The RSA group is pseudo-free

Daniele Micciancio[1]

Department of Computer Science and Engineering
University of California at San Diego, La Jolla CA 92093, USA,
daniele@cs.ucsd.edu,
WWW home page: http://www.cse.ucsd.edu/users/daniele

Abstract. We prove, under the strong RSA assumption, that the group of invertible integers modulo the product of two safe primes is pseudo-free. More specifically, no polynomial time algorithm can output (with non negligible probability) an unsatisfiable system of equations over the free abelian group generated by the symbols $g_1, \ldots, g_n$, together with a solution modulo the product of two randomly chosen safe primes when $g_1, ..., g_n$ are instantiated to randomly chosen quadratic residues. Ours is the first provably secure construction of pseudo-free abelian groups under a standard cryptographic assumption, and resolves a conjecture of Rivest (TCC 2004).

1 Introduction

The notion of "pseudo-free group", put forward by Hohenberger in [10] and subsequently refined by Rivest [20], informally describes a finite computational group (i.e. a group that admits an efficient algorithmic implementation) with the security property that it is computationally hard to find solutions to any non-trivial equation over the group. More specifically, Rivest [20] defines pseudo-free (abelian) groups as computational (commutative) groups such that no polynomial time adversary, given random group elements $g_1, \ldots, g_n$ (chosen using to an appropriate sampling procedure), can output (with non negligible probability) an equation which is unsatisfiable over the free abelian group generated by the symbols $g_1, \ldots, g_n$, together with a solution to the equation in the computational group. As shown in [20], pseudo-freeness is a very strong assumption, and it implies many other computational assumptions typically used in cryptography, like the hardness of computing discrete logarithms and the RSA assumption in its standard and strong version. Each of these computational assumptions corresponds to a specific class of equations, e.g., the strong RSA assumption asserts that it is computationally infeasible to come up with an equation of the form $x^e = g$ (which is unsatisfiable over the free group $\{g^i : i \in \mathbb{Z}\}$ for $e > 1$) together with a solution $x = h$ such that $h^e = g$ in the multiplicative group $\mathbb{Z}_N^*$ of the invertible integers modulo the product $N = PQ$ of two large primes.

Free groups are widely used in computer science, and most modern cryptography relies on the hardness of computational problems over finite groups. So, as argued in [20], pseudo-free groups are a very interesting notion from a cryptographic perspective. For example, (non abelian) free groups are used in the so called Dolev-Yao model [7] for the symbolic analysis of public key cryptographic protocols. In the last few years, there have been several efforts to bridge the gap between the symbolic model of [7] (typically used in the area of formal methods for the analysis of security protocols) and the standard computational model used in cryptography (see for example [1, 18, 19, 17, 13, 11, 3]) with the goal of proving computational soundness results for symbolic analysis methods. An interesting question is whether pseudo-free groups can be used to extend (in a computationally sound way) the Dolev-Yao security model (in which encryption and decryption are viewed as black-box operations with no algebraic properties) with richer data structures and cryptographic functions (e.g., homomorphic encryption schemes) that make fundamental use of computational groups. Other motivations for studying pseudo-free groups mentioned in [20] are the following:

- Using a stronger assumption (that subsumes many other common cryptographic assumptions, like the hardness of computing discrete logarithms, and the strong RSA assumption) may make proofs easier.
- As the strong RSA assumption has been very useful in the construction of many cryptographic functions [8, 4, 6] which are not known to be secure under the standard version of the RSA assumption, assuming that a group is pseudo-free may allow an even wider range of applications.
- Pseudo-freeness has been linked [10] to the construction of specific cryptographic primitives, like directed transitive signature schemes, for which no solution is currently known.

The main question left open by Rivest in [20] is: do pseudo-free groups exist?

In [20] Rivest suggested the RSA group $\mathbb{Z}_N^*$ (where $N = PQ$ is the product of two large primes) as a possible candidate pseudo-free abelian group, and nicknamed the corresponding conjecture the "super-strong RSA assumption". In this paper we resolve Rivest's conjecture and prove that $\mathbb{Z}_N^*$ is pseudo-free under the strong RSA assumption, at least when $N = PQ$ is the product of two "safe primes" (i.e., odd primes such that $p = (P-1)/2$ and $q = (Q-1)/2$ are also prime[1]), a special class of prime numbers widely used in cryptography. In other words, we prove that if the strong RSA assumption holds true, then the super-strong RSA assumption also holds. Our result is the first example of provably secure pseudo-free group based on a standard cryptographic assumption. In fact, we prove that the RSA group satisfies an even stronger version of the pseudo-freeness property than the one defined in [20]: we show that no adversary can efficiently compute an unsatisfiable *system* of equations (as opposed to a single equation) together with a solution in the given computational group.

Our proof is based on a rewriting process that, starting from an arbitrary equation (or system of equations), yields simpler and simpler equations with the following properties:

- unsatisfiable equations over the free group are mapped to unsatisfiable equations over the free group, and

[1] Equivalently, using more standard mathematical terminology, $P = 2p + 1$ and $Q = 2q + 1$ where p and q are *Sophie-Germain* primes.

– solutions to the original equations (over a computational group) can be efficiently mapped to solutions to the resulting equations (over the same computational group).

Some of our transformations work for arbitrary groups and might be of independent interest. For example, we show how to transform systems of equations into a single equation (Theorem 4), how to map equations in several variables to univariate equations (Lemma 3), and how to map unsatisfiable equations of the form $x^e = g^d$ (where e and d are arbitrary integers) to equations where the exponents are of the form $e = c^{h+1}$ and $d = c^h$ (Theorem 3).

Organization The rest of the paper is organized as follows. In Section 2 we introduce basic definitions and notation for equations and groups. In Section 3 we prove that the RSA group satisfies the basic definition of pseudo-free group (involving a single equation). In Section 4 we extend the result to systems of equations. Section 5 concludes with a discussion of open problems.

2 Preliminaries

A group is an algebra with a binary associative operation $\circ$, a unary operation $()^{-1}$ (inverse) and a constant 1 (identity) satisfying the equational axioms $(x \circ y) \circ z = x \circ (y \circ z)$, $x \circ 1 = 1 \circ x = x$, and $x \circ (x)^{-1} = (x)^{-1} \circ x = 1$. A group is abelian if the operation $\circ$ is commutative, i.e., it also satisfies $x \circ y = y \circ x$. In this paper we are interested in computational groups, i.e., groups that admit an efficient algorithmic implementation.

