Invariant-based Cryptosystems and Their Security Against Provable Worst-Case Break*

Dima Grigoriev[1], Arist Kojevnikov[2], Sergey Nikolenko[2]

[1] IRMAR, Université de Rennes
Beaulieu, 35042, Rennes, France
http://perso.univ-rennes1.fr/dmitry.grigoryev/
[2] St.Petersburg Department of V. A. Steklov Institute of Mathematics
27 Fontanka, 191023 St.Petersburg, Russia
http://logic.pdmi.ras.ru/~arist/
http://logic.pdmi.ras.ru/~sergey/

Abstract. Cryptography based on noncommutative algebra still suffers from lack of schemes and lack of interest. In this work, we show new constructions of cryptosystems based on group invariants and suggest methods to make such cryptosystems secure in practice.
Cryptographers still cannot prove security in its cryptographic sense or even reduce it to some statement about regular complexity classes. In this paper we introduce a new notion of cryptographic security, a *provable break*, and prove that cryptosystems based on matrix group invariants and also a variation of the Anshel-Anshel-Goldfeld key agreement protocol for modular groups are secure against provable worst-case break unless NP $\subseteq$ RP.

1 Introduction

Suppose that, as usual, Alice and Bob are engaged in a cryptographic protocol, and Charlie tries to eavesdrop, decoding the messages that Bob sends to Alice. But now Charlie does not really trust the results he receives (or, perhaps, he has a boss who does not trust Charlie's algorithm of breaking the protocol), so he wants to be able to *prove* that his decoded message is actually what Bob had in mind. This is (informally) what we call a *provable break*.

In this setting, it is not sufficient for Charlie just to recover the encrypted message m from a ciphertext c, he should also justify that it is possible to encode m into c. Namely, in the provable break security model an adversary given a codeword $E(m)$ should not only produce the message m, but also present suitable random bits of E that might lead to such a cipher.

There may be several sets of random bits $\{r_1, \ldots, r_k\}$ that produce the same cipher: $E(m, pk, r_1) = \ldots = E(m, pk, r_k)$. In this case, of course, an adversary

* The research was done during the stay at the Max-Planck-Institut für Mathematik, Bonn, Germany. The second and the third authors were supported in part by INTAS (YSF fellowship 05-109-5565) and RFBR (grants 05-01-00932, 06-01-00502).

only needs to present *some* random string that results in the cipher, not necessarily the one Bob actually used (when $k > 1$, Charlie has absolutely no chance to find it anyway).

Informal discussion of provable break began in connection with the Rabin–Goldwasser–Micali cryptosystem based on quadratic residues [1]. It was shown that provable break of this cryptosystem implies that factoring is contained in RP. However, we know of no reference where a formal definition was presented.

One of the most fundamental questions in theoretical cryptography is to construct a secure encryption scheme based on some natural complexity assumption like P$\neq$NP. It is likely to be impossible (see [2] for recent results). Moreover, it is unknown if hard on average problems imply one-way functions [3]. Partial results were obtained under the assumption of a very strong adversary, *worst-case adversary*, who breaks the code *in all cases* [4, 5].

In this paper, we present two slightly different definitions of provable break (one weaker than the other) and prove that two different cryptographic protocols, namely the Anshel-Anshel-Goldfeld key agreement protocol and cryptosystems based on group invariants are all secure against provable worst-case break provided NP$\not\subseteq$RP. For the latter cryptosystem, we develop new ways to provide for their security in the usual cryptographic sense.

2 Definitions

First we define provable break of public key cryptosystems and then extend it to key agreement protocols. We present two separate definitions, one of them worst-case. The following definition is taken from [1].

Definition 1. *A* public-key encryption scheme S *consists of three probabilistic worst-case polynomial-time algorithms* (G, E, D) *for key generation, encryption and decryption respectively.*

The key generation algorithm G *on input* 1^n *(the security parameter) produces a pair* $G(1^n) = (e, d)$ *of public and secret keys. The encryption algorithm* E *takes as input a public key* e *and a plaintext message* m *and produces a ciphertext*

$$E(e, m) = c.$$

Finally, the decryption algorithm D *takes as input a secret key* d *and a ciphertext* c*. The output of* D *is a message*

$$D(d, c) = m',$$

which may fail to equal the original message m *when* $E(e, m) = E(e, m')$*. These situations are called* collisions*; we assume that collisions happen with negligible probability.*

Remark 1. In what follows we (equivalently) redefine the encryption algorithm E to be a deterministic worst-case polynomial-time algorithm with access to a random string r. It takes as input a public key e, a plaintext message m, and a random string r and produces ciphertext $E(r, e, m) = c$.

