta\| + \varepsilon\cdot(\varepsilon + 2)\cdot\|\boldsymbol{x}\| + (\varepsilon + 1)^2\cdot\|\Delta\boldsymbol{x}\|.\end{aligned}$$

The inequality (B.8) is proved. Lemma B.5 is proved. ■

Now we are ready to finish the proof of Proposition 4.2. Let r be any number satisfying (4.9). We have to prove (4.1), that is,

$$\mathrm{Round}_{\mathbb{P}}(\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\mathrm{Round}_{\mathbb{C}}(\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(p)))) = p,$$

for any vector $p = [p_1, p_2, \ldots, p_n]$ satisfying (4.7), that is,

$$-2^{\mathrm{N}} < p_i < 2^{\mathrm{N}}$$

for $i = 1, 2, \ldots, n$. Indeed, using the formula

$$\|p\|^2 = (p, p) = p_1{}^2 + p_2{}^2 + \ldots + p_n{}^2,$$

we obtain:

$$\|p\|^2 \le (2^{\mathrm{N}} - 1)^2 + (2^{\mathrm{N}} - 1)^2 + \ldots + (2^{\mathrm{N}} - 1)^2 = (2^{\mathrm{N}} - 1)^2\cdot n.$$

Therefore,

(B.9) $$\|p\| \le (2^{\mathrm{N}} - 1)\cdot\sqrt{n}$$

Now, denote $\boldsymbol{y} = \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(p)$ and denote by $\Delta\boldsymbol{y}$ the error of computation of $\boldsymbol{y}$. Then

$$\mathrm{Round}_{\mathbb{C}}(\boldsymbol{y} + \Delta\boldsymbol{y}) = \boldsymbol{y} + \Delta\boldsymbol{y} + \Delta,$$

where Δ is the error of rounding. The entire setup fits exactly into Lemma B.5 with $\Delta\boldsymbol{x} = 0$. Therefore, formula (B.8) specializes to the following one:

(B.10) $$\|\tau\Delta \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| \le (\varepsilon + 1) \cdot \|\Delta\| + \varepsilon \cdot (\varepsilon + 2) \cdot \|\boldsymbol{x}\|.$$

One can easily show that the error Δ of rounding any vector of $\mathbb{R}^n$ to r binary places after the dot can be estimated as

(B.11) $$\|\Delta\| \le (2^{-r-1}) \cdot \sqrt{n},$$

because each coordinate Δ_i of Δ satisfies $|\Delta_i| \le ½ \cdot 2^{-r} = 2^{-r-1}$.

Substituting (B.9) and (B.11) in (B.10), we obtain:

$$\|\tau\Delta \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| \le (\varepsilon + 1) \cdot (2^{-r-1}) \cdot \sqrt{n} + \varepsilon \cdot (\varepsilon + 2) \cdot (2^{\mathrm{N}} - 1) \cdot \sqrt{n} =$$
$$= ((\varepsilon + 1) \cdot 2^{-r-1} + \varepsilon \cdot (2 + \varepsilon) \cdot (2^{\mathrm{N}} - 1)) \cdot \sqrt{n}.$$

Therefore, by (4.9),

$$\|\tau\Delta \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})\| < ½$$

We interpret this result as follows. First, denote for shortness $\Delta' = \tau\Delta \mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\boldsymbol{y})$. This is the total error of computing $\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}{}^{-1}(\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(\boldsymbol{x})) = \boldsymbol{x}$. In other words, the result of the computation is $\boldsymbol{x} + \Delta'$. The above inequality guarantees that $\|\Delta'\| < ½$. Therefore, each coordinate Δ'_i of Δ' satisfies $|\Delta'_i| < ½$. Since the vector $\boldsymbol{x} = p$ has integer coordinates, this implies that

$$\mathrm{Round}_{\mathbb{P}}(p + \Delta') = p,$$

which result is equivalent to (4.1). This proves Proposition 4.2. ■

Therefore, Theorem 4.1 is proved (see the argument right after Proposition 4.2) ■

Appendix C *GCS and Cryptanalysis*

In this section we shall consider some basic attacks against GCS.

1) Brute force attack aimed at reconstructing the key of GCS.
2) Length-preservation-based attack aimed at reconstructing individual plaintexts.
3) Algebro-geometric attack aimed at reconstructing the key of GCS.

