J. Cryptology (1992) 5: 41–52

Journal of Cryptology

© 1992 International Association for Cryptologic Research

Which New RSA-Signatures Can Be Computed from Certain Given RSA-Signatures?[1]

Jan-Hendrik Evertse
Department of Mathematics and Computer Science, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

Eugène van Heyst
CWI Centre for Mathematics and Computer Science,
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Abstract. We consider the following problem. A signature authority issues RSA-signatures of certain types to an individual, and the individual tries, by using the signatures he received, to compute an RSA-signature of a type not issued by the authority. Is the individual able to do this? The RSA-signatures are products of rational powers of residue classes modulo the composite number N of the underlying RSA-system, and the residue classes are chosen at random by the signature authority. The rational exponents in the product determine the type of the signature.

We prove that computing an RSA-signature of a particular type, from given RSA-signatures of other types, is polynomial time reducible to computing RSA-roots $x^{1/d}$ (mod N) for random x and some positive integer d. This extends results of Akl and Taylor [1] and Shamir [11] from one variable to arbitrarily many variables. As an application of this, under the assumption that for the individual it is infeasible to compute RSA-roots, we give necessary and sufficient conditions describing whether it is feasible for that individual to compute RSA-signatures of a prescribed type from signatures of other types that he received before from the authority.

Key words. RSA, RSA-scheme, RSA-signature, Cryptographic protocol.

1. Introduction

Several more complicated cryptographic protocols use as a building block simple *signature protocols* in which only one party, called the *signature authority*, can create signatures and issues them to the other parties, called the *individuals*. Such protocols are used, for instance, in credential systems (e.g., [3]) and payment systems (e.g., [2]), in which a signature represents a credential or money. In fact, in such credential

[1] Date received: September 5, 1990. Date revised: April 20, 1991. A preliminary version of this paper was presented at Eurocrypt '90, May 21–24, Århus, Denmark, and has appeared in the proceedings, pp. 83–97. The research of Jan-Hendrik Evertse has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences (K.N.A.W.)

Signature authority $\mathscr{Z}$		Individual $\mathscr{A}$
chooses $x \pmod N$ randomly,	$\xrightarrow{x,S}$	verifies that
computes $S \equiv x^{e_1/e_2} \pmod N$		$S^{e_2} \equiv x^{e_1} \pmod N$

Fig. 1. A signature issuing protocol in which the signature authority $\mathscr{Z}$ issues a signature to individual $\mathscr{A}$.

systems or payment systems, the signature authority issues different types of signatures, corresponding to different credentials or different values of money. The security of these systems depends on whether an individual (or a group of conspiring individuals) is not able to compute a signature of a type not issued by the signature authority, by using the signatures which were issued before by the authority.

In Fig. 1, we give an example of a signature protocol, based on the RSA-system [9], in which the signature authority $\mathscr{Z}$ issues a signature to an individual $\mathscr{A}$. Initially, $\mathscr{Z}$ chooses two large primes P, Q and computes their product N. Further, $\mathscr{Z}$ chooses two integers e_1, e_2 with e_2 coprime to $\varphi(N) = (P-1)(Q-1)$. $\mathscr{Z}$ makes N, e_1, e_2 public, and keeps P and Q secret.

In general, for every integer b coprime to $\varphi(N)$, the congruence $y^b \equiv x \pmod N$ has a unique solution $\pmod N$ which we denote by $x^{1/b} \pmod N$. If $b > 1$, we call $x^{1/b} \pmod N$ an RSA-root mod N. For $a \in \mathbb{Z}$ we put $x^{a/b} \equiv (x^{1/b})^a \pmod N$. $\mathscr{Z}$ can easily compute the RSA-root $x^{1/b} \pmod N$ by computing first an integer b' with $bb' \equiv 1 \pmod{\varphi(N)}$, and then $x^{b'} \pmod N$. We assume that for individuals it is infeasible to compute RSA-roots $\pmod N$.

We consider a generalization of this protocol: $\mathscr{Z}$ chooses at random several residue classes $\pmod N$, computes a number of RSA-signatures which are products of rational powers of these residue classes modulo N, and issues these signatures to $\mathscr{A}$, together with the residue classes. The exponents in the product determine the type of signature. It will appear to be useful to consider also the variation in which $\mathscr{Z}$ sends only the signatures but not the residue classes to $\mathscr{A}$ (so that $\mathscr{A}$ cannot verify the signatures). $\mathscr{A}$ might also have received the signatures without the residue classes by eavesdropping. It is conceivable that an individual learns several RSA-signatures issued by $\mathscr{Z}$ (by participating in a signature issuing protocol or by eavesdropping) and that he uses these to compute useful signatures not issued by $\mathscr{Z}$.