Definition 2. *An adversary C performs* provable break *of a cryptosystem (G, E, D) if for a uniform distribution over messages m and random bits of all participating algorithms (the public key pk is taken from the pair (pk, sk) generated by the key generation algorithm $G(1^n)$)*

$$\Pr\left[C(E(m, pk, r), pk) = (m, r')\right] \geq \frac{1}{poly(n)},$$

where $E(m, pk, r') = E(m, pk, r)$, and n is the security parameter.

If E is deterministic then provable break is equivalent to usual break. However, such cryptosystems are usually easy to break, because they allow an adversary to check that he has the right answer by re-encrypting the message. This is precisely the idea of provable break. An adversary should not only decipher the message, but also check that the cipher is actually a valid one; while the former may be trivial (as it will be in some of our examples), the latter may be very hard.

We also introduce the notion of a very strong adversary, that performs a worst-case break. The difference with the usual break is that the adversary should be successful on *all* inputs.

Definition 3. *An adversary C performs* provable worst-case break *of a cryptosystem (G, E, D) if for all messages m*

$$\Pr\left[C(E(m, pk, r), pk) = (m, r')\right] \geq \frac{1}{poly(n)},$$

where $E(m, pk, r') = E(m, pk, r)$, n is the security parameter, and the distribution is taken over random bits of all participating algorithms (public key pk is taken from the pair (pk, sk) generated by the key generation algorithm $G(1^n)$).

We say that a cryptosystem (G, E, D) is *secure against provable (worst-case) break* is there is no polynomial probabilistic Turing machine C performing provable (worst-case) break of (G, E, D).

Remark 2. It is easy to think of a trivial cryptosystem which is secure against provable break. Let Bob transfer the message openly (decryption is thus trivial), but add a value of some one-way function to the end of the message. Alice can disregard this one-way function, but Charlie would have to invert this one-way function in order to get a valid set of Bob's random bits. Therefore, our task is not to simply devise cryptosystems that are secure against provable break, but to devise them in such a way that they are or at least may be made secure in the usual cryptographic sense. Of course, we cannot *prove* their security (nobody currently can prove security of any cryptosystem at all), but we provide constructions that we believe to produce reasonably secure cryptosystems.

3 Invariant-based cryptosystems and their provable break

3.1 Cryptosystems based on group invariants

In [6], D. Grigoriev suggested a new class of public-key cryptosystems based on group invariants. In an invariant-based cryptosystem, Alice chooses a group $G \leq GL(n, F)$ acting on some vector space F^n. As a secret key, Alice chooses an invariant $f : F^n \to X$ such that $\forall g \in G\ f(gx) = f(x)$. She also selects a set (or a space given by generators) of messages $M \subseteq F^n$ such that for all $m_1 \neq m_2 \in M$ $f(m_1) \neq f(m_2)$. Thus, an invariant-based cryptosystem is defined by a triple (G, f, M). As the public key Alice transmits generators of G and M.

Bob selects a vector $m \in M$ (m is Bob's message) and a random element $g \in G$. After that, Bob communicates to Alice gm. Alice can decipher the message by taking the invariant $f(gm) = f(m)$.

It is now clear that the primary concern of the security of invariant-based cryptosystems is to find a well-concealed invariant. In what follows we give several ways to do so. These ways are similar to the ones employed in [7] and may be summarized with the following construction. Consider a tree such that each node of the tree contains a triple (G, f, M). Alice builds this tree from the leaves to the root, at each step keeping track of G, f, and M. After the tree is created, Alice takes the cryptosystem from the root and uses it.

An adversary will thus be able to break the cryptosystem if he knows the structure of the tree. This structure is equivalent to the description of the invariant from the security point of view, and may also be considered as Alice's secret key. The security of this cryptosystem will rely on the difficulty of the conjugacy and membership problems, as in [7].

3.2 An invariant-based cryptosystem secure against provable break unless NP⊆RP

The construction is based on the modular group. The *modular group* is the multiplicative group $SL_2(\mathbb{Z})$ of 2×2-matrices of determinant 1 (*unimodular matrices*).