The attacks 1) and 2) are aimed against GCS as a “real” (i.e., implemented metric) cryptosystem, as discussed in Section 4 and Appendix B. Attack 3) is an “ideal” attack against the normalized inner component. It is ideal in the sense that it ignores possible errors caused by rounding or by imprecision of real computational devices. Attack 3) aims at reconstructing normalized inner component for a given encryption transformation $\mathrm{T}_{(\boldsymbol{v},\, \boldsymbol{i})}$ (See Section 3). In this attack $\mathrm{T}_{(\boldsymbol{v},\, \boldsymbol{i})}$ is given as an $n\times n$-orthogonal matrix factorized as a product of l reflections. We will show the, in order to succeed, the cryptanalyst will have to face an algebraic variety of possibilities of dimension at least $n\cdot(n-1)/2$. In other words, it shows that in the discrete version of this attack the cryptanalyst may face a large number of algebraic equations with potentially infinite number of integer solutions.

Recall that in the definition of GCS we used only *repetition-free outer components* $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$, where the indices $i_1, i_2, \ldots, i_l$ form an l-element subset of the set $\{1, 2, ..., m\}$. In particular, $n \le l \le m$. The number of such outer components of the length l is

$$m \cdot (m-1) \cdot \ldots \cdot (m-l+1) = m!/(m-l)!.$$

C.1 A brute force attack aimed at reconstructing the key of GCS

The goal of the cryptanalyst is to reconstruct normalized inner component $\boldsymbol{e} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m)$. We proceed from the assumption that the cryptanalyst intercepted a number of messages in the form $(\boldsymbol{i}, c)$, where $\boldsymbol{i}$ is the outer component, $c = \mathrm{Round}_{\mathbb{C}}(\mathrm{T}_{(\boldsymbol{e},\boldsymbol{i})}(p)))$ is the corresponding ciphertext, and $\boldsymbol{e}$ is the (yet unknown to him) normalized inner component. His strategy is to choose a normalized inner component $\boldsymbol{e'} = (\boldsymbol{e'}_1, \boldsymbol{e'}_2, \ldots, \boldsymbol{e'}_m)$ in such a way that applying the decryption transformation $\mathrm{T}_{(\boldsymbol{e'},\boldsymbol{i})}{}^{-1}$ to the ciphertext c will result in a "readable" plaintext $p' = \mathrm{T}_{(\boldsymbol{e'},\boldsymbol{i})}{}^{-1}(c)$. This p' is an approximation of the actual text p, that is, $p' = p + \mathrm{D}$. The readability of p' will be for him a criterion for a correct choice of $\boldsymbol{e'}$. We say that $p' = p + \mathrm{D}$ is *not* readable if

$$\text{(C.1)} \qquad ||\mathrm{D}||^2 > ||p||.$$

If the opposite of (C.1) holds (i.e., if $||\mathrm{D}||^2 \le ||p||$), we say that p' is *possibly* readable.

Remark. We would argue that this definition of possible readability is too liberal. For instance, if p is a binary executable file, even an error in one bit of p will irreparably damage it as a computer program. The same is true when p a credit card number, then ***any*** error D is sufficient to damage it irreparably.

Based on this definition of non-readability, the cryptanalyst has to test various $\boldsymbol{e'}$ to guarantee that one of them will make p' possibly readable. In other words, he has no other options but to list all possible $\boldsymbol{e'}$ with some increment until he gets a possibly readable plaintext p'. At the moment when the cryptanalyst managed to recognize p' as an *actually* readable text, the cryptanalyst may decide that $\boldsymbol{e'}$ is sufficiently close to the original inner component $\boldsymbol{e}$. Therefore, he should proceed as follows.

Since each $\boldsymbol{e}_i$ has length 1, it depends only on $n-1$ of its coordinates. Therefore, in order for a given $\boldsymbol{i}$ to list all possible vectors $\boldsymbol{e'}_i$, one has to list only $n-1$ coordinates of the vector. Assume for the sake of simplicity that this listing is obtained by listing the coordinates of each of the vectors $\boldsymbol{e'}_1, \boldsymbol{e'}_2, \ldots, \boldsymbol{e'}_m$ independently from each other. That is, it proceeds as a listing of $[m \cdot (n-1)]$-tuples of real numbers. To simplify this work of the cryptanalyst we assume that each of these $m \cdot (n-1)$ numbers is bounded between $-1/\sqrt{(n-1)}$ and $1/\sqrt{(n-1)}$. This assumption also guarantees that each vector $\boldsymbol{e'}_i$ indeed has length 1. We also assume that the cryptanalyst lists each coordinate with the same increment d. This implies that in order to list all possible values of one coordinate he needs to perform at least $2/[d\cdot\sqrt{(n-1)}]$ operations. Therefore, to list all such $[m\cdot(n-1)]$-tuples requires performing at least

(C.2) $$(2/[d\cdot\sqrt{(n-1)}])^{m\cdot(n-1)}$$

operations.

