Algebraic cryptography: new constructions and their security against provable break*

Dima Grigoriev[1], A. Kojevnikov[2], Sergey I. Nikolenko[2]

[1] CNRS, Mathematiques, Universite de Lille,
59655, Villeneuve d'Ascq, France
Dmitry.Grigoryev@math.univ-lille1.fr
[2] Steklov Mathematical Institute,
Russia, 191023, St. Petersburg, nab. r. Fontanka, 27
http://logic.pdmi.ras.ru/~{arist,sergey}/

Abstract. Very few known cryptographic primitives are based on noncommutative algebra. Each new scheme is of substantial interest, because noncommutative constructions are secure agains many standard cryptographic attacks. On the other hand, cryptography does not provide security proofs that would allow to base the security of a cryptographic primitive on structural complexity assumptions. Thus, it is important to investigate weaker notions of security.
In this paper we introduce new constructions of cryptographic primitives based on group invariants and offer new ways to strengthen them for practical use. Besides, we introduce the notion of *provable break* which is a weaker version of the regular cryptographic break. In this version, an adversary should have a proof that he has correctly decyphered the message. We prove that cryptosystems based on matrix groups invariants and a version of the Anshel-Anshel-Goldfeld key agreement protocol for modular groups are secure against provable break unless NP = RP.

1 Algebraic cryptography

Public-key cryptography, since its very beginning [16, 53], has been actively employing algebraic constructions. The RSA protocol, for example, is based on number theory; the very construction of the protocol requires computing the Euler totient, $\varphi(n)$. Its security is based on factoring a number into prime divisors, or, more precisely, on the hardness of the so-called «RSA problem»: find roots of a given degree modulo a number $n = pq$, where p and q are prime (this task may not be equivalent to factoring; see [14, 15, 54] for more information).

However, usually the term *algebraic cryptography* is used in a narrower meaning. Algebraic cryptography deals with constructions where encoding and decoding are both group homomorphisms. In [29] Grigoriev and Ponomarenko give the following definition of a *homomorphic cryptosystem* (compare with Definition 2, where we introduce the general notion of a cryptosystem).

Definition 1 *Let H be a finite nonidentity group, G a finitely generated group, and $f : G \to H$ an epimorphism. Assume that R is a set of distinct representatives of the right cosets of*

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

$\mathrm{Ker}(f)$ *in* G*,* A *is a set of words in some alphabet, and a mapping* $P : A \to G$ *satisfies* $\mathrm{Im}(P) = \mathrm{Ker}(f)$*. A triple* $S = (R, A, P)$ *is called a* homomorphic cryptosystem *over* H *with respect to* f *if the following conditions are satisfied:*

- *one can generate random elements (of the sets* A*,* G*,* H*), compute the inverse of an element and the product of two elements (in the group* G *or* H*) in probabilistic polynomial (in* N*) time where* N *is the size of presentations of* G*,* H*, and* A*;*
- $|R| = |H|$*, and the image* $f(g)$ *of every element* $g \in R$ *as well as the unique preimage* $g \in R$ *such that* $f(g) = h$ *of every element* $h \in H$ *can be computed in probabilistic polynomial (in* N*) time;*
- *the mapping* P *is a trapdoor function.*

Formally speaking, even in this stricter sense algebraic cryptography was introduced almost at the same time as public key cryptography: the quadratic residue cryptosystem was the first homomorphic cryptosystem and one of the first known cryptosystems in general [22, 23].

However, algebraic cryptography in its modern sense stems from the works on elliptic curve cryptography [37, 45]. The basic constructions of elliptic curve primitives differ very little from the constructions based on discrete logarithm, such as the Diffie-Hellman key agreement protocol. The main difference is that elliptic curve crypto computes in an abelian group of points of an elliptic curve of the form $y^2 = x^3+ax+b$ or $y^2+xy = x^3+ax^2+b$; it allows to reduce the key size substantially and make the cryptosystems more efficient [10, 32, 58]. Lately the US government introduced a number of cryptographic standards based on elliptic curve cryptography.