In [8, Corollary 11.5] Blass and Gurevich proved that the following *bounded membership problem* (BM) for the modular group is NP-complete.

Problem 1. Let X be an unimodular matrix, S be a finite set of unimodular matrices and N be a positive integer. Can X be represented as $\prod_{i=1}^{m} Y_i$, where $m \leq N$ and for each i either Y_i or Y_i^{-1} is in S?

Remark 3. Do not confuse this problem with other problems that in [8] are proven to be RNP-complete. The primary difference is that in this case we are dealing with *group* membership, while RNP-complete problems arise from checking membership in *semigroups*.

Let us take G to be the unimodular group $\left(\begin{smallmatrix} 1 & * \\ 0 & 1 \end{smallmatrix}\right)$. As the invariant we take $f\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) = x_2$ and as the message space — the space of vectors $\left(\begin{smallmatrix} 1 \\ * \end{smallmatrix}\right)$. Bob takes a random element g in the given group (obtained by multiplying not more than, say, N generators), transports the message vector m into gm and transmits gm and N. Alice computes $f(gm)$ and decides which m it was.

Note that this "cryptosystem" is trivial to break: encryption does not change the part of the vector that actually carries the message. However, we will presently see that its provable break is NP-hard.

Theorem 1. *If there is a polynomial adversary C performing provable worst-case break of the invariant-based cryptosystem described above then* NP $\subseteq$ RP.

Proof. In short, the provable break is NP-hard because the Integer Sum problem is easily reduced to deciding bounded membership in a subgroup of the modular group, as shown in [8].

First, note that

$$\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \lambda + \mu \\ 0 & 1 \end{pmatrix}.$$

Thus, the problem of deciding bounded membership in a subgroup of the modular group is equivalent to the problem of deciding whether a given number is expressible as a bounded sum of other given numbers. This is the Integer Sum problem, shown to be NP-complete in [8].

If a polynomial-time algorithm solves a search problem with success probability $\frac{1}{n^{Const}}$, this probability can be easily amplified to 3/4 by repeating the algorithm for a polynomial number of times and taking the majority vote as an answer. Therefore, if a polynomially bounded adversary provably worst-case breaks the cryptosystem presented, NP $\subseteq$ RP.

In Sections 4 and 5 we present constructions aimed at making invariant-based cryptosystems more reasonable from the security viewpoint.

4 The tree of groups

The invariant-based protocol described in the previous section shares a discouraging property with the cryptosystem presented in a remark in Section 2. It is easy to break in the common cryptographic sense. In this section we provide a construction that allows us to "hide" these primitives inside a large tree of groups. We can also use it to improve security of the Anshel-Anshel-Goldfeld key agreement protocol.

We basically follow the lines of [7] (where one can find more details) to produce a tree of group–invariant–messages triples such that knowing the structure of the tree one can efficiently calculate the invariant in its root, while without knowing the structure the invariant is "concealed" in the tree.

In what follows, we concentrate on the invariant-based cryptosystems (introduced in Section 3) since [7] can be directly applied to key agreement protocols described in Section 7. However, we have to develop several new techniques to handle invariant-based cryptosystems. We will consider the same operations as in [7] and look at what happens with the invariants. But first we should introduce the basic notions.

To each vertex v of the tree a triple (G_v, f_v, M_v) is attached and triples are produced by recursion on the vertices of the tree starting with leaves towards the root to each vertex one of the following operations is assigned which allow the recursive step. For every vertex v the group G_v is a matrix group for some n and some underlying ring R.

To a tree we attach a resulting triple (G, f, M), where $G \leq GL(n, R)$ is a group, f is an invariant, that is, a function $f : R^n \to R$ such that $\forall g \in G$ $\forall x \in R^n$ $f(gx) = f(x)$, and $M \subset F^n$ is a canonical set of *messages* with the property that $\forall m \neq m' \in M$ $f(m) \neq f(m')$.

The public key will consist of R, n, G (given by generators), and M. The point of building such a tree is to conceal the secret invariant. Note that in situations where we change the invariant we can either change the invariant from f to $f \circ h$ or change the message space from M to $h(M)$. Since we care about concealing the invariant, and the message space will be given publicly, we will always choose the first alternative.

We want to combine this regular security with provable worst-case security of the modular group that we have proven in Theorem 1. To do this, we place a provably secure construction based on the modular group in one of the leaves of the tree. Then, for Charlie to solve the membership problem in the root of the tree, Charlie would have to solve the membership problem for all leaves of the tree (our construction has this property).