We give an example of the kind of problems we are faced with. Suppose that an individual $\mathscr{A}$ received two randomly chosen residue classes $x_1, x_2 \pmod N$ and a signature $S \equiv x_1^{2/3} \cdot x_2^{1/9} \pmod N$, and that he wants to compute $S' \equiv x_2^{1/9} \pmod N$. $\mathscr{A}$ can easily compute $x_2^{1/3} \pmod N$, since $x_2^{1/3} \equiv x_1^{-2} S^3 \pmod N$. But then $\mathscr{A}$ has still to compute some cube RSA-root. From our Theorem 1, stated in Section 2, it follows that computing $x_2^{1/9}$ from $\{x_1, x_2, S\}$ is just as difficult as computing $x^{1/3} \pmod N$ for each residue class $x \pmod N$. So if $\mathscr{A}$ cannot compute RSA-roots, then he cannot compute $x_2^{1/9}$ from $\{x_1, x_2, S\}$.

Akl and Taylor [1] and Shamir [11] considered related problems. Shamir showed, roughly speaking, that for pairwise coprime integers $k_1, \ldots, k_t$ computing x^{1/k_1} from $\{x, x^{1/k_2}, \ldots, x^{1/k_t}\}$ is just as difficult as computing u^{1/k_1} from u alone, for random u.

Akl and Taylor proved, that if k, k_1, ..., k_t are integers with $\gcd(k_1, \dots, k_t)/\gcd(k, k_1, \dots, k_t) = r$, then computing x^k from $\{x^{k_1}, \dots, x^{k_t}\}$ (with x unknown) is at least as difficult as computing $u^{1/r}$ from u for random u. We generalize these results to arbitrarily many variables as in the example above. Our main result is stated in Section 2, independently of the context of protocols mentioned above. Let $S_1 \equiv \prod x_j^{a_{1,j}}$, ..., $S_s \equiv \prod x_j^{a_{s,j}}$, $S' \equiv \prod x_j^{b_j}$ (mod N), where the x_i are uniformly chosen residue classes (mod N) and the $a_{i,j}$, b_j are rational numbers (we are not precise here). Then computing S' from $S_1, \dots, S_s$ is polynomial time reducible to computing a certain RSA-root on random residue classes (mod N) and vice versa.

This paper is organized as follows. In Section 2 we state our results and prove these in Section 3. In Section 4 we discuss the consequences of our results for protocols as mentioned above; in particular, we consider two payment systems.

2. Statements of the Theorems

We first introduce some notation and terminology.

S^k	the set of vectors $(a_1, \dots, a_k)$ with $a_1, \dots, a_k \in S$, for any set S; we use boldface characters to denote vectors.
$\mathbf{a} \equiv \mathbf{b} \pmod{m}$	$m^{-1}(\mathbf{b} - \mathbf{a}) \in \mathbb{Z}^k$; this is defined for $\mathbf{a}, \mathbf{b} \in \mathbb{Q}^k$, $m, k \in \mathbb{N}$.
$\mathbb{Z}\{\mathbf{a}_1, \dots, \mathbf{a}_s\}$	$\{\sum_{i=1}^s \xi_i \mathbf{a}_i \mid \xi_1, \dots, \xi_s \in \mathbb{Z}\}$: the abelian group generated by $\mathbf{a}_1, \dots, \mathbf{a}_s \in \mathbb{Q}^k$.
$\mathbb{Q}\{\mathbf{a}_1, \dots, \mathbf{a}_s\}$	$\{\sum_{i=1}^s \xi_i \mathbf{a}_i \mid \xi_1, \dots, \xi_s \in \mathbb{Q}\}$: the vector space generated by $\mathbf{a}_1, \dots, \mathbf{a}_s \in \mathbb{Q}^k$.
$\langle \mathbf{a}, \mathbf{b} \rangle$	$a_1 b_1 + \cdots + a_k b_k$: the scalar product of $\mathbf{a} = (a_1, \dots, a_k)$ and $\mathbf{b} = (b_1, \dots, b_k)$.
N	a composite, odd number; so $N = p_1^{k_1} \cdots p_t^{k_t}$ with $p_1, \dots, p_t$ distinct odd primes and $k_1, \dots, k_t \in \mathbb{N}$.
$\mathbb{Z}_N^*$	the set $\{a \mid a \in \mathbb{N}, 1 \le a \le N, \gcd(a, N) = 1\}$.
$a^{-1} \pmod{N}$	the number $b \in \mathbb{Z}_N^*$ with $ab \equiv 1 \pmod{N}$; for $a \in \mathbb{Z}_N^*$.
$\varphi(N)$	Euler's Totient function: $\varphi(N) = \lvert\mathbb{Z}_N^*\rvert = \prod_{i=1}^t p_i^{k_i - 1}(p_i - 1)$.
$\tilde{\mathbb{Q}}_N$	the ring $\{a/d \mid a, d \in \mathbb{Z}, d > 0, \gcd(d, \varphi(N)) = 1\}$.
$\lambda(N)$	Carmichael's function: $\lambda(N) = \operatorname{lcm}(p_1^{k_1-1}(p_1 - 1), \dots, p_t^{k_t-1}(p_t - 1))$.
$x^{1/d}$	the dth *RSA-root* of x (mod N): the unique solution $S \in \mathbb{Z}_N^*$ to $S^d \equiv x \pmod{N}$ for $x \in \mathbb{Z}_N^*$ and $d \in \mathbb{Z}$ with $\gcd(d, \varphi(N)) = 1$.
$\mathbf{x}^{\mathbf{a}} \pmod{N}$	the number $S \in \mathbb{Z}_N^*$ with $S \equiv x_1^{a_1} x_2^{a_2} \cdots x_k^{a_k} \pmod{N}$, for $\mathbf{x} = (x_1, \dots, x_k) \in (\mathbb{Z}_N^*)^k$ and $\mathbf{a} = (a_1, \dots, a_k) \in (\tilde{\mathbb{Q}}_N)^k$.