Definition 1. *A computational group (associated to group $(G, \circ, ()^{-1}, 1)$) is defined by a mapping $\langle\cdot\rangle: G \to \{0,1\}^*$ (the representation function) such that the following operations can be performed in polynomial time:*

- *Test membership in $\langle G\rangle$, i.e., given a string x, determine if it is a valid representation of a group element.*
- *Given $\langle x\rangle$ and $\langle y\rangle$, compute $\langle x \circ y\rangle$.*
- *Given $\langle x\rangle$, compute $\langle x^{-1}\rangle$.*
- *Compute the representation of the group identity element $\langle 1\rangle$.*
- *Sample a group element $\langle g\rangle \in \langle G\rangle$ (with not necessarily uniform probability distribution.)*

For simplicity, in the definition above, we focused on computational groups in which each group element has a unique representation, as all the computational groups studied in this paper have this property. (The definition of computational group can be easily extended to cases where group elements may have multiple representations, by introducing an efficiently computable equivalence relation on group representations.) The requirement that membership in $\langle G\rangle$ be efficiently decidable is also not strictly necessary, but convenient, and all computational groups studied in this paper have this property. Also, sometimes the definition of computational group requires the distribution output by the sampling algorithm $\langle g \rangle \in \langle G \rangle$ to be uniform over G, while other times no sampling algorithm is required at all. In this paper $\langle g \rangle \in \langle G \rangle$ is an arbitrary sampling procedure, which is used to generate nontrivial group elements.

For brevity, we identify computational groups with the underlying mathematical group, and write $x \circ y$, x^{-1}, etc., to denote the corresponding operation on the representations of the group elements. Also, we use multiplicative notation xy for the group binary operation $x \circ y$, and use exponential notation x^n to denote the n-fold composition of x with itself. Formally, x^n is defined inductively by the rules $x^0 = 1$, $x^{n+1} = x \circ x^n$. The notation is extended to negative exponents in the obvious way $x^{-n} = (x^n)^{-1}$.

For any set of symbols A, the free abelian group $\mathcal{F}(A)$ generated by A is the initial algebra with constant symbols A satisfying the abelian group equations. It is easy to see that each group element has a unique representation as a product $\prod_{a \in A} a^{d_a}$, where $d_a \in \mathbb{Z}$ for all $a \in A$.

Let X and A be two disjoint finite sets of variable and constant symbols. We define $X^{-1} = \{x^{-1} : x \in X\}$ and $A^{-1} = \{a^{-1} : a \in A\}$. A group equation over variables X and constants A, is a pair $E = (w_1, w_2)$ of words (usually written as $E : w_1 = w_2$) over the alphabet $X \cup X^{-1} \cup A \cup A^{-1}$. Unless otherwise specified, we interpret E as an equation over the free group $\mathcal{F}(A)$. A solution to $E : w_1 = w_2$ (over the free group $\mathcal{F}(A)$) is a function $\sigma : X \to \mathcal{F}(A)$ such that $\sigma(w_1) = \sigma(w_2)$, where σ is extended to words over $X \cup X^{-1} \cup A \cup A^{-1}$ homomorphically in the obvious way. We say that an equation $E : w_1 = w_2$ is satisfiable (over the free group) if it admits a solution. We say it is unsatisfiable otherwise.

Let G be a (computational) group. A group equation over G (denoted E_α) is defined by an equation E over variables X and constants A, and a function $\alpha : A \to G$. A solution to equation $E_\alpha : w_1 = w_2$ is a function $\xi : X \to G$ such that $(\alpha \cup \xi)(w_1) = (\alpha \cup \xi)(w_2)$.

From a computational point of view, we assume that equations $E : w_1 = w_2$ are represented using compact notation for exponential expressions a^i, so that exponentially large exponents i can be stored in polynomial space. This is easily seen to be equivalent to many other formalisms to compactly represent terms w_1, w_2, like, for example, the straight-line programs used in [10].

2.1 Statistical Distance

A function f is negligible if it decreases faster than any inverse polynomial, i.e., for any $c > 0$ there is an n_0 such that $|f(n)| \leq 1/n^c$ for any $n > n_0$.

Let X and Y be two discrete random variables over a (countable) set A. The statistical distance between X and Y is the quantity

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in A} |\Pr\{X = a\} - \Pr\{Y = a\}|.$$

In this paper we use the fact that for any two random variables X and Y over set A, and predicate $p : A \to \{0, 1\}$,

$$|\Pr[p(X) = 1] - \Pr[p(Y) = 1]| \leq \Delta(X, Y).$$

In particular, if $p(X)$ happens with non negligible probability, and $\Delta(X,Y)$ is negligible, then also $p(Y)$ happens with non negligible probability.

2.2 Pseudo-free groups

Intuitively, a computational group is pseudo-free if no efficient algorithm can find a nontrivial relation among randomly chosen group elements, i.e., an equation (or system of equations) which is unsatisfiable over the free group, together with a solution over the computational group. Since for any finite group G, the equation $x^{|G|+1} = a$ is unsatisfiable over the free group $\mathcal{F}(\{a\})$, but has solution $x = a$ over G, in order to properly define pseudo-free groups we need to consider families of groups $\{G_N\}$ where N is chosen at random from a probability ensemble $\mathcal{N}$. In particular, given a randomly chosen $N \in \mathcal{N}$, the order of the group $o(G_N)$ should be hard to compute.

Definition 2. *A family of computational groups $\mathcal{G} = \{G_N\}_{N \in \mathcal{N}}$ is pseudo-free if for any set A of polynomial size $|A| = p(k)$ (where k is a security parameter), and probabilistic polynomial (in k) time algorithm $\mathcal{A}$, the following holds. Let $N \in \mathcal{N}(k)$ be a randomly chosen group index, and $\alpha: A \to G_N$ a function defining $|A|$ group elements chosen independently at random according to the computational group sampling procedure. Then, the probability that $\mathcal{A}(N, \alpha) = (E, \xi)$ outputs an unsatisfiable equation E (over variables X and constants A) together with a solution $\xi: X \to G_N$ to E_α over G_N, is negligible.*

2.3 The RSA group

In this paper, we study the group $\mathbb{Z}_N^*$ of invertible integers modulo N. This is a computational group, with the usual representation of each group element as an integer in $\{0, \ldots, N-1\}$. Membership $g \in \mathbb{Z}_N^*$ can be easily tested by computing $\gcd(g, N)$ and checking that $\gcd(g, N) = 1$. The group $\mathbb{Z}_N^*$ can be efficiently sampled uniformly at random by picking an integer $g \in \{0, \ldots, N-1\}$ with uniform distribution, and checking if $g \in \mathbb{Z}_N^*$. However, in this paper, it is more convenient to consider the computational group $\mathbb{Z}_N^*$ together with a different sampling procedure that chooses g at random from a subgroup of $\mathbb{Z}_N^*$. An element $g \in \mathbb{Z}_N^*$ is called a quadratic residue if $g = h^2 \bmod N$ for some $h \in \mathbb{Z}_N^*$. The set of quadratic residues modulo N is denoted QR_N, and it is a subgroup of $\mathbb{Z}_N^*$. The subgroup QR_N can be efficiently sampled by picking $h \in \mathbb{Z}_N^*$ uniformly at random and setting $g = h^2 \bmod N$. Unless otherwise specified, in this paper we always consider the computational group $\mathbb{Z}_N^*$ with this sampling procedure that selects g uniformly at random from QR_N.