Let us now list the "building blocks" of the tree according to [9] and see what happens with the invariants in these cases.

1. *Changing the underlying ring* $\phi : R \to R'$. If the ring becomes smaller (R' embeds in R with $\varphi : R' \to R$, and $\phi\varphi = id$), an invariant f transforms into an invariant $\phi(f)$ that acts like $\phi(f)(x') = f(\varphi(x'))$. If $\forall x \in R^n, g \in G$ $f(x) = f(gx)$ then
$$\forall x' \in R'^n, g \in G \quad \phi(f)(gx') = f(g\varphi(x')) = f(\varphi(x')) = \phi(f)f(x')$$
If the ring becomes larger, bad things may happen (since there are new elements in the ring now, old equalities may not hold anymore). This property allows us to reason that any invariant known from invariant theory over fields will carry on to the rings that are subsets of these fields; e.g. any invariant over $\mathbb{C}$ will be an invariant over $\mathbb{Z}$.
However, this action requires care about the message space. If there were different representatives $m, m' \in M$ such that $\phi(m) = \phi(m')$ then the corresponding messages will be considered identical in the resulting message space $\phi(M)$. Therefore, it is sensible to reduce the underlying ring only if $\phi(M)$ is nontrivial.
2. *Conjugation* $g \mapsto h^{-1}gh$. The invariant $f(x)$ becomes the invariant $f'(x) = f(hx)$. If $\forall g \in G \forall x \in R^n$ $f(gx) = f(x)$ then
$$\forall g \in G \forall x \in R^n \quad f'(h^{-1}ghx) = f(hh^{-1}ghx) = f(g(hx)) = f(hx) = f'(x).$$

The message space M does not change.

3. *Direct product* $G_1, G_2 \mapsto G_1 \times G_2$. We here consider the natural representation of the direct product; if $G_1 \leq GL(n_1, F)$ and $G_2 \leq GL(n_2, F)$ then $G_1 \times G_2 \leq GL(n_1 + n_2, F)$, acting componentwise. In this situation, if $f_1(x)$, $f_2(x)$ were invariants of G_1, G_2, any element $f \in \langle f_1(x), f_2(y) \rangle \leq R[x, y]$ will be an invariant of $G_1 \times G_2$. We can choose a random element of this set, and the message space will in any case become $M_1 \times M_2$ (if we do not need that many different messages, we can choose several at random and discard the others).
4. *Wreath product* $G \wr H$, where $G \leq GL(n, R)$, $H \leq S_m$. In this case, we take the natural representation of $G \wr H$ on R^{mn} acting as
$$(g_1, \dots, g_m, \pi) \begin{pmatrix} x_1 \\ \dots \\ x_m \end{pmatrix} = \begin{pmatrix} g_1 x_{\pi(1)} \\ \dots \\ g_m x_{\pi(m)} \end{pmatrix}.$$
In this case, for any invariant f, if $\forall g \in G, x \in R^n$ $f(gx) = f(x)$ the same will hold for $G \wr H$ if we take f^m to act componentwise. The permutation disturbs nothing in the invariant equality. The message space will grow correspondingly to M^m (again, we may choose several messages at random or choose the diagonal $\Delta = \{(x, \dots, x) \mid x \in M\}$ if we do not need that many messages).

Apart from the old ways to extend the tree, invariant theory suggests new ways. The following will only work if f is a polynomial.

1. *Hessians* $H(f)$. If f is a polynomial invariant of G, and $\forall g \in G \leq GL(n, F)$ $\det g = \pm 1$ (note that F is a field) then
$$H(f) = \det \left(\frac{\partial^2 f}{\partial z_i \partial z_j} \right)$$
is also an invariant. The group G and the message space M remain unchanged.
2. *Jacobian* J. If $f_1, \dots, f_n$ are polynomial invariants of $G \leq SL(n, F)$ (note that F is a field) then
$$J(f_1, \dots, f_n) = \det \left(\frac{\partial f_i}{\partial z_j} \right)$$
is also an invariant. In this way we can unite n identical groups with different invariants into one; this will probably be useful only on the first level of the tree, where we can choose arbitrarily many identical groups.

5 The leaves of the tree

The previous section explains how to build a new invariant out of existing ones (thus, the recursive step). The question that remains is to find the base of this recursion. What should we put in the leaves of this tree?