The rest of our discussion of this attack is devoted to obtaining the *upper* bound of the increment d (see formula (C.7) below) and, therefore, to obtaining the *lower* bound for the number of operations needed for a successful attack (see formula (C.8) below).

Furthermore, we can think of the vectors $\boldsymbol{e'}_1, \boldsymbol{e'}_2, \ldots, \boldsymbol{e'}_m$ as approximation to $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m$ vectors respectively. We will write them as $\boldsymbol{e'}_1 = \boldsymbol{e}_1 + \Delta\boldsymbol{e}_1$, $\boldsymbol{e'}_2 = \boldsymbol{e}_2 + \Delta\boldsymbol{e}_2$, … , $\boldsymbol{e'}_m = \boldsymbol{e}_m + \Delta\boldsymbol{e}_m$ in the same way as in Appendix B. The cryptanalyst uses this approximate inner component $\boldsymbol{e'}$ in his construction of the decryption transformation $\mathrm{T}_{(\boldsymbol{e'},\boldsymbol{i})}^{-1}$. By the very definition, this is an approximate (i.e., not precise in any sense) decryption transformation. Applying the approximate decryption transformation $\mathrm{T}_{(\boldsymbol{e'},\boldsymbol{i})}^{-1}$ to the intercepted vector p, he obtains an approximately decrypted vector p', i.e. a vector of the form $p' = p + \mathrm{D}$, where D is the error of his approximate decryption. The goal of the cryptanalyst is to extract information from p'. At the moment when p' becomes readable, the cryptanalyst stops his work. He stops because there is a high probability that $\boldsymbol{e'}$ is close to the original inner component $\boldsymbol{e}$.

Let us apply the formula (B.7) to our situation. Namely, we take $\mathrm{T}_{(\boldsymbol{e},\,\boldsymbol{i})}^{-1}$ instead of $\mathrm{T}_{(\boldsymbol{e},\,\boldsymbol{i})}$ and then set $\boldsymbol{x} = p$, $\Delta\boldsymbol{x} = 0$, and $\tau\Delta\mathrm{T}_{(\boldsymbol{e},\,\boldsymbol{i})}^{-1}(\boldsymbol{x}) = \mathrm{D}$. Therefore (B.7) becomes

(C.3) $$\|\mathrm{D}\| \le \varepsilon \cdot \|\boldsymbol{x}\|,$$

where $\varepsilon = \varepsilon_{i_1} + \varepsilon_{i_2} + \ldots + \varepsilon_{i_l}$, as in (4.6), and where $\varepsilon_i = 4\cdot\|\Delta\boldsymbol{e}_i\| + 2\cdot\|\Delta\boldsymbol{e}_i\|^2$, for $i = 1, 2, \ldots, m$, as in (4.5).

Then combining (C.1) and (C.2) we obtain

$$\|p\| < \|\mathrm{D}\|^2 \le \varepsilon^2 \cdot \|p\|^2.$$

Hence,

(C.4) $$\varepsilon > 1/\sqrt{\|p\|}.$$

Informally speaking, (C.4) means that if p' is not readable, then $\boldsymbol{e'}$ is still not close enough to $\boldsymbol{e}$, that is, the *approximation error* ε is still big.

We assume, again for the sake of simplicity, that as long as the approximation error ε of $\boldsymbol{e}$ satisfies the inequality (C.4), the cryptanalyst cannot obtain any useful information from the approximately decrypted vector $p' = p + \mathrm{D}$.

Therefore, a successful attack would require inequality opposite to (C.4), i.e.,

(C.5) $$\varepsilon \leq 1/\sqrt{||p||}.$$

On the other hand, it is easy to see from the definition of the increment d and the approximation error ε that

(C.6) $$4 \cdot l \cdot d < \varepsilon.$$

This inequality holds because each coordinate of each $\Delta \boldsymbol{e}_i$ changes with the increment d during the attack, and, therefore $||\Delta \boldsymbol{e}_i|| \geq d$ for each i involved in the outer component $\boldsymbol{i}$. Combining (C.5) with (C.6), we can see that, for a successful attack, the increment d should be bounded from above as follows.