Note that both classical constructions and elliptic curve crypto deal with *abelian* groups. This additional structure allows to make encoding and decoding algorithms even more efficient, while the analysis of arising computational tasks simplifies. However, over the last ten years abelian constructions were found to be susceptible to a new kind of cryptographic attacks, *quantum computing*. Beginning from the seminal works of Peter Shor [9, 55], factoring and discrete logarithm were solved in a simple and efficient manner on a quantum computer for abelian groups [24] (see also [42, 48] and Chapter 20 of [7]). In fact, a quantum computer is able to efficiently solve the problem of calculating the order of an element in an abelian group (via quantum Fourier transforms), and this problem generalized discrete logarithm.

Discrete logarithm and factoring were effectively the base of all classical crypto, including elliptic curve crypto. Thus, in a situation when quantum computing is already gradually becoming feasible (although the prospects are not clear yet), cryptography cannot limit itself to commutative constructions anymore. Cryptographers aim to create *nonabelian* constructions which are expected to be more secure [5].

That's why elliptic constructions and classical Diffie-Hellman and RSA constructions were generalized, and the task of constructing homomorphic cryptosystems was introduced in [19,60]. First steps in using algebraic constructions, in particular group theory, for building

cryptographic primitives were made in [8, 47, 49, 50]. An important step on the road of nonabelian cryptographic constructions were the works of Anshel, Anshel, and Goldfeld [2–4]. We will focus on one of their constructions in Section 9

These ideas were further developed in the works of Grigoriev and Ponomarenko. They developed basic definitions, such as Definition 1, and build constructions of nonabelian homomorphic public-key cryptosystems [29], homomorphic public-key cryptosystems over rings [28], and a general scheme of building complex homomorphic cryptosystems from smaller "blocks" [31] (this scheme will play an important part in this work). They also investigated the connections between homomorphic cryptosystems and encoding Boolean circuits [30].

In [25], Grigoriev suggested a method to use group invariants for public-key cryptography. In Section 5 we will discuss this construction in detail, but now we proceed to provable break — the second basic idea of the present work. This paper is a revised and extended version of [27].

2 Weak results in modern cryptography

Modern cryptography virtually does not allow one to prove the security of public key primitives. From the very first Diffie-Hellman key agreement protocol [16] and the RSA public key cryptosystem [53], numerous cryptographic constructions have been devised, but *not a single* proof of their security has appeared yet. Indeed, it would be very hard to find an unconditional proof since it would necessarily include the proof of P $\neq$ NP. However, conditional proofs of the kind «if P $\neq$ NP then a protocol is secure» are also hard to get. The matter here is that the classic notion of cryptographic security is naturally connected with average-case complexity rather than classical worst-case complexity.

Let us begin with a classic notion of *semantic security* of a cryptosystem; to define it we will have to begin with the definition of a cryptosystem [22].

Definition 2 *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.*

Definition 3 *An encryption scheme* (G, E, D) *is* semantically secure *if for all probabilistic polynomial time algorithms* M *and* A*, functions* h*, polynomials* Q *there exists a probabilistic polynomial time algorithm* B *such that for sufficiently large* k*,*

$$\Pr_r\left[A(1^k, c, e) = h(m) \mid (e,d) \leftarrow^r G(1^k), m \leftarrow^r M(1^k), c \leftarrow^r E(e,m)\right] \leq$$
$$\leq \Pr_r\left[B(1^k) = h(m) \mid m \rightarrow^r M(1^k)\right] + \frac{1}{Q(k)}.$$

Informally speaking, every adversary who knows the distribution of messages M and receives as input the encoded message and the public key will not be able to decode it substantially more often than an algorithm that does not know anything except M (M is necessary because if, for example, only one message were transmitted all the time then an adversary would indeed be able to decode it, not even needing the code and the public key). The main problem here is that the probabilities in this definition are taken over the distribution on the inputs of the cryptosystem, over the messages; this is, of course, natural because in practice it is usually enough to break a cryptosystem on a substantial fraction of the inputs, not necessarily on all of them.