5.1 General remarks

The first remark we should make is that in computer science, we cannot truly work over $\mathbb{C}$ or $\mathbb{R}$. Anything we do is actually over $\mathbb{Q}$. Invariant theory over $\mathbb{Q}$ is a little different from the classic well-known invariant theory over $\mathbb{C}$. Fortunately, we don't have to throw away the theory: if f is an invariant of a group $G \leq GL(n, \mathbb{C})$ represented by matrices with rational coefficients, then it is still an invariant of the group $G \leq GL(n, \mathbb{Q})$ because elements of G have rational coefficients. Therefore, in what follows we will refer to invariants over $\mathbb{C}$ but they will always be the same for $\mathbb{Q}$.

We may also look at invariants over finite fields, usually called *modular invariants*, but they provide a completely different story with completely different theory.

5.2 Orbit Chern classes

As an example of a standard well-known construction from invariant theory (see, e.g., [10]) we remind the so-called *orbit Chern classes*. They provide most known invariants of finite groups. The idea is simple: take an orbit a^G of an element $a \in F^n$ (suppose for the moment that G acts over a field) and note that $\prod_{b \in a^G}(x + b)$, where x is a formal variable, is invariant under G (elements of G only permute the factors in this expression). Its coefficients are called *orbit Chern classes*. For example, $\sum_{b \in a^G} b$ is an invariant.

All orbit Chern classes are nothing more than symmetric functions in the elements of the orbit; if we take a to be an unknown, we obtain the invariants we are looking for. The similar for compact groups.

5.3 Examples of finite groups' invariants

In this subsection we give several examples of invariants of different finite groups. The examples may be easily multiplied.

Example 1. The symmetric group S_n has a monomial representation on F^n $S_n \to GL(n, F)$ that permutes the variables. The ring of invariants in this case is generated by all symmetric polynomials, from $x_1 + \ldots + x_n$ to $x_1 \ldots x_n$. This is a simple example of orbit Chern classes.

Example 2. A cyclic group $\mathbb{Z}_n$ may be represented by any matrix $g \in GL(m, F)$ such that $g^n = e$ (a unipotent matrix of a matching order). For a function f to be an invariant of a cyclic group's representation, it suffices to ensure that it remains unchanged under the action of the only generator: $f(x) = f(gx)$.

For example, a cyclic group $\mathbb{Z}_n$ is naturally represented by a subgroup generated by $\xi_n e$, where ξ_n is a primitive n-th root of unity and e is the identity matrix. Obviously, any homogeneous polynomial of degree n is an invariant of this group. We can go one step further and consider the representation of a cyclic group $\mathbb{Z}_n$ generated by a matrix

$$\begin{pmatrix} \xi_1 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & \xi_m \end{pmatrix},$$

where ξ_i are (possibly different) primitive roots of unity, $\xi_i^n = 1$. The invariant ring of this group will be $\mathbb{C}[x_1^n, \dots, x_m^n]$.

Note that invariants depend not only on groups themselves, but also on their representations; the same group with different representations has very different invariants.

Example 3. A dihedral group D_{2k} has a representation $D_{2k} \to GL(2, \mathbb{R})$ as the symmetry group of a regular polygon. In this representation D_{2k} is generated by two matrices:

$$D_{2k} = \left\langle \begin{pmatrix} \cos\frac{2\pi}{k} & -\sin\frac{2\pi}{k} \\ \sin\frac{2\pi}{k} & \cos\frac{2\pi}{k} \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \right\rangle.$$

Then the invariant ring of the dihedral group in this representation is generated by polynomials

$$q = x^2 + y^2, \qquad h = \prod_{i=0}^{k-1} \left(\left(\cos\frac{2\pi i}{k} \right) x + \left(\sin\frac{2\pi i}{k} \right) y \right).$$

Example 4. For an odd prime p the dihedral group D_{2p} has a representation $D_{2p} \to GL(2, \mathbb{F}_p)$ over the finite field $\mathbb{F}_p$ given by the matrices

$$D_{2k} = \left\langle \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \right\rangle.$$

In this case the invariant ring is isomorphic to $\mathbb{F}_p[y, (xy^{p-1} - x^p)^2]$. However, if we switch to the dual representation (by simply transposing the matrices), the invariants will change substantially; the ring will now be isomorphic to $\mathbb{F}_p[x^2, y(y^{p-1} - x^{p-1})]$. In this example it was important that the group was represented over a finite field of degree not coprime with the group's degree.