(C.7) $$d < 1/(4l \cdot \sqrt{||\boldsymbol{x}||}).$$

Therefore, combining the estimation (C.2) with the inequality (C.7), we obtain that the number of operations needed for a successful brute force attack is at least

(C.8) $$(8l \cdot \sqrt{||\boldsymbol{x}||}/[\sqrt{(n-1)}])^{m \cdot (n-1)}.$$

For example, if $m = l = n = 4$, and $||\boldsymbol{x}|| \approx 2^{40}$ (as it is currently implemented – see Appendix A above), the number of operations required for successful brute force attack is at least

$$(8 \cdot 4 \cdot 2^{20}/2)^{4 \cdot 3} = (2^4 \cdot 2^{20})^{4 \cdot 3} = (2^{24})^{12} = \mathbf{2^{288}}.$$

Which is a much better estimation then the estimation for AES 256. To put it into a perspective, let us imagine that one could build a machine that could recover a AES 256-bit key in a *day* (i.e., try 2^{256} keys per day), then it would take that machine approximately **eleven million (!!!)** years to crack GCS in this case.

Another example, when n is merely increased by 1, will yield a drastically different result. Namely, if $m = l = 4$, $n = 5$, and $||\boldsymbol{x}|| \approx 2^{40}$, the number of operations required for successful brute force attack is at least

$$(8 \cdot 4 \cdot 2^{20}/2)^{5 \cdot 4} = (2^4 \cdot 2^{20})^{5 \cdot 4} = (2^{24})^{20} = \mathbf{2^{480}} \approx 3 \cdot 10^{144}.$$

This number greatly exceeds the number of particles in the universe.

C.2 A length-preservation-based attack aimed at reconstructing individual plaintexts

This attack on GCS is based on the fact that

$$||f(\boldsymbol{x})|| = ||\boldsymbol{x}||$$

for any orthogonal transformation f and for any vector $\boldsymbol{x}$ in $\mathbb{R}^n$. In our case $\boldsymbol{x} = p$ is a plain text and $f = T_{(\boldsymbol{v},\boldsymbol{i})}$ The goal of the cryptanalyst is to use this fact in order to guess a plaintext p out of a given ciphertext c. The expectation is that the attack will take less time than the above brute force attack.

The following is our argument explaining why the attack doomed to fail.

Let p be a plaintext message and $c = f(p) + \Delta$ is the corresponding encrypted message, where Δ is the encryption error (see Section 4 and Appendix B). Denote $\Delta' = f^{-1}(\Delta)$. Note that

$$\text{(C.9)} \qquad ||\Delta'|| \leq (2^{-r-1}) \cdot\sqrt{n}$$

because $||\Delta'||^2 = ||\Delta||^2$ and because Δ satisfies (B.11).

We have

$$|| \, c \,||^2 = || \, f(p) + \Delta||^2 = || \, f(p + f^{-1}(\Delta))||^2 = \; || \, f(p + \Delta')||^2 = \; ||p + \Delta'||^2$$

Assume that $d = ||c||^2$. According to definition of the metric GCS in the beginning of Section 4, $2^{2r}\cdot d$ is an integer, i.e., as a binary number, d has at most $2r$ digits after the dot. If we write $p = [p_1, p_2, \ldots, p_n]$ and $\Delta' = [\Delta'_1, \Delta'_2, \ldots, \Delta'_n]$ then the cryptanalyst has to deal with the following equation:

$$d = (p_1+ \Delta'_1)^2 + (p_2+ \Delta'_2)^2 + \ldots + (p_n+ \Delta'_n)^2,$$

where d is a given binary rational number having at most $2r$ digits after the dot, Δ'_1, Δ'_2, …, Δ'_n are real numbers such that sum of their squares is less than $2^{-2r-2}\cdot n$, and $p_1, p_2, \ldots, p_n$ are the integers he is looking for. Note that the errors Δ'_1, Δ'_2, …, Δ'_n are unknown to the cryptanalyst. Even in the case that he knew all of these errors are zero (in which case d is an integer), he would have to solve the equation

$$d = (p_1)^2 + (p_2)^2 + \ldots + (p_n)^2,$$

that is, he would have to list all lattice points in the surface of the ball of radius $\sqrt{d}$ and having its center at the origin. This problem is known to be hard (see, for example, [9]).