Recent results allowed to connect cryptographic security (in the sense of Definition 3) with worst-case complexity assumptions for certain problems [1, 17, 51, 52]. However, these assumptions so far appear artificial and, as before, do not translate into assumptions about (in)equality of some complexity classes. In general, it seems that modern cryptography still has a very long way to go before any *provably* secure constructions. There even exists evidence supporting the view that such constructions may not exist or may lead to improbable corollaries for complexity classes [13].

Thus, is it natural that researchers, having encountered a problem too hard, began work on alternative definitions, criteria, and contexts trying to prove security of some constructions in some definitions.

The first natural approach would be to consider not the "real" cryptographic primitives but their weaker counterparts, like *one-way functions* (note that a one-way function does not imply a public-key cryptosystem, one would need a trapdoor function for it). Indeed, results on one-way functions are easier to prove. For example, complete one-way functions (by a *complete* one-way function we mean such a function f that if one-way functions exist then f is a one-way function) have been known for quite a long time already [21, 40], while complete public-key cryptosystems appeared only recently in the works of Harnik et al. and Grigoriev et al. [26,33]. Moreover, recently Levin developed more natural combinatorial constructions of complete one-way functions [41]; this work was continued by recent results by the Kojevnikov and Nikolenko [38].

Another approach that also leads to weaker results is to change the notion of security itself. If we consider weaker notions of security we are able to construct provably secure primitives. For example, consider the theory of *feebly one-way functions*, developen by A. Hiltgen [34,35]. Hiltgen considered the circuit complexity of functions in the complete binary

basis which is the hardest case for proving lower bounds [12,59]. He managed to devise a linear invertible function such that computing this function is almost twice easier than inverting it. These bounds, however weak, were the first unconditional security proofs for one-way functions. Lately, Hirsch and Nikolenko have created a feebly secure trapdoor function [36]; it appears possible to devise a provably secure cryptosystem along the same lines (security understood in an extremely restricted sense, of course).

On the other hand, partial results were obtained under the assumption of a very strong adversary, a *worst-case adversary* who breaks the code *in all cases*. No wonder that in this setting one could base security on worst-case complexity assumptions [18,39]. For a detailed survey on the subject we refer the reader to the book [46] and to previous papers of the first author [30,31].

In the next section we introduce and investigate another approach to weakening the notion of security. It turns out that our definition of *provable break*, while being somewhat artificial, allows to associate provable break of invariant-based cryptosystems and Anshel-Anshel-Goldfeld key agreement protocols with worst-case structural complexity assumptions.

3 Provable break

Consider a system with three participants: Alice, Bob, and Charlie. Suppose that, as usual, Alice (A) and Bob (B) are engaged in a cryptographic protocol (in a key agreement protocol Alice and Bob are equal peers, and in a public key cryptosystem Alice forms a pair of public and secret keys and emits the public key, while Bob encodes his message and sends it to Alice over an open channel), and Charlie tries to eavesdrop, decoding the messages that Bob sends to Alice. But now Charlie's task is different: not only does he need to decipher Bob's message, he also he wants to be able to *prove* that his decoded message is actually what Bob had in mind. Perhaps, Charlie does not really trust the results he receives; perhaps, he has a boss who does not trust Charlie's algorithm of breaking the protocol. 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. What could serve as such a justification? We take the following natural idea as definition: 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.

Remark 1. In what follows we (equivalently) redefine the encoding algorithm E as a deterministic worst-case polynomial time algorithm with access to a random string r. It receives as input the public key e, the message m, and a random bit string r and outputs the encoded message $E(r, e, m) = c$.

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 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 [22]. 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 and studied.

As we have already mentioned in Sec. 2, one of the most fundamental unsolved questions in theoretical cryptography is to construct a secure encryption scheme based on some natural complexity assumptions like P≠NP. 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⊈RP. For the latter cryptosystem, we develop new ways to provide for their security in the usual cryptographic sense.

4 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 for the average case, and one for the worst case. First, we recollect Definition 2 ad define the corresponding provable break.

Definition 4 *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.

The security parameter in Definition 4 is effectively the key length; the security parameter should be chosen such that the known adversary algorithms would run long enough to make break hopeless.