These two examples show how much invariants depend on the actual representation. Some other examples of invariants of finite and classical groups one can find in [6].

5.4 Invariants of classical groups

In this subsection, we will give some examples of well-known invariants of classical groups. They may also lie in the leaves of the tree of groups.

Example 5. The orthogonal group in an even dimension $SO(2l, F)$ has the well-known *Dickson invariant*: if $\mathrm{char} F \neq 2$, which we will assume to be the case, it is $(-1)^{\det g}$ for a $g \in SO(2l, F)$. This invariant works for any field with characteristic not equal to two. Note that this invariant only has two values, so it is good for encrypting only one bit.

Example 6. The symplectic group $Sp(2n, F)$ by definition preserves a nondegenerate skew-symmetric bilinear form. The value of this form is an invariant (and, unlike the previous example, a polynomial invariant).

6 Attacks on invariant-based cryptosystems

When a new cryptosystem (or a family of cryptosystems) is presented, it is common to analyze the attacks on such cryptosystems. In this section we analyze several attacks on invariant-based cryptosystems and give practical advises on how to avoid their success.

6.1 Linear algebra attacks

The most dreaded attacks on algebraic cryptosystems usually go by linear algebra: an adversary constructs a system of linear equations and finds the secret key (the most notable example of this approach breaks the Polly Cracker scheme [11] that was only recently augmented with special techniques to make linear algebra attacks less efficient [12]).

Suppose that the invariant f is a polynomial of degree d. In this case, an adversary can view it as a polynomial with $\binom{n+d+1}{d}$ indefinite coefficients. To find the coefficients, he considers the equations $f(g_i m_j) = f(m_j)$ for all elements of the message space $m_j \in M$ and all generators $g_i \in G$. The space of solutions will yield an invariant separating the orbits of M (along with trivial invariants like $f = const$, of course). If d is a constant this attack will actually succeed, so Alice should choose invariants in such a way that $\binom{n+d+1}{d}$ is superpolynomial.

Example 7. Suppose that we are trying to build an invariant-based cryptosystem based on the monomial representation of the symmetric group S_n generated by transpositions τ_{ij} and its first degree invariant

$$f(x_1, \ldots, x_n) = x_1 + \ldots + x_n.$$

For the message space we should choose a number of vectors such that the sums of their coordinates are different; we denote them by $m_i = (m_{i1}, \ldots, m_{in})$. An adversary performing this kind of attack will simply consider a polynomial

$$h = \lambda_1 x_1 + \ldots + \lambda_n x_n$$

and solve a system of equations to ensure that transpositions do not change h. The equation corresponding to τ_{ij} is $h(\tau_{ij} x) = h(x)$ which is equivalent to

$\lambda_i = \lambda_j$. So, the adversary will arrive to the correct invariant (or a constant factor of it) after performing a polynomial algorithm. Note that in order to overcome this algorithm one should choose the message space in such a way that it contains messages with identical sums of elements. The adversary does not need to find *the same* invariant, he only needs to find *an* invariant that separates the vectors of M.

6.2 Monte-Carlo attack and orbit sizes

Another concern comes from the sizes of the orbits of elements of M. Indeed, suppose that an element $m \in M$ has an orbit m^G of polynomial size. In this case, an adversary has a polynomial chance of hitting the correct cipher $E(m)$ by simply picking an element $g \in G$ at random and comparing $E(m)$ and gm. Thus, the elements of the message space should be chosen with care to ensure that their orbits are large.

Example 8. For a trivial yet representative example consider a message space consisting of a zero vector and some other vector (the following analysis will do for any subgroup of $GL(n, F)$ and any invariant). The size of the zero vector orbit is 1, so an adversary does not have to do anything: if he sees a zero vector, the message was zero, if he sees a nonzero vector — it was the other vector that got “encrypted”.

6.3 Tree reconstruction attack

Finally, an adversary may attempt to reconstruct the tree with which the invariant was built. Along this way he will encounter, for example, of finding a matrix a such that $a^{-1}Ga = H$ for given G and H. This is a well-known hard problem; for example, in [13] it is shown that Graph Isomorphism reduces to the problem of group conjugation. This kind of attacks was considered in detail in [9]; the same reasoning applies in this case, since the task of reconstructing the tree has not become any easier. In fact, it has become harder, as the tree nodes are now augmented with invariants that may change nontrivially when going up the tree; consequently, to reconstruct a tree an adversary needs not only to reconstruct the groups but also to reconstruct invariants.