When $N = P \cdot Q$ is the product of two prime numbers, $\mathbb{Z}_N^*$ is commonly called an RSA group, after the encryption function of Rivest, Shamir and Adleman [21], which started a widespread use of these groups in cryptography. In this paper we are interested in RSA groups where P and Q are primes of special form. A prime number p is called a Sophie-Germain prime if $2p+1$ is also prime. In the cryptographic literature, the number $2p+1$ (where p is a Sophie-Germain prime) is usually called a safe prime. In other words, a safe prime $P = 2p+1$ is an odd prime number such that $p = (P-1)/2$ is also prime. Safe primes are relatively easy to find in practice (e.g., by choosing p at random and testing p and $2p+1$ for primality), although there is no known mathematical proof showing that there are infinitely many of them. Safe primes are widely used in cryptography. For example, the RSA group $\mathbb{Z}_N^*$ where $N = P \cdot Q$ is the product of two safe primes has been used in [8, 9, 6].

Let $\mathcal{N}$ be the set of all safe prime products. We assume some standard probability distribution on $\mathcal{N}$, as typically used in many cryptographic applications. (E.g., choose N as the product of two random k-bit safe primes.) The following computational problems are considered hard, and have been used as the basis for many cryptographic applications:

- Factoring problem: given an integer $N \in \mathcal{N}$, compute prime factors P, Q such that $N = P \cdot Q$;
- RSA problem: given an integer $N \in \mathcal{N}$, an integer e relatively prime with $\phi(N) = (P-1)(Q-1)$, and a randomly chosen group element $\gamma \in \mathbb{Z}_N^*$, compute a $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$;
- Strong RSA problem: given an integer $N \in \mathcal{N}$, and a randomly chosen group element $\gamma \in \mathbb{Z}_N^*$, output an integer $e > 1$ and a group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$.

In this paper we are primarily interested in the strong RSA problem and its relation to pseudo-freeness. It is convenient to consider the following variant of the strong RSA problem where the input γ is chosen as a random quadratic residue:

- Strong QR-RSA problem: given an integer $N \in \mathcal{N}$, and a randomly chosen quadratic residue $\gamma \in QR_N$, output an integer $e > 1$ and a group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$.

It can be easily shown [6] that this variant is not any easier than the standard strong RSA problem.

Theorem 1 (See [6], Section 4). *If the strong RSA problem modulo a safe prime product is hard, then the strong QR-RSA problem modulo a safe prime product is also hard.*

For any prime product $N = P \cdot Q$, the group $\mathbb{Z}_N^*$ has cardinality $o(\mathbb{Z}_N^*) = \phi(N) = (P-1)(Q-1)$ and it is isomorphic to $\mathbb{Z}_P^* \times \mathbb{Z}_Q^*$, with isomorphism given by $\xi \mapsto (\xi \bmod P, \xi \bmod Q)$. If $P = 2p+1$ and $Q = 2q+1$ are safe primes, the group $\mathbb{Z}_N^*$ has order $4pq$, and the subgroup $QR_N \subset \mathbb{Z}_N^*$ has order $o(QR_N) = pq$. In particular, all elements in QR_N have order[2] $1, p, q$ or pq.

[2] The order of an element γ in a group G is the smallest positive integer $o(\gamma) \geq 1$ such that $\gamma^{o(\gamma)} = 1$.

3 The RSA group is pseudo-free

In this section we prove, under the strong RSA assumption, that the RSA group $\mathbb{Z}_N^*$ (where N is the product of two safe primes, and elements are sampled uniformly at random from QR_N) is pseudo-free.

Theorem 2. *Let $\mathcal{N}$ be a distribution over safe prime products such that the strong RSA problem modulo $N \in \mathcal{N}$ is hard. Then the family of computational groups $\mathbb{Z}_N^*$ of invertible integers modulo $N \in \mathcal{N}$ (with the modular multiplication group operation, and uniform sampling procedure over QR_N) is pseudo-free.*

Proof. Assume that $\mathbb{Z}_N^*$ is not pseudo-free, i.e., there is a probabilistic polynomial time algorithm $\mathcal{A}$ that on input a randomly chosen $N \in \mathcal{N}(k)$ and random group elements $\alpha: A \to QR_N$ (for some polynomial sized set A), outputs an unsatisfiable equation $E: w_1 = w_2$ (over constants A and variables X) together with a solution $\xi: X \to \mathbb{Z}_N^*$ to E_α over the group $\mathbb{Z}_N^*$. We use $\mathcal{A}$ to solve the strong QR-RSA problem for the same distribution of the modulus N. Namely, given a randomly chosen $N \in \mathcal{N}(k)$ and $\gamma \in QR_N$, we compute an integer $e > 1$ and group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma$. By Theorem 1 this also implies an algorithm to solve the standard strong RSA problem.

The reduction works as follows. Let (N, γ) be an instance of the strong QR-RSA problem. We begin by checking if γ is a generator for QR_N. This can be easily done using the following lemma.[3]

Lemma 1. *Let $N = P \cdot Q$ be the product of two distinct safe primes, and $\gamma \in QR_N$ a quadratic residue. Then γ is a generator for QR_N if and only if* $\gcd(\gamma - 1, N) = 1$.

If γ is not a generator for QR_N, then we can easily solve the strong QR-RSA problem instance (N, γ) as described below. Given N and $\gamma \in QR_N$, we compute $g = \gcd(\gamma - 1, N)$. Since $N = PQ$, it must be $g \in \{1, P, Q, PQ\}$. We distinguish three cases.

- If $g = PQ = N$, then N divides $\gamma - 1$, and $\gamma = 1 \pmod{N}$. So, we can immediately output a solution to the strong QR-RSA input problem (N, γ), e.g., $(\xi, e) = (1, 3)$.
- If $g \in \{P, Q\}$, then we can easily compute $\phi(N) = (P - 1) \cdot (Q - 1) = (g - 1)(N/g - 1)$. This also easily yields a solution $(\xi, e) = (\gamma, \phi(N) + 1)$ to the strong QR-RSA problem (N, γ).
- If $g = 1$, then by Lemma 1 γ is a generator for QR_N and we proceed as follows.