In this paper we use notions like *deterministic* and *probabilistic algorithms* which can be given a precise mathematical meaning, for instance, using deterministic and probabilistic Turing machines, see [4]. The only nondeterministic operations allowed in a probabilistic algorithm are unbiased coin tosses. In the algorithms we consider, the *inputs* are tuples of integers and rationals, and the *length* of such an input is the sum of the lengths of the binary representations of the integers and the numerators and denominators of the rational numbers in the input. In general, both the output and the running time of a probabilistic algorithm are stochastic variables depending on the input and the random coin tosses. However, in this paper we consider only probabilistic algorithms whose running time is determined by only the input. Thus, if a probabilistic algorithm is used to solve a particular problem, then it may output a solution not with certainty but only with some probability of success. Generally, the underlying probability space consists of the strings of bits, chosen during the execution of the algorithm, with uniform distribution, and a set J of possible inputs, from which input I is chosen with probability p_I. Thus, if some algorithm solves a problem with conditional probability of success s_I given input I, then its unconditional probability of success is $\sum_{I \in J} p_I s_I$. By a *polynomial time algorithm* we mean a deterministic algorithm whose running time depends polynomially on the length of the input.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$. We consider the problem of computing $\mathbf{x}^{\mathbf{b}}$ for random $\mathbf{x} \in (\mathbb{Z}_N^*)^k$, if $\{\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}\}$ (but not necessarily $\mathbf{x}$) are given as inputs. We distinguish two cases, to each of which a theorem is devoted.

In Theorem 1 below, $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}$ are vectors in $(\tilde{\mathbb{Q}}_N)^k$, satisfying

$$\left.\begin{array}{l} \mathbf{b} \in \mathbb{Q}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\};\ \text{length}(N, \mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}) = L; \\ \gcd(d, \varphi(N)) = 1, \text{ where } d = \min\{x \in \mathbb{N} \mid x\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}\}. \end{array}\right\} \qquad (1)$$

Theorem 1. *Let* $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}$ *satisfy* (1).

(i) *For every probabilistic algorithm* AL *that with input* $\{N, \mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}, \mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}\}$ *computes* $\mathbf{x}^{\mathbf{b}}$ *in time* $\le T_{\mathrm{AL}}$ *with probability of success* $\ge \varepsilon_{\mathrm{AL}}$ *for* random $\mathbf{x} \in (\mathbb{Z}_N^*)^k$, *there exists a probabilistic algorithm* $\overline{\mathrm{AL}}$ *that for* arbitrary $u \in \mathbb{Z}_N^*$ *computes* $u^{1/d}$ *in time* $\le T_{\mathrm{AL}} + L^{O(1)}$ *with probability of success* $\ge \frac{1}{2}\varepsilon_{\mathrm{AL}}$.

(ii) *For every probabilistic algorithm* AL *that with input* $\{N, u\}$ *computes* $u^{1/d}$ *in time* $\le T_{\mathrm{AL}}$ *with probability of success* $\ge \varepsilon_{\mathrm{AL}}$ *for* random $u \in \mathbb{Z}_N^*$, *there exists a probabilistic algorithm* $\overline{\mathrm{AL}}$ *that for* arbitrary $\mathbf{x} \in (\mathbb{Z}_N^*)^k$ *computes* $\mathbf{x}^{\mathbf{b}}$ *from* $\{N, \mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}, \mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}\}$ *in time* $\le T_{\mathrm{AL}} + L^{O(1)}$ *with probability of success* $\ge \frac{1}{2}\varepsilon_{\mathrm{AL}}$.