If E is deterministic then provable break is equivalent to usual break (the set of random bits is empty). An adversary can prove that he has correctly decoded a message by encoding it again. 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 breaks the cryptosystem in the worst case. The difference with the usual break is that the adversary should be successful on *all* inputs.

Definition 5 *An adversary C performs* provable worst-case break *of a cryptosystem (G, E, D) if for all messages m, all pairs of keys (pk, sk) generated by $G(1^n)$, and all random bits of the encoding algorithm E*

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

where $E(m, pk, r') = E(m, pk, r)$, n is the security parameter, and the distribution is taken over the random bits of the adversary C.

We say that a cryptosystem (G, E, D) is *secure against provable (worst-case) break* if 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 (in the sense of Definition 4 which automatically makes it secure against worst case provable break in the sense of Definition 5). 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 may 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, but we provide constructions that we believe to produce reasonably secure cryptosystems.

5 Invariant-based cryptosystems and their provable break

5.1 Cryptosystems based on group invariants

In [25], 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 a set X and 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 gm to Alice. Alice can decipher the message by taking the invariant $f(gm) = f(m)$. Since she had chosen the set of messages M as a transversal set for the orbits, the value of $f(m)$ allows her to uniquely determine the original message m. We call a triple (G, f, M) *admissible* if it correctly defines an invariant–based cryptosystem.

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 [31] 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 (we suppose that cryptosystems in the leaves are easy to break, otherwise there is no point to construct a 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 [31] (see Section 6 for details).

5.2 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*). Algebraic properties of this group are described in detail in, for example, [6].

In [11, 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 [11] are proven to be DistNP-complete (complete with respect to average case reductions). 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

$$G = \left\{ \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, x \in \mathbb{Z} \right\}.$$

As the invariant we take a rather trivial map

$$f \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_2,$$

and as the message space — the space of vectors

$$M = \left\{ \begin{pmatrix} 1 \\ x \end{pmatrix}, x \in \mathbb{Z} \right\}.$$

Bob selects 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 [11].

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 [11].

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 then NP $\subseteq$ RP.

In Sections 6 and 7 we present constructions aimed at making invariant-based cryptosystems more reasonable from the security viewpoint.

6 The tree of groups

6.1 General remarks

The invariant-based protocol described in the previous section shares a discouraging property with the cryptosystem presented in a remark in Section 4. 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 follow the lines of [31] 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" behind computationally hard problems in the tree.

In what follows, we concentrate on the invariant-based cryptosystems (introduced in Section 5) since [31] can be directly applied to key agreement protocols described in Section 9. However, we have to develop several new techniques to handle invariant-based cryptosystems. We will consider the same operations as in [31] and look at what happens with the invariants. But first let us introduce some basic notions.

To each vertex v of the tree we attach a triple (G_v, f_v, M_v). Triples are produced by recursion on the vertices of the tree starting with leaves towards the root to each vertex. At each step, we apply one of the operations described below. For every vertex v the group G_v is a matrix group $G_v \leq \mathrm{GL}(n, R)$ for some n and some base ring R.

Thus, a tree corresponds to the 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 consists of R, n, G (given by matrix generators), and M. The point of building such a tree is to conceal the secret invariant.

Remark 4. 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 is to 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).

6.2 Base of recursion

To treat the construction formally, consider a class of groups $\mathcal{G}$ closed under a certain set of group–theoretical operations $\mathcal{O}$ (we list the relevant operations below) on triples (G, f, M) that preserve admissibility. For a set $\mathcal{G}_0 \subset \mathcal{G}$ (which is the base of the construction) we define recursively a class $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ of quadruples (G, f, M, T) in the following way.

- *Base of recursion*: any quadruple (G, f, M, T), where $G \in \mathcal{G}_0$, (G, f, M) is an admissible triple, and T is a single node labeled by G.
- *Recursive step*: given quadruples $\{(G_i, f_i, M_i, T_i)\}_{i=1}^{s}$ and an operation $o \in \mathcal{O}$ of arity s, the class $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ contains the quadruple (G, f, M, T), where $G = o(G_1, \ldots, G_s)$, $f = o(f_1, \ldots, f_s)$, $M = o(M_1, \ldots, M_s)$, and T is the tree obtained from $T_1, \ldots, T_s$ by adding a new root labeled by o, its sons being the roots of $T_1, \ldots, T_s$.