In the rest of the proof we assume that γ is a generator of QR_N. We use γ to sample the group elements $\alpha(a) \in QR_N$ and generate an input instance (N, α) for algorithm $\mathcal{A}$. Since $\mathcal{A}$ works only with non negligible probability, we

[3] The proof of this and other lemmas are omitted because of space limitations. All proofs can be found in the full version of the paper on the author's web page.

need the input values $\alpha(a)$ to be distributed (almost) uniformly at random over QR_N. The following lemma shows that γ can be used to sample QR_N almost uniformly at random.

Lemma 2. *For any cyclic group G and generator $\gamma \in G$, if $v \in \{0, \ldots, B-1\}$ is chosen uniformly at random, then the statistical distance between γ^v and the uniform distribution over G is at most $|G|/2B$.*

For any $a \in A$, choose $v_a \in [0, \ldots, N \cdot |A| \cdot K - 1]$ uniformly at random for some super-polynomial function $K(k) = k^{\omega(1)}$, and set $\alpha(a) = \gamma^{v_a}$. By Lemma 2, the statistical distance between $\alpha(a)$ and the uniform distribution over QR_N is at most $|QR_N|/2N|A|K \leq 1/|A| \cdot K$. Since the values $\alpha(a)$ are independently chosen, the statistical distance between α and a uniformly chosen assignment is at most $1/2K = k^{-\omega(1)}$.

Invoke algorithm $\mathcal{A}$ on input (N, α). We know that when α is distributed uniformly at random, algorithm $\mathcal{A}$ is successful with non negligible probability $\delta(k) = k^{-O(1)}$. Since α is within negligible statistical distance $1/K(k)$ from uniform, $\mathcal{A}$ succeeds on input α at least with non negligible probability $\delta(k) - 1/K(k)$. In the rest of the proof, we assume $\mathcal{A}$ is successful, and we consider the conditional success probability of the reduction. We will show that the conditional success probability is at least $1/3$.

Fix the value of N, generator $\gamma \in QR_N$, and input (N, α) passed to algorithm $\mathcal{A}$. Let $E : w_1 = w_2$ and ξ be the equation and solution to E_α returned by $\mathcal{A}$. Remember that, for every $a \in A$, $\alpha(a) = \gamma^{v_a}$ for a randomly chosen $v_a \in \{0, \ldots, N \cdot |A| \cdot K - 1\}$. For any $a \in A$, let $w_a = v_a \bmod pq$ and $z_a = (v_a - w_a)/pq$. We remark that although the values v_a are known, and w_a, z_a are uniquely determined by v_a, the values w_a and z_a cannot be easily computed from v_a because the product pq is not known. Therefore, the values w_a and z_a cannot be used in the reduction process. We will use w_a and z_a only in the analysis of the reduction.

Notice that, given w_a, the conditional distribution of z_a is uniform over the set

$$S_a = \{0, \ldots, \lfloor (N|A|K - 1 - w_a)/pq \rfloor\}. \tag{1}$$

Also, given w_a, $\alpha(a) = \gamma^{v_a} = \gamma^{w_a}$ is uniquely determined, and z_a is uniformly distributed over the set S_a independently from α, E and ξ. In particular, the integers $z_a \in S_a$ are uniformly distributed independently from the success of algorithm $\mathcal{A}$.

Assume that $\mathcal{A}$ is successful, i.e., E is unsatisfiable over $\mathcal{F}(A)$, and $\xi : X \to QR_N$ is a valid solution to E_α. We use equation E and solution ξ to solve the original strong QR-RSA problem (N, γ). This is done in two steps. First, we transform equation E and solution ξ to E_α, into a new unsatisfiable equation E' and solution ξ' to E'_α containing only one variable symbol. Then, E' and ξ' are used to solve the strong QR-RSA problem (N, γ).

The equation and solution (E, ξ) is transformed into a univariate equation and solution (E', ξ') using the following lemma.

Lemma 3. *For any computational group G, there is a polynomial time algorithm that on input an equation E over constants A and variables X, and a variable assignment $\xi : X \to G$, outputs a univariate equation E' and value $\xi' \in G$, such that*

- *if E is unsatisfiable over the free group $\mathcal{F}(A)$, then E' is also unsatisfiable over $\mathcal{F}(A)$; and*
- *for any assignment $\alpha : A \to G$, if ξ is a solution to E_α then ξ' is a solution to E'_α.*

At this point we have an unsatisfiable equation of the form $E' : x^e = \prod_a a^{d_a}$ and a solution $\xi' \in \mathbb{Z}_N^*$ to E'_α. Notice that E' is satisfiable over the free group $\mathcal{F}(A)$ if and only if $e | \gcd(d_a : a \in A)$. So, it must be $e \nmid \gcd(d_a : a \in A)$. Also, from the definition of $\alpha(a)$, we know that

$$(\xi')^e = \prod_a \alpha(a)^{d_a} = \gamma^{\sum_a v_a d_a}. \tag{2}$$

Assume without loss of generality that $e \geq 0$ and $d = \sum_a v_a d_a \geq 0$. (Otherwise, change the sign of e and/or the d_a for all $a \in A$, and possibly replace ξ' with $(\xi')^{-1}$ in order to satisfy (2).) In the rest of the proof we distinguish various cases, depending on the value of $\gcd(e, pq)$.

- If $\gcd(e, pq) = pq$ and $e \neq 0$, then we can immediately output the solution $(\gamma, e + 1)$ to the strong QR-RSA problem (N, γ) because $o(\gamma) = pq$ and $\gamma^{e+1} = \gamma \cdot \gamma^e = \gamma \pmod N$. We remark that, although we cannot compute $\gcd(e, pq)$ (or even check if $\gcd(e, pq) = pq$) because pq is not know, we can guess that this is the case, and simply check if $\gamma, e + 1)$ is indeed a solution to the given strong QR-RSA problem. Similar remarks apply to the other cases below.
- If $\gcd(e, pq) \in \{p, q\}$, then $o(\gamma^e) = pq/\gcd(e, pq) \in \{p, q\}$. In particular, γ^e is not a generator of QR_N, and, by Lemma 1, $\gcd(\gamma^e - 1, N) \neq 1$. Since $\gamma^e \neq 1 \pmod N$, we also have $\gcd(\gamma^e - 1, N) \neq N$. Therefore, it must be $g = \gcd(\gamma^e - 1, N) \in \{P, Q\}$. So, we can compute $\phi(N) = (P - 1)(Q - 1) = (g-1)(N/g-1)$, and output the solution $(\gamma, \phi(N)+1)$ to the strong QR-RSA problem (N, γ).
- The remaining cases are when $e = 0$ or $\gcd(e, pq) = 1$, and are described below.