The actual problem the cryptanalyst faces is significantly harder. He has to list all lattice points between two concentric balls $B(\sqrt{d} + \mu)$ and $B(\sqrt{d})$, where $\mu = (2^{-r-1})\cdot\sqrt{n}$ as in (C.9).

Since μ is a constant for our implementation (i.e., μ is not too close to 0) and d may be arbitrarily big, we expect that the number of lattice points in the difference $B(\sqrt{d} + \mu) - B(\sqrt{d})$ is close enough to the volume of this difference. Denote by $V_n(d, \mu)$ this volume. It is easy to see that

$$V_n(d, \mu) = A_n \cdot ((\sqrt{d} + \mu)^n - (\sqrt{d})^n) > A_n \cdot n \cdot (\sqrt{d})^{n-1} \cdot \mu,$$

where $A_n = 2^{n-k} \cdot \pi^k / [n \cdot (n-2) \cdot \ldots]$ and where k is the largest integer preceding $n/2$.

For $n \geq 4$ we expect that the number of lattice points the cryptanalyst has to list is at least $A_n \cdot n \cdot (\sqrt{d})^{n-1} \cdot \mu$. Since $\sqrt{d}$ is between 0 and $2^N \cdot \sqrt{n}$ and $\mu = (2^{-r-1}) \cdot \sqrt{n}$, the expected number of points the cryptanalyst has to list is commensurable with

$$A_n \cdot n \cdot (\sqrt{d})^{n-1} \cdot \mu \approx A_n \cdot n \cdot (½ \cdot 2^N \cdot \sqrt{n})^{n-1} \cdot (2^{-r-1}) \cdot \sqrt{n} = A_n \cdot n \cdot (\sqrt{n})^n \cdot 2^{(N-1) \cdot (n-1) - r-1}$$

This shows that $n \geq 4$ the attack cannot take significantly less time than the brute force listing of plaintexts.

For example, if N = 40 and $n = 4$, then (taking into account that $A_4 = ½ \cdot \pi^2$) the number of points is of an order magnitude

$$½ \cdot \pi^2 \cdot 4 \cdot 2^{(40-1) \cdot (4-1)} = \pi^2 \cdot 2^{118} \approx 3 \cdot 10^{36}.$$

This is rather a big number of points to list in order to reconstruct a single plaintext p. However, even after this huge listing is accomplished, it is not guaranteed that the attack is successful. For instance, if the plaintext p is a binary executable file, then the cryptanalyst faces the real problem of selecting this p out of $3 \cdot 10^{36}$ candidates. At the same time, even if completely successful, the attack of this kind would not help in reconstructing other plaintexts or the parameters (i.e., the normalized inner component) of GCS.

C.3 Algebro-geometric attack aimed at reconstructing the key of GCS

In this attack we assume that one transformation $T_{(\boldsymbol{v},\boldsymbol{i})}$ for a certain outer component $\boldsymbol{i} = (i_1, i_2, \ldots, i_l)$ became known to the cryptanalyst. (This knowledge of $T_{(\boldsymbol{v},\boldsymbol{i})}$ could have been obtained as a result of intercepting a number of ciphertexts for which he also somehow knows corresponding plaintexts.)

For the sake of simplifying the cryptanalyst's work we assume that $T_{(\boldsymbol{v},\boldsymbol{i})}$ is known to him precisely, i.e., without any rounding. We also assume that the inner component $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ consists of vectors with integer coordinates, i.e., each v_i belongs to $\mathbb{Z}^n$ (where $\mathbb{Z}$ is the set of all integers).

In other words we assumed that the cryptanalyst knows precisely the standard matrix A (with rational coefficients) of the linear encryption transformation $T_{(\boldsymbol{v},\boldsymbol{i})}$. His goal is to reconstruct the inner component $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ or, more precisely, the normalized inner component $\boldsymbol{e} = (e_1, e_2, \ldots, e_m)$, as in (4.2).

If successful in achieving his goal, the cryptanalyst also solves the following algebro-geometric problem (which we refer to as GCS problem).