6.3 Recursive step

Let us now list the "building blocks" of the tree, the operations acting on admissible triples. A number of such operations were introduced in [30]; we have to check what happens with the invariants in these cases.

1. *Changing the base 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

$$(g_1, \ldots, g_m, \pi) \begin{pmatrix} x_1 \\ \ldots \\ x_m \end{pmatrix} = \begin{pmatrix} g_1 x_{\pi(1)} \\ \ldots \\ g_m x_{\pi(m)} \end{pmatrix}.$$

   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 $\Delta = \{(x, \ldots, x) \mid x \in M\}$ if we do not need that many messages).

Apart from the previously considered ways to extend the tree, invariant theory suggests new ways. We can consider several transformations $o \in \mathcal{O}$ that leave the group intact ($o(G) = G$) and only change the invariant f and the message space M. 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 $f_1, \ldots, f_n$ are polynomial invariants of $G \leq SL(n, F)$ (note that F is a field) then
$$J(f_1, \ldots, f_n) = \det\left(\frac{\partial f_i}{\partial z_j}\right)$$
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.

7 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?

7.1 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 (see Example 4 in Section 7.3).

7.2 Orbit Chern classes

As an example of a standard well-known construction from invariant theory (see, e.g., [56]) 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 of G, namely the first orbit Chern class.

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. Similar statements hold for compact groups.

7.3 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

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

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

Note that invariants depend not only on groups themselves, but also on their representations; the same group with different representations can have 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 [25].

7.4 Invariants of classical groups

In this subsection, we give two 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 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).

8 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.

8.1 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 [20] that was only recently augmented with special techniques to make linear algebra attacks less efficient [44]).

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.

8.2 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".

8.3 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 [43] it is shown that Graph Isomorphism reduces to the problem of group conjugation. This kind of attacks was considered in detail in [30]; 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.

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

First we recall the construction of the Anshel-Anshel-Goldfeld key agreement protocol [3]. Let G be a group, and let two players A and B choose two subgroups of G

$$G_A = \langle a_1, \ldots, a_m \rangle, \qquad G_B = \langle b_1, \ldots, b_n \rangle.$$

REMARK 5. Note that everything shown below goes without change if instead of G_A and G_B we consider subsemigroups of G $\tilde{G}_A$ and $\tilde{G}_B$ generated by the same elements $\langle a_1, \ldots, a_m \rangle$ and $\langle b_1, \ldots, b_n \rangle$, respectively, but generated as semigroups rather than 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 instead of G_A and G_B we consider subsemigroups of G $\tilde{G}_A$ and $\tilde{G}_B$ generated by the same elements $\langle a_1, \ldots, a_m \rangle$ and $\langle b_1, \ldots, b_n \rangle$, respectively, but generated as semigroups rather than groups.*

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 $G_A = \langle a_1, \ldots, a_m \rangle$ and $G_B = \langle b_1, \ldots, b_n \rangle$ are subgroups of the modular group, $a \in G_A$, $b \in G_B$, $a' = \prod_{i=1}^{f} a_i^{'s_i}$, $b' = \prod_{j=1}^{g} b_j^{'t_j}$, $a'_i \in \{a_i\}_{i=1}^{m}$, $b'_j \in \{b_j\}_{j=1}^{n}$, $a^{'-1} b_j a' = a^{-1} b_j a$, for all $1 \le j \le n$, $b^{'-1} a_i b' = b^{-1} a_i b$, for all $1 \le i \le m$, s_i and t_j are in $\{-1, 1\}$ for all $1 \le i \le f$ and $1 \le j \le g$, $f, g \le N$ 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 $Y' \in \{Y_i\}_i$, $m \le N$, $s_i \in \{-1, 1\}$ (if we consider G_A and G_B as semigroups, here we take positive degrees only), then $Pr[M' \text{ accepts}] \ge 1/2$.
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 $a^{-1} b_1 a, \ldots, a^{-1} b_m a$, $b^{-1} a_1 b, \ldots, b^{-1} a_n b$ in polynomial time. By [11, Corollary 11.5] the BM problem is NP-complete, hence, NP $\subseteq$ RP. □

REMARK 6. If G_A and G_B are semigroups, the BM problem is hard, moreover, on average [57].