If $e = 0$, Lemma 4 below shows that $d = 0$ with probability at most $1/2$. It follows that with probability at least $1/2$, $(\gamma, d + 1)$ is a solution to the strong QR-RSA problem (N, γ) because $d + 1 > d \geq 1$ and $\gamma^{d+1} = \gamma \cdot \gamma^d = \gamma \cdot \xi^0 = \gamma$. So, the conditional success probability of the reduction is at least $1/2$.

Lemma 4. *The conditional probability (given α, $e = 0$, and $\{d_a : a \in A\}$ such that $e \nmid \gcd\{d_a : a \in A\}$) that $d = \sum_a v_a d_a \neq 0$ is at least $1/2$.*

The last case to consider is when $\gcd(e, pq) = 1$. This is the most complicated of all cases. This time, we first show that $e|d$ with probability at most $2/3$.

Lemma 5. *The conditional probability (given α, $\gcd(e, pq) = 1$, and $\{d_a : a \in A\}$ such that $e \nmid \gcd\{d_a : a \in A\}$) that e divides $d = \sum_a v_a d_a$ is at most $2/3$.*

We conclude the reduction showing that if $e > 0$ and $e \nmid d$, then we can solve the strong QR-RSA problem (N, γ). The proof is based on the following theorem.

Theorem 3. *For any abelian group, there is a polynomial time algorithm that on input (γ, ξ, e, d), where γ, ξ are group elements and e, d integers, satisfying $\xi^e = \gamma^d, e \neq 0, e \nmid d$, outputs (θ, c, h) such that $\theta^{c^{h+1}} = \gamma^{c^h}$, $|c| \geq 2$, $c^{h+1}|e$ and $c^h|d$.*

Proof. We define the algorithm $A(\gamma, \xi, e, d)$ recursively, by induction on the size of e and d. At each iteration, either d or e is replaced by a proper factor, while the other number is unchanged. It follows that the algorithm terminates within at most $\log_2(de)$ iterations.

Algorithm $A(\gamma, \xi, e, d)$ works as follows:

- If $d \nmid e$, compute $d_1 = \gcd(e, d)$ using the extended Euclidean algorithm to find integers e', d' such that $d_1 = e \cdot e' + d \cdot d'$. Then invoke recursively $A(\gamma, \xi^{d'}\gamma^{e'}, e, d_1)$ and output the result.
- Otherwise, $d|e$, and we can compute $c = e/d$. If d is a power of c, i.e., $d = c^h$ for some integer h, return (ξ, c, h).
- If d is not a power of c, let h be the largest exponent such that $c^h|d$. Invoke recursively $A(\gamma, \xi^{d/c^h}, c^{h+1}, d)$ and return the result.

We need to prove that the algorithm is correct, and that either e or d decreases at every iteration.

First consider the case $d \nmid e$. The input to the recursive call is given by $\xi_1 = \xi^{d'}\gamma^{e'}$, $e_1 = e$ and $d_1 = \gcd(e, d)$. Notice that

$$\xi_1^{e_1} = (\xi^{d'}\gamma^{e'})^e = (\xi^e)^{d'}\gamma^{ee'} = \gamma^{dd'+ee'} = \gamma^{d_1}.$$

Moreover, $e_1 = e \neq 0$, and $e_1 \nmid d_1$, because otherwise $e = e_1|d_1|d$, contradicting $e \nmid d$. So, the input to the recursive call is valid, i.e., it satisfies the assumptions in the theorem. In order to ensure termination, we need to check that d_1 properly divides d. Assume for contradiction $d = d_1$. From the definition of d_1, it follows that $d|e$, but this contradicts the condition $d \nmid e$ tested by the algorithm.

Now assume $d|e$. Notice that $d \neq 0$, because otherwise $e|d$. So, the quotient $c = e/d$ is well defined. Moreover, $|c| > 1$ because $e \neq 0$, and $e \nmid d$. If $d = c^h$, then the algorithm terminates with output $\theta = \xi$, c, h. Notice that

$$\theta^{c^{h+1}} = \xi^{dc} = \xi^e = \gamma^d = \gamma^{c^h},$$

i.e., the output is correct.

Finally, consider the case when $d|e$, but d is not a power of c. Notice that $d \neq 0$ because $e \nmid d$. Since $|c| > 1$ and $d \neq 0$, the maximum $h = \max\{h : c^h|d\}$

is well defined, and d/c^h is an integer. This time, the algorithm is recursively invoked on input $\xi_1 = \xi^{d/c^h}$, $e_1 = c^{h+1}$ and $d_1 = d$. This input satisfies

$$\xi_1^{e_1} = \xi^{dc^{h+1}/c^h} = \xi^{dc} = \xi^e = \gamma^d = \gamma^{d_1}.$$

Moreover $e_1 = c^{h+1} \neq 0$ because $c \neq 0$. Also, $e_1 \not| d_1$, because otherwise $c^{h+1} = e_1 | d_1 = d$, contradicting the maximality of h. This time, we want to prove that e_1 properly divides e. Clearly, $e_1 = c^h c | dc = e$. Now, assume $e_1 = e$. Then, $c^h = e_1/c = e/c = d$, and d is a power of c. □

Applying Theorem 3 to equation $\xi^e = \gamma^d$, we get values $c \geq 2$, $h \geq 0$ and $\theta \in \mathbb{Z}_N^*$ such that $\theta^{c^{h+1}} = \gamma^{c^h}$. If $\theta^c = \gamma$, then (θ, c) is a solution to the strong QR-RSA problem. So, assume $(\theta^c/\gamma) \neq 1$, and assume also, without loss of generality that h is the smallest integer such that $(\theta^c/\gamma)^{c^h} = 1$. Let $\delta = (\theta^c/\gamma)^{c^{h-1}}$. We know that $\delta \neq 1$ and $\delta^c = 1$.

The following lemma shows that δ is a quadratic residue.

Lemma 6. *The value $\delta = (\theta^c/\gamma)^{c^{h-1}}$ is a quadratic residue.*

Apply Lemma 1 to quadratic residue δ. Since $\delta \neq 1$, either δ is a generator for QR_N, or $\gcd(\delta - 1, N) \in \{P, Q\}$. As before, if $g = \gcd(\delta - 1, N) \in \{P, Q\}$, we can compute $\phi(N) = (P-1)(Q-1) = (g-1)(N/g-1)$ and output the trivial strong QR-RSA solution $(\gamma, \phi(N)+1)$

So, assume δ is a generator for QR_N. Since $\delta^c = 1$, it must be $pq = o(\delta)|c$. In particular, we also have $\gamma^c = 1$, and $(\gamma, c+1)$ is a solution to the strong QR-RSA problem.