GCS problem is: given an orthogonal $n\times n$ matrix A with rational coefficients, find all l-tuples of non-zero vectors $(u_1, u_2, \ldots, u_l)$ in $\mathbb{Z}^n$ such that

$$\text{(C.10)} \qquad S_{u_1} \circ S_{u_2} \circ \ldots \circ S_{u_l} = A,$$

where S_u stands for the reflection about the hyperplane orthogonal to the vector u, that is,

$$S_u(x) = \boldsymbol{x} - [2(\boldsymbol{x},u)/(u,u)]\cdot u.$$

Remark. In the above definition, each S_{u_j} is treated as a $n\times n$ matrix, so that the left hand side of (C.10) is merely a product of these l matrices.

Let us consider an example to illustrate complexity of GCS problem.

Take $l = 2$. Then $F(u, v) = S_u \circ S_v$. That is,

$F(u, v)(\boldsymbol{x}) = S_u(S_v(\boldsymbol{x})) = S_u(\boldsymbol{x} - [2(\boldsymbol{x}, v)/(v, v)]\cdot v) = S_u(\boldsymbol{x}) - [2(\boldsymbol{x}, v)/(v, v)]\cdot S_u(v) = \boldsymbol{x} - [2(\boldsymbol{x}, u)/(u, u)]\cdot u - [2(\boldsymbol{x}, v)/(v, v)]\cdot (v - [2(v, u)/(u, u)]\cdot u) = \boldsymbol{x} - [2(\boldsymbol{x}, u)/(u, u)]\cdot u - [2(\boldsymbol{x}, v)/(v, v)]\cdot v - [2(\boldsymbol{x}, v)/(v, v)]\cdot [2(v, u)/(u, u)]\cdot u$

That is,

$$F(u,v)(\boldsymbol{x}) = \boldsymbol{x} - [2(\boldsymbol{x}, u)/(u, u)]\cdot u - [2(\boldsymbol{x}, v)/(v, v)]\cdot v - [2(\boldsymbol{x}, v)/(v, v)]\cdot[2(v, u)/(u, u)]\cdot u$$

By definition, the matrix A of the transformation $F(u,v)$ is given by the following formula:

$$A_{i,j} = (F(u,v)(e_j), e_i),$$

for $i, j = 1, 2, \ldots, n$, where $e_1, e_2, \ldots, e_n$ comprize the standard orthonormal base in $\mathbb{R}^n$. Therefore,

$$A_{i,j} = \delta_{i,j} - 2\, u_j \cdot u_i /(u, u) - 2\, v_j \cdot v_i /(v, v) - 4\, v_j \cdot u_i \cdot (v, u) /[(v, v)\cdot(u, u)],$$

where $\delta_{i,j}$ is 0 if $i \neq j$, and 1 if $i = j$; and

$$(u, u) = u_1^2 + \ldots + u_n^2;\ (v, v) = v_1^2 + \ldots + v_n^2;\ (v, u) = v_1\cdot u_1 + \ldots + v_n\cdot u_n.$$

In the general case when $l > 2$, the problem is equivalent to solving $n \cdot (n - 1)/2$ equations, each of the degree $2l$, in $n \times l$ integer varriables.

In fact, the problem that the cryptanalyst faces is harder because, by the very nature of GCS, each matrix coefficient $A_{i,j}$ can be known to him only in a rounded form, that is, instead of a single rational number $A_{i,j}$, he has to deal with an interval of such numbers.

In its turn, the problem of solving polynomial equations in integers is closely related to the famous 10-th Hilbert problem (the solution to the original 10-th Hilbert's problem has been established as negative in [3]). More precisely, our problem is a particular case of the distributional Diophantine problem (See [5], Section 7). GCS problem seems to be harder than the average distributional Diophantine problem because our parameter A is not one-dimensional, and, therefore, can take more different values. We also expect that, for a generic parameter A, GCS problem may have very many solutions. This expectation is based on the fact that the continuous version of GCS problem indeed has too many solutions.

Continuous GCS problem is: given an orthogonal $n \times n$ matrix A with real coefficients, find all l-tuples of non-zero vectors $(u_1, u_2, \ldots., u_l)$ in $\mathbb{R}^n$ each of the unit length such that

$$\text{(C.11)} \qquad S_{u_1} \circ S_{u_2} \circ \ldots \circ S_{u_l} = A.$$

We will show below that. If $l \geq n$, the set of solutions is uncountable, more precisely, this set is a variety of the dimension at least $(l - n/2) \cdot (n - 1) \geq n \cdot (n - 1)/2$.

In order to prove this, we first reformulate the problem.