Note that the described key agreement protocol can be insecure against linear algebra attack (cf. Subsection 8.1): 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 6.

Formally speaking, we produce the following recursive construction for a class of groups $\mathcal{G}$ closed under a certain set of group–theoretical operations $\mathcal{O}$; this time we do not have to worry about admissible triples, and the operations are defined simply on groups of $\mathcal{G}$. For a set $\mathcal{G}_0 \subset \mathcal{G}$ (which is the base of the construction) we define recursively a class $\mathcal{P}(\mathcal{G}_0, \mathcal{O})$ of pairs (G, T).

The recursive definition is done precisely as in 6.2, omitting the constructions of the invariants and message spaces. In our case, the set $\mathcal{O}$ of admissible operations consists of changing the underlying ring, direct products, wreath products, and conjugations (same as for invariant–based cryptosystems, but without invariant–specific operations).

The security of the Anshel–Anshel–Goldfeld key agreement protocol for matrix groups relies on the following problem.

Linear Transporter Problem (LTP). Let R be a commutative ring, V be an R–module and $G \le \mathrm{GL}(V, R)$. Given $u \in V$ and $v \in u^G = \{u^g : g \in G\}$ find $g \in G$ such that $v = u^g$.

If an adversary can efficiently solve LTP, he can obviously break the Anshel–Anshel–Goldfeld protocol. In [31], the following proposition was proven (Lemma 3.4).

Proposition 1 *Let $G \in \mathcal{G}$. Then, given a derivation tree of G, LTP for G can be solved in time polynomial in the size of the tree and the times of solving LTP for leaves of the tree.*

Of course, this does not prove that security of the Anshel–Anshel–Goldfeld key agreement protocol in the root of the tree depends on the security of this protocol in the leaves of the tree. We have a much weaker statement that goes in the undesirable direction *twice*: if we can solve LTP we can break the Anshel–Anshel–Goldfeld protocol, and if we can solve LTP for leaves of the tree, we can solve LTP for its root. To prove security we would need to reverse both statements. However, this is the best we can do, and we know of no similar constructions with stronger dependencies.

10 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 on of the few known notions for which provable positive statements are possible. We have provided three examples of cryptographic protocols: an invariant-based cryptosystem secure against provable break (we have also substantially advanced the theory of invariant-based cryptosystems since [25]) and a key agreement protocol secure against provable break, a special case of the Anshel-Anshel-Goldfeld key agreement protocol. 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 (for example, Bob may not use random bits at all); 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 [1] later augmented by Regev [51, 52]. They managed to reduce a worst-case problem to an average-case one and thus produced a cryptosystem that is secure under some worst-case assumptions.

Acknowledgements.The authors are grateful to Edward A. Hirsch for valuable discussions and for Remark 2.

References

1. Ajtai, M., and Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In *The* 29*th annual ACM symposium on Theory of computing* (1997), pp. 284–293.
2. Anshel, I., Anshel, M., Fisher, B., and Goldfeld, D. New key agreement protocols in braid group cryptography. In *Proceedings of the* 2001 *Conference on Topics in Cryptology: The Cryptographer's Track at RSA, LNCS* (2001), vol. 2020, pp. 13–27.
3. Anshel, I., Anshel, M., and Goldfeld, D. An algebraic method for public-key cryptography. *Mathematical Research Letters 6* (1999), 287–291.
4. Anshel, I., Anshel, M., and Goldfeld, D. Non-abelian key agreement protocols. *Discrete Applied Mathematics 130*, 1 (2003), 3–12.