This completes the proof that if the RSA group is not pseudo-free, then we can solve the strong RSA problem. □

4 Systems of equations

The intuition behind the definition of pseudo-free group is that no polynomial time adversary can "prove" that the given computational group is not free. The kind of proofs implicit in Definition 2 consist of a single equation which is unsatisfiable over the free group, but satisfiable over the computational group. This choice is motivated by the fact that unsatisfiability of equations over free groups and satisfiability over computational groups can be efficiently demonstrated. (Specifically, unsatisfiability over free abelian groups is decidable in polynomial time, and satisfiability over arbitrary computational groups can be proved by giving a satisfying assignment.) An immediate extension that comes to mind is to consider systems of equations. Satisfiability for systems of equations is defined in the obvious way: a variable assignment satisfies a system of equations if it simultaneously satisfies all the equations in the system. As observed in [20], for the case of non abelian free groups, the results in [14] (see also [12, Lemma 3 and Corollary 2 and 3]) allow to combine systems of equations into a single equation. Specifically, the method is based on showing that the two equations

$x = 1$ and $y = 1$ are equivalent to the single equation $x^2ax^2a^{-1} = (ybyb^{-1})^2$, and it allows to transform any finite system of equations into a single equation with exactly the same set of solutions. Unfortunately, the same is not true for abelian groups, and the set of solutions of a system of equations cannot in general be represented by a single equation. Consider for example the equations $x = 1$ and $y = 1$. The solution to this system is clearly unique. However, no single equation in two variables can have a unique solution. (Any bivariate equation has always either zero or infinitely many solutions over the free group.)

In this section we show that in the case of abelian groups, it is still possible to transform systems of equations into a single equation which is equivalent to the system, but in a weaker sense than having exactly the same set of solutions. The transformation maps any system of equations to a single equation whose solution set is a superset of the solutions to the system. However, if the system is unsatisfiable, then also the single equation is guaranteed to be unsatisfiable. This weaker notion of equivalence is enough to prove that Definition 2 is equivalent to the following seemingly stronger definition.

Definition 3. *A family of computational groups $\mathcal{G} = \{G_N\}_{N\in\mathcal{N}}$ is pseudo-free if for any set A of polynomial size $|A| = p(k)$ (where k is a security parameter), and probabilistic polynomial (in k) time algorithm $\mathcal{A}$, the following holds. Let $N \in \mathcal{N}(k)$ be a randomly chosen group index, and $\alpha\colon A \to G_N$ a function defining $|A|$ group elements chosen independently at random according to the computational group sampling procedure. Then, the probability that $\mathcal{A}(N, \alpha) = (\{E^i\}_{i\in I}, \xi)$ outputs an unsatisfiable system of equations $\{E^i\}_{i\in I}$ (over variables X and constants A) together with a solution $\xi\colon X \to G_N$ to $\{E^i_\alpha\}_{i\in I}$ over G_N, is negligible.*

The transformation from systems of equations to single equations is described in the following theorem.

Theorem 4. *There is a polynomial time algorithm that on input a system of equations $\{E^i\}_{i\in I}$ over constants A and variables X, outputs a single equation E over the same sets of constants A and variables X, such that the following holds.*

- *If $\{E^i\}_{i\in I}$ is unsatisfiable (over the free abelian group generated by A), then E is also unsatisfiable;*
- *For any computational group G and assignment $\alpha\colon A \to G$, any solution $\xi\colon X \to G$ to $\{E^i_\alpha\}_{i\in I}$ is also a solution to E_α.*

The proof of the theorem is based on elementary lattice techniques. For a detailed introduction to lattices and their computational complexity the reader is referred to [16]. Here we briefly recall the basic definitions and simple facts about lattices used in the proof of Theorem 4. For any matrix $\mathbf{M}$ with rational entries the lattice generated by a matrix $\mathbf{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n]$ is the set $\mathcal{L}(\mathbf{M}) = \{\sum_i x_i\boldsymbol{m}_i\colon x_i \in \mathbb{Z} \text{ for } i = 1, \ldots, n\}$ of all integer linear combinations of the columns of $\mathbf{M}$. There is a polynomial time algorithm that on input two rational matrices $\mathbf{M}$ and $\mathbf{M}'$, determines if $\mathcal{L}(\mathbf{M}) \subseteq \mathcal{L}(\mathbf{M}')$, and if not, finds a vector $\boldsymbol{u} \in \mathcal{L}(\mathbf{M}) \setminus \mathcal{L}(\mathbf{M}')$. The dual of a lattice $\mathcal{L}(\mathbf{M})$ is the set of all vectors $\boldsymbol{u}$ in the linear span of the columns of $\mathbf{M}$ that have integer scalar product with all lattice vectors in $\mathcal{L}(\mathbf{M})$. The dual of a lattice is a lattice, and the dual of the dual of a lattice equals the original lattice. The dual of a lattice $\mathcal{L}(\mathbf{M})$ is denoted $\hat{\mathcal{L}}(\mathbf{M})$. Moreover, there is a polynomial time algorithm that on input a rational matrix $\mathbf{M}$ outputs a rational matrix $\mathbf{M}'$ such that $\mathcal{L}(\mathbf{M}') = \hat{\mathcal{L}}(\mathbf{M})$. It immediately follows from the definition of dual lattice that $\mathcal{L}(\mathbf{M})$ is a sub-lattice of $\mathcal{L}(\mathbf{M}')$ (i.e., $\mathcal{L}(\mathbf{M}) \subseteq \mathcal{L}(\mathbf{M}')$) if and only if $\hat{\mathcal{L}}(\mathbf{M}')$ is a sub-lattice of $\hat{\mathcal{L}}(\mathbf{M})$ (i.e., $\hat{\mathcal{L}}(\mathbf{M}') \subseteq \hat{\mathcal{L}}(\mathbf{M})$). We are now ready to prove Theorem 4.