7 Anshel-Anshel-Goldfeld key agreement protocol secure against provable break

First we recall the definition of the Anshel-Anshel-Goldfeld key agreement protocol [14]. Let G be a group, and let two players A and B choose two subgroups of G

$$G_A = < a_1, \ldots, a_m >, \qquad G_B = < b_1, \ldots, b_n > .$$

Remark 4. Note that everything shown below goes without change if G_A and G_B are semigroups, not regular groups. All commutators are taken in the larger group G.

The group G and elements a_i, $1 \leq i \leq m$, and b_j, $1 \leq i \leq n$, are made public. Both players A and B randomly choose secret elements $a \in G_A$ and $b \in G_B$ as products of not more than N generators and transmit to each other the following sequences:

$$X_A = \{a^{-1}b_j a\}_{j=1}^{n}, \qquad X_B = \{b^{-1}a_i b\}_{i=1}^{m}.$$

After this transmission, player A (resp. B) has a representation of the element a (resp. b) in the subgroup G_A (resp. G_B). Therefore, he can compute a representation of the element $b^{-1}ab$ (resp. $a^{-1}ba$) using elements of the sequence X_A (resp. X_B). Thus, both players have shared a common key, namely the commutator

$$a^{-1}(b^{-1}ab) = [a, b] = (a^{-1}ba)^{-1}b.$$

An obvious necessary condition for this protocol to be secure is that the set of all commutators with $a \in G_A$ and $b \in G_B$ should contain at least two elements.

To provably break the Anshel-Anshel-Goldfeld key agreement protocol, one has to find representations of certain elements $a^{'}$ in G_A and $b^{'}$ in G_B, where

$$X_A = \{a^{'-1}b_j a^{'}\}_{j=1}^{n}, \qquad X_B = \{b^{'-1}a_i b^{'}\}_{i=1}^{m}.$$

Theorem 2. *The Anshel-Anshel-Goldfeld key agreement protocol for a modular group G and its subgroups G_A and G_B is secure against provable worst-case break unless NP $\subseteq$ RP. The same statement holds if G_A and G_B are considered as subsemigroups of G, rather than subgroups.*

Proof. Assume that there is a probabilistic polynomial-time Turing machine M such that for infinitely many security parameters N, and input $I = \{a_1, \ldots, a_m, b_1, \ldots, b_n, a^{-1}b_1a, \ldots, a^{-1}b_ma, b^{-1}a_1b, \ldots, b^{-1}a_nb\}$ it is true that

$$Pr[M(I) = a_1^{'}, s_1, \ldots, a_f^{'}, s_f, b_1^{'}, t_1, \ldots, b_g^{'}, t_g] \geq 1/p(N),$$

where $G_A =< a_1, \ldots, a_m >$ and $G_B =< b_1, \ldots, b_n >$ are subgroups of the modular group, $a \in G_A$, $b \in G_B$, $a^{'} = \prod_{i=1}^{f} a_i^{'s_i}$, $b^{'} = \prod_{j=1}^{g} b_j^{'t_j}$, $a_i^{'} \in \{a_i\}_{i=1}^{m}$, $b_j^{'} \in \{b_j\}_{j=1}^{n}$, $a^{'-1}b_ja^{'} = a^{-1}b_ja$, for all $1 \leq j \leq n$, $b^{'-1}a_ib^{'} = b^{-1}a_ib$, for all $1 \leq i \leq m$, s_i and t_j are in $\{-1, 1\}$ for all $1 \leq i \leq f$ and $1 \leq j \leq g$, $f, g \leq N$ and p is some polynomial. Note that we can check the correctness of the answer of M, so we also assume that M produces only correct answers.

Using M, we can a construct probabilistic polynomial-time Turing machine M' that contains $p(N)/2$ copies of M such that on input $(X, \{Y_i\}_i, N)$ it does the following.

1. If $X = \prod_{i=1}^{m} Y_i^{'s_i}$, where $Y' \in \{Y_i\}_i$, $m \leq N$, $s_i \in \{-1, 1\}$ (if we consider G_A and G_B as semigroups, here we take positive degrees only), then $Pr[M' \text{ accepts}] \geq 1/2$.
2. Otherwise, $Pr[M' \text{ accepts}] = 0$.