5. Anshel, M. Braid group cryptography and quantum cryptoanalysis. In *Proceedings of the 8th International Wigner Symposium* (2003), pp. 13–27.
6. Apostol, T. M. *Modular Functions and Dirichlet Series in Number Theory*. Springer, New York, 1990.
7. Arora, S., and Barak, B. *Complexity Theory: A Modern Approach*. `http://www.cs.princeton.edu/theory/complexity/`, 2008.
8. Benaloh, J. Dense probabilistic encryption. In *Proceedings of the 1st Annual Workshop on Selected Areas in Cryptology* (1994), pp. 120–128.
9. Bennett, C. H., and Shor, P. W. Quantum information theory. *IEEE Transactions on Information Theory 44*, 6 (1998), 2724–2742.
10. Blake, I., Seroussi, G., and Smart, N. *Elliptic Curves in Cryptography*, vol. 265 of *London Mathematical Society*. Cambridge University Press, 1999.
11. Blass, A., and Gurevich, Y. Matrix transformation is complete for the average case. *SIAM J. Comput. 24*, 1 (1995), 3–29.
12. Blum, N. A boolean function requiring $3n$ network size. *Theoretical Computer Science 28* (1984), 337–345.
13. Bogdanov, A., and Trevisan, L. On worst-case to average-case reductions for np problems. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)* (2003), pp. 308–317.
14. Boneh, D., and Venkatesan, R. Breaking rsa may not be equivalent to factoring. In *Proceedings of EuroCrypt '98, LNCS* (1998), vol. 1233, pp. 59–71.
15. Brown, D. R. L. Breaking rsa may be as difficult as factoring. Tech. Rep. 2005/380, Cryptology ePrint Archive, 2005.
16. Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory IT-22* (1976), 644–654.
17. Dwork, C. Positive applications of lattices to cryptography. In *The 22nd International Symposium on Mathematical Foundations of Computer Science (MFCS'97), LNCS* (1997), vol. 1295, pp. 44–51.
18. Evan, S., and Yacobi, Y. Cryptography and np-completeness. In *7th International Colloquium on Automata, Languages and Programming (ICALP'80)* (1980), pp. 195–207.
19. Feigenbaum, J., and Merritt, M. Open questions, talk abstracts, and summary of discussions. In *DIMACS series in discrete mathematics and theoretical computer science*, vol. 2. AMS, 1991, pp. 1–45.
20. Fellows, M., and Koblitz, N. Combinatorial cryptosystems galore! *Contemporary Mathematics 168* (1994), 51–61.
21. Goldreich, O. *Introduction to Complexity Theory. Lecture Notes*. Weizmann Institute of Science, 1998-99.
22. Goldwasser, S., and Bellare, M. *Lecture notes on cryptography*. Summer course on cryptography at MIT, 2001.
23. Goldwasser, S., and Micali, S. Probabilistic encryption. *Journal of Computer System Sciences 28* (1984), 270–299.
24. Grigoriev, D. Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. *Theoretical Computer Science 180* (1997), 217–228.
25. Grigoriev, D. Public-key cryptography and invariant theory. *Journal of Mathematical Sciences 126*, 3 (2005), 1152–1157.
26. Grigoriev, D., Hirsch, E. A., and Pervyshev, K. A complete public-key cryptosystem. *Groups, Complexity, Cryptology 1* (2009), 1–12.
27. Grigoriev, D., Kojevnikov, A., and Nikolenko, S. I. Invariant-based cryptosystems and their security against provable break. Tech. Rep. 158, Max-Planck-Institüt preprints, 2007.
28. Grigoriev, D., and Ponomarenko, I. Homomorphic public-key cryptosystems over groups and rings. *Quaderni di Mathematica 13* (2004), 305–325.
29. Grigoriev, D., and Ponomarenko, I. N. On non-Abelian homomorphic public-key cryptosystems. *Journal of Mathematical Sciences 126*, 3 (2005), 1158–1166.
30. Grigoriev, D., and Ponomarenko, I. Homomorphic public-key cryptosystems and encrypting boolean circuits. *Applicable Algebra in Engineering, Communication, and Computing 17* (2006), 239–255.
31. Grigoriev, D., and Ponomarenko, I. Constructions in public-key cryptography over matrix groups. In *Contemporary Mathematics: Algebraic Methods in Cryptography*, L. Gerritzen, D. Goldfeld, M. Kreuzer, R. Gerhard, and V. Shpilrain, Eds., Contemporary Mathematics, vol. 418. AMS, Providence, RI, 2007, pp. 103–120.
32. Hankerson, D., Menezes, A., and Vanstone, S. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