Let U_l be the set of all l-tuples of vectors $u_1, u_2, \ldots., u_l$ of $\mathbb{R}^n$ such that $(u_j, u_j) = 1$ for all j. That is, U_l is the Cartesian product of l copies of the unit $(n - 1)$-dimensional sphere. In particular, the dimension of U_l is $l \cdot (n - 1)$. Let $O_n(\mathbb{R})$ is the set of all orthogonal $n \times n$ matrices with real coefficients. Define the map $F: U_l \rightarrow O_n(\mathbb{R})$ as follows:

$$F(u_1, u_2, \ldots, u_l) = S_{u_1} \circ S_{u_2} \circ \ldots \circ S_{u_l}.$$

The problem is: to describe each fiber

$$F^{-1}(A) = \{(u_1, u_2, \ldots., u_l) \in X: F(u_1, u_2, \ldots., u_l) = A\}.$$

Here is our solution of the problem. By definition, each reflection S_u belongs to the orthogonal group $O_n(\mathbb{R})$. Moreover, it is well-known that $O_n(\mathbb{R})$ is generated by reflections. It is also well known that $O_n(\mathbb{R})$ is the disjoint union of $O_n^+(\mathbb{R})$ and $O_n^-(\mathbb{R})$, where the former one consists of all the orthogonal matrices of the determinant +1 and the latter one consists of the all the orthogonal matrices of the determinant –1.

Proposition C.1. Let A be an orthogonal matrix in $O_n(\mathbb{R})$ and a positive integer $l \geq n$ such that $\det(A) = (-1)^l$. Then there are uncountably many l-tuples of reflections $S_1, S_2, \ldots, S_l$ of $\mathbb{R}^n$ such that:

$$A = S_1 \circ S_2 \circ \cdots \circ S_l.$$

(Informally speaking, there are uncountably many ways to factorize A into a product of l reflections). More precisely, the variety of all such l-tuples is a manifold of dimension at least $n\cdot(n-1)/2$.

Proof. Indeed, the dimension of $O_n(\mathbb{R})$ as manifold is $n\cdot(n-1)/2$. Let U_l be the set of all l-tuples of vectors $u_1, u_2, \ldots, u_l$ of $\mathbb{R}^n$ such that $(u_j, u_j) = 1$ for all j. That is, U_l is the Cartesian product of l copies of the unit $(n-1)$-dimensional sphere. In particular, the dimension of U_l is $l\cdot(n-1)$. Define the map $F: U_l \rightarrow O_n(\mathbb{R})$ as follows:

$$F(u_1, u_2, \ldots, u_l) = S_{u_1} \circ S_{u_2} \circ \ldots \circ S_{u_l},$$

where S_u stands for the reflection about the hyperplane orthogonal to the vector u, that is,

$$S_u(x) = \boldsymbol{x} - [2(\boldsymbol{x},u)/(u,u)]\cdot u.$$

In fact, the image $F(U_l)$ belongs to $O_n^{\varepsilon}(\mathbb{R})$, where $\varepsilon = +$ if l is even, and $\varepsilon = -$ if l is odd. Note that for u and v such that $(u, u) = 1$ and $(v, v) = 1$ the reflection S_u equals the reflection S_v if and only if either $u = v$ or $u = -v$.

It is well known that each element A of $O_n(\mathbb{R})$ can be factored into at most n reflections because:

(a) Each A may be presented as a composition of some number $k \leq n/2$ of two-dimensional rotations and at most $n - 2k$ reflections.

(b) Each two-dimensional rotation is a composition of two reflections.

This implies that, if $l \geq n$, then F is surjective onto $O_n^{\varepsilon}(\mathbb{R})$ (that is, the image of F is the entire $O_n^{\varepsilon}(\mathbb{R})$). Moreover, each matrix A in $O_n^{\varepsilon}(\mathbb{R})$ can be obtained as an image of F in uncountably many ways which fact follows from the lemma below.