For inputs of all copies of M we take $a = b = X, a_i = b_i = Y_i$, and compute all $a^{-1}b_1a, \ldots, a^{-1}b_ma$, $b^{-1}a_1b, \ldots, b^{-1}a_nb$ in polynomial time. By [8, Corollary 11.5] the BM problem is NP-complete, hence, NP $\subseteq$ RP. $\square$

Remark 5. If G_A and G_B are semigroups, the BM problem is hard, moreover, on average [15].

Remark 6. Note that the described key agreement protocol can be insecure against linear algebra attack (cf. Subsection 6.1): it gives an adversary the decision of the conjugacy problem which could be unique, provided that the ring generated by G_A (or by G_B) coincides with the whole ring of matrices (that is the case if we build our protocol on the Blass-Gurevich groups). To make a cryptosystem more resistant against linear algebra attacks, one can replace G by a tree-like construction of groups or semigroups as in Section 4.

8 Conclusions and further work

In the paper, we have introduced a new notion of provable break and provable security in general. While this notion is undoubtedly much weaker than regular cryptographic security, it appears natural, well-defined, and sensible. Moreover, this notion of security is the only notion known to us for which provable positive statements are possible. We have provided three examples of cryptographic protocols: an invariant-based cryptosystem secure against provable break and a key agreement protocol secure against provable break. We are sure that one can produce more examples along the same lines.

Therefore, on one hand, further work lies in the search for more cryptographic primitives secure against provable break. On the other hand, one also wishes to look for connections between provable break and other notions of security. It is easy to think of a trivial cryptosystem for which provable security is equivalent to regular cryptographic security; however, it may be useful to look for nontrivial examples of the same. These lines will probably be similar to the research carried out by Ajtai and Dwork [16] who managed to reduce a worst-case problem to an average-case one and thus obtained a cryptosystem that is secure under some worst-case assumptions.

Acknowledgements. The authors are grateful to Edward A. Hirsch for valuable discussions.

References

1. Goldwasser, S., Bellare, M.: Lecture notes on cryptography. Summer course on cryptography at MIT (2001)
2. Bogdanov, A., Trevisan, L.: On worst-case to average-case reductions for np problems. In: FOCS'03. (2003) 308–317
3. Levin, L.A.: The tale of one-way functions. Problems of Information Transmission **39**(1) (2003) 92–103
4. Evan, S., Yacobi, Y.: Cryptography and np-completeness. In: ICALP'80. (1980) 195–207
5. Lempel, A.: Cryptography in transition. Computing Surveys **11**(4) (1979) 215–220

6. Grigoriev, D.: Public-key cryptography and invariant theory. J. Math. Sci. **126**(3) (2005) 1152–1157
7. Grigoriev, D., Ponomarenko, I.: Constructions in public-key cryptography over matrix groups. In Gerritzen, L., Goldfeld, D., Kreuzer, M., Gerhard, R., Shpilrain, V., eds.: Contemporary Mathematics: Algebraic Methods in Cryptography. Volume 418. AMS, Providence, RI (2007) 103–120
8. Blass, A., Gurevich, Y.: Matrix transformation is complete for the average case. SIAM J. Comput. **24**(1) (1995) 3–29
9. Grigoriev, D., Ponomarenko, I.: Homomorphic public-key cryptosystems and encrypting boolean circuits. Applicable Algebra in Engineering, Communication, and Computing **17** (2006) 239–255
10. Smith, L.: Polynomial Invariants of Finite Groups. Volume 6 of Research Notes in Mathematics. A. K. Peters, Wellesley, Massachusets (1996)
11. Fellows, M., Koblitz, N.: Combinatorial cryptosystems galore! finite fields: theory, applications, and algorithms. Contemp. Math. **168** (1992) 51–61
12. Ly, L.V.: Polly two: A new algebraic polynomial-based public-key scheme. Applicable Algebra in Engineering, Communication, and Computing **17** (2006) 267–283
13. Luks, E.M.: Permutation groups and polynomial-time computation. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Volume 11., (DIMACS, 1991), AMS (1993) 139–175
14. Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Mathematical Research Letters **6** (1999) 287–291
15. Venkatesan, R., Rajagopalan, S.: Average case intractability of matrix and diophantine problems. In: STOC'92. (1992) 632–642
16. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: The twenty-ninth annual ACM symposium on Theory of computing. (1997) 284–293