33. Harnik, D., Kilian, J., Naor, M., Reingold, O., and Rosen, A. On robust combiners for oblivious transfers and other primitives. In *Proceedings of EuroCrypt '05, LNCS* (2005), vol. 3494, pp. 96–113.
34. Hiltgen, A. P. Constructions of freebly-one-way families of permutations. In *Proc. of AsiaCrypt '92* (1992), pp. 422–434.
35. Hiltgen, A. P. Towards a better understanding of one-wayness: Facing linear permutations. In *Proceedings of EuroCrypt '98, LNCS* (1998), vol. 1233, pp. 319–333.
36. Hirsch, E. A., and Nikolenko, S. I. A feebly trapdoor function. *PDMI Preprints 16/2008*, 2008.
37. Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation 48* (1987), 203–209.
38. Kojevnikov, A., and Nikolenko, S. I. New combinatorial complete one-way functions. In 25^{th} *Symposium on Theoretical Aspects of Computer Science* (2008).
39. Lempel, A. Cryptography in transition. *Computing Surveys 11*, 4 (1979), 215–220.
40. Levin, L. A. One-way functions and pseudorandom generators. *Combinatorica 7*, 4 (1987), 357–363.
41. Levin, L. A. The Tale of One-Way Functions. *Problems of Information Transmission 39*, 1 (2003), 92–103.
42. Lo, H.-K., Spiller, T., and Popescu, S. *Introduction to Quantum Computation and Information.* World Scientific Publishing Company, 1998.
43. Luks, E. M. Permutation groups and polynomial-time computation. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (1993), vol. 11, (DIMACS, 1991), AMS, pp. 139–175.
44. Ly, L. V. Polly two: A new algebraic polynomial-based public-key scheme. *Applicable Algebra in Engineering, Communication, and Computing 17* (2006), 267–283.
45. Miller, V. S. Use of elliptic curves in cryptography. In *Proceedings of Crypto '85* (1985), vol. 218 of *Lecture Notes in Computer Science*, pp. 417–426.
46. Myasnikov, A. G., Shpilrain, V., and Ushakov, A. *Group–based cryptography.* Birkhäuser, 2008.
47. Naccache, D., and Stern, J. A new public-key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communication Security* (1998), pp. 59–66.
48. Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.
49. Okamoto, T., and Uchiyama, S. A new public-key cryptosystem as secure as factoring. In *Proceedings of EuroCrypt'98, LNCS* (1998), vol. 1403, pp. 308–317.
50. Rappe, D. K. Algebraisch homomorphe kryptosysteme. Diplomarbeit, Dem Fachbereich Mathematik der Universitat Dortmund, 2000.
51. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *The 37th annual ACM symposium on Theory of computing* (2005), pp. 84–93.
52. Regev, O. Lattice-based cryptography. In *The 26th Annual International Cryptology Conference (CRYPTO'06), LNCS* (2006), vol. 4117, pp. 131–141.
53. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.
54. Rivest, R. L., and Kaliski, B. Rsa problem. In *Encyclopedia of Cryptography and Security.* Kluwer Publishing House, 2005.
55. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing 26*, 5 (1997), 1484–1509.
56. Smith, L. *Polynomial Invariants of Finite Groups*, vol. 6 of *Research Notes in Mathematics.* A. K. Peters, Wellesley, Massachusets, 1996.
57. Venkatesan, R., and Rajagopalan, S. Average case intractability of matrix and diophantine problems. In *24th Annual ACM Symposium on Theory of Computing (STOC'92)* (1992), pp. 632–642.
58. Washington, L. *Elliptic Curves: Number Theory and Cryptography.* Chapman & Hall / CRC, 2003.
59. Wegener, I. *The Complexity of Boolean Functions.* B. G. Teubner, and John Wiley & Sons, 1987.
60. Yao, A. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science (FOCS'86)* (1986), pp. 162–187.