Proof. Let $\{E^i\}_{i\in I}$ be a system of equations over the set of constant symbols A and variables X, and let $\sigma\colon X \to \mathcal{F}(A)$ be a generic variable assignment. Write each equation E^i and the assignment $\sigma(x)$ as

$$E^i\colon \prod_{x\in X} x^{e_{i,x}} = \prod_{a\in A} a^{d_{i,a}}$$
$$\sigma(x) = \prod_{a\in A} a^{s_{x,a}},$$

where the $e_{i,x}$, $d_{i,a}$ and $s_{x,a}$ are integers for all $i \in I$, $x \in X$ and $a \in A$. We use notation $e_{*,*}$ to denote the matrix with $|I|$ rows and $|X|$ columns with integer entries $(e_{i,x})_{i\in I, x\in X}$, and $e_{i,*}$ and $e_{*,x}$ to denote the rows and columns of matrix $e_{*,*}$. The matrices $d_{*,*}$, $s_{*,*}$ and vectors $d_{i,*}$, $d_{*,a}$, $s_{x,*}$, $s_{*,a}$ are defined similarly. Notice that σ is a solution to the system of equations over the free group if and only if

$$\sum_{x\in X} e_{i,x} s_{x,a} = d_{i,a}$$

for all $i \in I$ and $a \in A$, or, equivalently, in matrix notation, $e_{*,*}s_{*,*} = d_{*,*}$. So, the system of equations is solvable over the free group if and only if the integer lattice $\mathcal{L}(e_{*,*})$ contains $\mathcal{L}(d_{*,*})$ as a sub-lattice. Moreover, the two lattices satisfy $\mathcal{L}(e_{*,*}) \supseteq \mathcal{L}(d_{*,*})$ if and only if their duals satisfy the reverse inclusion $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$. The inclusion $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$ can be checked using standard techniques, and if it is not satisfied, one can efficiently find a vector $(u_i)_{i\in I} \in \hat{\mathcal{L}}(e_{*,*})$ such that $(u_i)_{i\in I} \notin \hat{\mathcal{L}}(d_{*,*})$.

If $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$, then the system of equations $\{E^i\}_{i\in I}$ is satisfiable over the free group, and the algorithm can simply output an arbitrary equation $E = E^i$ from the system. Clearly, any solution to the system is also a solution to E. Moreover, the other condition in the theorem is vacuously satisfied because $\{E^i\}_{i\in I}$ is satisfiable over the free group.

So, let us assume that $\hat{\mathcal{L}}(e_{*,*}) \not\subseteq \hat{\mathcal{L}}(d_{*,*})$, and let $u_* = (u_i)_{i\in I}$ be a vector such that $(u_i)_{i\in I} \in \hat{\mathcal{L}}(e_{*,*}) \setminus \hat{\mathcal{L}}(d_{*,*})$. We know that $\sum_i u_i e_{i,x}$ is an integer for all $x \in X$ because u_* belongs to the dual lattice $\hat{\mathcal{L}}(e_{*,*})$. Moreover, since $\mathcal{L}(e_{*,*})$ is an integer lattice, all entries u_i are rational numbers. It follows that for any $a \in A$, $\sum_i u_i d_{i,a}$ is a rational number, but $\sum_i u_i d_{i,a}$ is not an integer for some

$a \in A$. Let M be the smallest integer such that $M \cdot \sum_i u_i d_{i,a}$ is an integer for all $a \in A$. In other words, let M be the least common multiple of the denominators of the fractions $\sum_i u_i d_{i,a}$ for all $a \in A$. The output of the algorithm is the equation

$$E\colon \prod_{x \in X} x^{M \cdot \sum_i u_i e_{i,x}} = \prod_{a \in A} a^{M \cdot \sum_i u_i d_{i,a}}.$$

We need to show that this equation satisfies the two properties in the theorem.

Let $\alpha\colon A \to G_N$ and $\xi\colon X \to G_N$ be two assignments such that ξ is a solution to the system $\{E_\alpha^i\}_{i \in I}$ over computational group G_N, i.e., $\prod_{x \in X} \xi(x)^{e_{i,x}} = \prod_{a \in A} \alpha(a)^{d_{i,a}}$ for all $i \in I$. It follows that

$$\begin{aligned} \xi\left(\prod_{x \in X} x^{M \cdot \sum_i u_i e_{i,x}}\right) &= \prod_{i \in I}\left(\prod_{x \in X} \xi(x)^{e_{i,x}}\right)^{M u_i} \\ &= \prod_{i \in I}\left(\prod_{a \in A} \alpha(a)^{d_{i,a}}\right)^{M u_i} \\ &= \alpha\left(\prod_{a \in A} a^{M \cdot \sum_i u_i d_{i,a}}\right), \end{aligned}$$

i.e., ξ is also a solution to equation E_α. This proves the second property. For the first property, since the system is unsatisfiable, we need to prove that E is also unsatisfiable over the free group $\mathcal{F}(A)$. Assume for contradiction that E is satisfiable over the free group and let $\sigma(x) = \prod_{a \in A} a^{s_{x,a}}$ be a solution, i.e.,

$$\prod_{x \in X}\left(\prod_{a \in A} a^{s_{x,a}}\right)^{M \cdot \sum_i u_i e_{i,x}} = \prod_{a \in A} a^{M \cdot \sum_i u_i d_{i,a}}.$$

Since the group $\mathcal{F}(A)$ is free, this is true if and only if

$$M \sum_{x \in X} s_{x,a} \sum_{i \in I} u_i e_{i,x} = M \cdot \sum_{i \in I} u_i d_{i,a}$$

for all $a \in A$. Since $\sum_{x \in X} s_{x,a} \sum_{i \in I} u_i e_{i,x}$ is an integer, the left hand side of the last equation is a multiple of M. So, the right hand side is also a multiple of M, and $\sum_i u_i d_{i,a}$ is an integer for all $a \in A$. But this is a contradiction because by construction (namely, by the choice of $(u_i)_{i \in I}$) there exists an $a \in A$ such that $\sum_i u_i d_{i,a}$ is not an integer. $\square$

Corollary 1. *A family of computational groups $\{G_N\}_{N \in \mathcal{N}}$ satisfies Definition 2 if and only if it satisfies Definition 3.*

Proof. If a group family is pseudo-free in the sense of Definition 3, then it satisfies Definition 2 as well because single equations are a special case of systems containing only one equation. Conversely, assume a group family does not satisfies Definition 3, i.e., there exists an adversary $\mathcal{A}$ that on input a group index

$N \in \mathcal{N}$ and random assignment $\alpha: A \to G_N$, outputs an unsatisfiable system of equations $\{E_i\}_{i\in I}$ over constants A and variables X, together with a solution $\xi: X \to G_N$ to the system over the computational group G_N. Then, using Theorem 4, $\mathcal{A}$ can be easily converted into an adversary $\mathcal{A}'$ contradicting Definition 2. Namely, on input group index $N \in \mathcal{N}$ and random assignment $\alpha : A \to G_N$, adversary $\mathcal{A}'$ invokes $\mathcal{A}$ on input (N, α) to get an unsatisfiable system of equations $\{E_i\}_{i\in I}$ together with a solution ξ over the computational group G_N. Finally, $\mathcal{A}'$ transforms $\{E_i\}_{i\in I}$ into a single equation E using Theorem 4, and outputs E, ξ. By Theorem 4, equation E is unsatisfiable over the free group, and ξ is a solution to E_α over G_N, proving that the group family does not satisfies Definition 2. $\square$

The following corollary immediately follows from Theorem 2 and 1.