Lemma C.2. Assume that $l \geq n$ and A is an element of $O_n^{\varepsilon}(\mathbb{R})$. Then dimension of $F^{-1}(A)$ is at least

$$l\cdot(n-1) - n\cdot(n-1)/2 = (l - n/2)\cdot(n-1) \geq n\cdot(n-1)/2\ .$$

Proof. We have already proved that $F: U_l \rightarrow O_n^{\varepsilon}(\mathbb{R})$ is a surjective map (where $\varepsilon = +$ if l is even and $\varepsilon = -$ if l is odd). Furthermore, one can easily show that F is a smooth map, i.e., an infinitely differentiable map. It is well known that for a smooth surjective map of manifold $f: X \rightarrow Y$ the dimension of a fiber $f^{-1}(y_0)$ for each point y_0 in Y is at least the difference of the dimension of X and the dimension of Y. Thus, the fiber $F^{-1}(A)$ has dimension that is the difference of dimensions of the source and the target manifolds. This proves the lemma.

Lemma C.2 is proved. ■

This finishes the verification of the fact that each matrix A in $O_n^{\varepsilon}(\mathbb{R})$ can be factored into the product of l reflections (for any l such that $l \geq n$ and the parity of l is ε) in uncountably many ways.

This proves Proposition C.1. ■

C.4 GCS with a secret outer component

In what follows we shall use the notation of Section 4 and Appendix B. Consider a modification of GCS in which the encrypted messages are the triples ($\boldsymbol{i}$, c_0, c), where $\boldsymbol{i}$ is an outer component, c_0 is a ciphertext of the form

$$c_0 = \mathrm{Round}_{\mathbb{C}}\,(\mathrm{T}_{(\boldsymbol{v},\boldsymbol{i})}(\boldsymbol{j}\,)),$$

where $\boldsymbol{j}$ is another outer component considered as a vector of $\mathbb{R}^n$, and c is a ciphertext encrypted by means of $\boldsymbol{j}$, that is, $c = \mathrm{Round}_{\mathbb{C}}\,(\mathrm{T}_{(\boldsymbol{v},\boldsymbol{j})}(p))$, where p is a plaintext. The decryption of such a message proceeds as follows. First, $\boldsymbol{j}$ is decrypted as

$$\boldsymbol{j} = \mathrm{Round}_{\mathbb{P}}(\mathrm{T}_{(\boldsymbol{v},\,\boldsymbol{i}')}(c_0));$$

then, p is decrypted as $p = \mathrm{Round}_{\mathbb{P}}(\mathrm{T}_{(\boldsymbol{v},\,\boldsymbol{j}')}(c))$. (Here $\boldsymbol{j}'$ is the reversed outer component $\boldsymbol{j}$ and $\boldsymbol{i}'$ is the reversed outer component $\boldsymbol{i}$, as in Section 3).

The advantage of this modification is that the actual outer component $\boldsymbol{j}$ is unknown to the cryptanalyst which fact seriously complicates all above discussed attacks against GCS. Apparently, the only way to reconstruct the actual outer component $\boldsymbol{j}$ is to list all possible candidates for the component. This modification of GCS also deflects Chosen Ciphertext Attack (CCA).

References:

1. N. Koblitz, *P-adic numbers, p-adic analysis, and zeta-functions*. New York: Springer-Verlag, 1977.

2. J. E., Humphreys, *Reflection groups and Coxeter groups*, Cambridge; New York: Cambridge University Press, 1990.

3. Y. Matiyasevich*,* *Hilbert's Tenth Problem*. The MIT Press, Cambridge, London, 1993.

4. M. Ajtai, C. Dwork, "Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence", *Electronic Colloquium on Computational Complexity*, Report TR96-065, 1996.

5. Jie Wang**,** "Average-Case Intractable NP Problems": in *Advances in Languages, Algorithms, and Complexity* (D.-Z. Du and K.-I. Ko eds.), Kluwer Academic Publishers, pp. 313-378, 1997.

6. B. Jun and P. Kocher, "The Intel Random Number Generator," Cryptography Research, Inc. White Paper Prepared for Intel Corporation. April 22, 1999.

7. T. Moh, "A Public Key System With Signature And Master Key Functions," in *Communications in Algebra*, 27(5), pp 2207-2222 (1999).

8. G. Frey, "Applications of arithmetical geometry to cryptographic constructions", Proceedings of the Fifth International Conference on Finite Fields and Applications, Springer-Verlag, 2001, 128-161.

9. B. Hassibi and H. Vikalo, "The Expected Complexity of Sphere Decoding," in *Proc. 35th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Oct. 29-Nov. 1 2001.

10. A. Silverberg and K. Rubin, "Supersingular abelian varieties in cryptology," in *Advances in Cryptology* --- Crypto 2002, Lecture Notes in Computer Science 2442 (2002), Springer, 336-353