Corollary 2. *Let $\mathcal{N}$ be a distribution over safe prime products such that the strong RSA problem modulo $N \in \mathcal{N}$ is hard. Then the family of computational groups $\mathbb{Z}_N^*$ of invertible integers modulo $N \in \mathcal{N}$ (with the modular multiplication group operation, and uniform sampling procedure over QR_N) satisfies Definition 3, i.e., it is pseudo-free with respect to systems of equations.*

5 Conclusion

We have given the first example of provably secure pseudo-free group under standard cryptographic assumptions. In particular, we proved that the RSA group $\mathbb{Z}_N^*$ where N is the product of two safe primes is pseudo-free, assuming the hardness of the strong RSA problem. Many open problems remain. In this section we illustrate some of them.

Our proof uses the fact that N is the product of two safe primes, and elements are sampled uniformly at random from the subgroup QR_N of quadratic residues. A natural question is whether $\mathbb{Z}_N^*$ is pseudo-free even when N is the product of two arbitrary primes, and elements are sampled uniformly at random from the whole group $\mathbb{Z}_N^*$. Another open problem is to relax the hypothesis of Theorem 2, and prove that $\mathbb{Z}_N^*$ is pseudo-free assuming that factoring N is hard. Notice that this last problem is probably very hard, as it would imply that inverting the RSA function is at least as hard as factoring, a long standing open problem in cryptography. However, there are many other cryptographic problems that have been proved at least as hard as factoring, like the discrete logarithm problem [2], the Diffie-Hellman problem [15], and the generalized Diffie-Hellman problem [5] modulo Blum integers. We remark that while computing discrete logarithms in pseudo-free groups is provably hard [20], no relation between pseudo-freeness and the Diffie-Hellman problem is currently known. An interesting open question, already posed in [20], is to show that the Diffie-Hellman problem in pseudo-free groups is computationally hard.

Another interesting problem is to find other examples of pseudo-free groups, beside $\mathbb{Z}_N^*$, and possibly proving their security based on standard cryptographic assumptions. Of particular interest would be to find a good candidate of non abelian pseudo-free group.

Finally, it would be nice to find applications of pseudo-free groups, as those mentioned in [20] and in the introduction, to demonstrate the usefulness of the notion of pseudo-free group. It might be the case that some applications require even stronger notions of pseudo-freeness than the one defined in [20]. In Section 4 we already considered extending the definition to systems of equations, and proved that pseudo-freeness with respect to systems of equations (Definition 3) is equivalent to the basic definition of pseudo-free group. Another possible extensions is to consider more general boolean combinations of equations, e.g., one can consider systems of equations $w_1 = w_2$ and inequations $w_1 \neq w_2$. For example, $x^2 = 1$ and $x \neq 1$ cannot be simultaneously satisfied over the free group, but admit a solution $x = N - 1$ in $\mathbb{Z}_N^*$ for any $N \neq 2$. We remark that the satisfiability problem over free abelian groups for arbitrary boolean combinations of equations is NP-hard. (E.g., 3SAT can be immediately reduced to such a formula mapping each boolean variable x to a corresponding equation $x = 1$.) So, some unsatisfiable formula do not have short (polynomial size) proofs of unsatisfiability, unless NP=coNP. Extensions of the notion of pseudo-free group to general boolean combinations of formulas should require the adversary to output not only an unsatisfiable formula over the free group (together with a solution over the computational group), but also a short and easily verifiable proof that the formula is indeed unsatisfiable.

6 Acknowledgments

Research supported in part by NSF grants 0093029, 0313241, 0430595 and a Sloan research fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
2. E. Bach. Discrete logarithms and factoring. Technical Report CSD-84-186, University of California at Berkeley, 1984.
3. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proceedings of the 10th ACM conference on computer and communications security - CCS 2003*, pages 220–230, Washington, DC, USA, Oct. 2003. ACM.
4. N. Baric' and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in cryptology - EUROCRYPT '97, Proceedings of the international conference on the theory and application of cryptographic techniques*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Konstanz, Germany, May 1997. Springer.
5. E. Biham, D. Boneh, and O. Reingold. Breaking generalized Diffie-Hellman modulo a composite is no easier than factoring. *Information Processing Letters*, 70(2):83–87, Apr. 1999.

6. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000. Preliminary version in CCS 1999.
7. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
8. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. S. Kaliski, Jr., editor, *Advances in cryptology - CRYPTO '97, Proceedings of the 17th annual international cryptology conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Santa Barbara, California, USA, Aug. 1997. Springer.
9. R. Gennaro, T. Rabin, and H. Krawczyk. RSA-based undeniable signatures. *Journal of Cryptology*, 13(4):397–416, 2000. Preliminary version in Crypto 1997.
10. S. Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology, EECS Dept., Cambridge, MA, June 2003.
11. R. Impagliazzo and B. Kapron. Logics for reasoning about cryptographic constructions. In *Proceedings of the 44rd annual symposium on foundations of computer science - FOCS 2003*, pages 372–383, Cambridge, MA, USA, Nov. 2003. IEEE.
12. O. Kharlampovich and A. Myasnikov. Implicit function theorem over free groups. *Journal of Algebra*, 2005. To appear.
13. P. D. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polytime framework for protocol analysis. In *Proceedings of the fifth ACM conference on computer and communications security - CCS '98*, pages 112–121, San Francisco, California, USA, Nov. 1998. ACM.
14. A. I. Mal'cev. On some correspondence between rings and groups. *Math. Sbornik*, 50:257–266, 1960.
15. K. S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, 1(2):95–105, 1988.
16. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.
17. D. Micciancio and S. Panjwani. Adaptive security of symbolic encryption. In J. Kilian, editor, *Theory of cryptography conference - Proceedings of TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187, Cambridge, MA, USA, Feb. 2005. Springer.
18. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.
19. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Theory of cryptography conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, Feb. 2004. Springer.
20. R. L. Rivest. On the notion of pseudo-free groups. In M. Naor, editor, *Theory of cryptography conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 505–521, Cambridge, MA, USA, Feb. 2003. Springer.
21. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.