Remark 1. Theorem 1 can be generalized to the case that $\gcd(d, \varphi(N)) > 1$. Let G_d be the largest subgroup of $\mathbb{Z}_N^*$ whose order is coprime to d. Then for every $u \in G_d$, there is a unique $x \in G_d$ with $x^d \equiv u \pmod{N}$ and we denote this x by $u^{1/d}$. We can prove that for every probabilistic algorithm AL as in Theorem 1 there is a probabilistic algorithm $\overline{\mathrm{AL}}$ that for arbitrary $u \in G_d$ computes $u^{1/d}$ in time $\le T_{\mathrm{AL}} + L^{O(1)}$ with probability of success $\ge \frac{1}{2}\varepsilon_{\mathrm{AL}}$. As the proof of this generalization is precisely the same as that of Theorem 1, we do not work it out.

Remark 2. Theorem 1 deals with the situation that $\mathbf{x}^{\mathbf{b}}$ has to be computed from $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$ while $\mathbf{x}$ itself is not known. We can treat the case that $\mathbf{x}^{\mathbf{b}}$ has to be computed from $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$ *and* $\mathbf{x}$, by applying Theorem 1 with $\mathbf{a}_1, \ldots, \mathbf{a}_s$, $\mathbf{e}_1 = (1, 0, \ldots, 0), \mathbf{e}_2 = (0, 1, \ldots, 0), \ldots, \mathbf{e}_k = (0, \ldots, 0, 1)$, instead of $\mathbf{a}_1, \ldots, \mathbf{a}_s$. Note that $\mathbf{b} \in \mathbb{Q}\{\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{e}_1, \ldots, \mathbf{e}_k\}$ for all $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$. Further, if d is the smallest positive integer x with $x\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{e}_1, \ldots, \mathbf{e}_k\}$, then d is the gcd of all these integers x. Hence if $\mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$, then $\gcd(d, \varphi(N)) = 1$.

We can also treat the case in which $\mathbf{x}_1^{\mathbf{b}_1} \cdots \mathbf{x}_s^{\mathbf{b}_s}$ has to be computed from $\mathbf{x}_1^{\mathbf{a}_1}, \ldots, \mathbf{x}_s^{\mathbf{a}_s}$ for certain $\mathbf{b}_1, \ldots, \mathbf{b}_s \in (\tilde{\mathbb{Q}}_N)^k$, where $\mathbf{x}_1, \ldots, \mathbf{x}_s$ are distinct vectors from $(\mathbb{Z}_N^*)^k$:

namely, put $\mathbf{x}' := (\mathbf{x}_1, \dots, \mathbf{x}_s)$, $\mathbf{a}_1' := (\mathbf{a}_1, \mathbf{0}, \dots, \mathbf{0})$, $\mathbf{a}_2' := (\mathbf{0}, \mathbf{a}_2, \mathbf{0}, \dots, \mathbf{0})$, $\dots$, $\mathbf{a}_s' := (\mathbf{0}, \dots, \mathbf{0}, \mathbf{a}_s)$, $\mathbf{b}' := (\mathbf{b}_1, \dots, \mathbf{b}_s)$, and apply Theorem 1 with $\mathbf{x}', \mathbf{a}_1', \dots, \mathbf{a}_s', \mathbf{b}'$.

In Theorem 2 below, $\mathbf{a}_1, \dots, \mathbf{a}_s, \mathbf{b}$ are vectors in $(\tilde{\mathbb{Q}}_N)^k$, satisfying

$$\left.\begin{aligned} &\mathbf{b} \notin \mathbb{Q}\{\mathbf{a}_1, \dots, \mathbf{a}_s\};\ \mathrm{length}(N, \mathbf{a}_1, \dots, \mathbf{a}_s, \mathbf{b}) = L; \\ &d = \min\{x \in \mathbb{N} \mid \exists \xi_1, \dots, \xi_s \in \mathbb{Z}\colon x\mathbf{b} \equiv \textstyle\sum_{i=1}^s \xi_i \mathbf{a}_i \pmod{\lambda(N)}\}. \end{aligned}\right\} \tag{2}$$

Note that d is the gcd of all the integers x as in (2). Hence d divides $\lambda(N)$. We have:

Theorem 2. *Let $\mathbf{a}_1, \dots, \mathbf{a}_s, \mathbf{b}$ satisfy* (2).

(i) *There exists a polynomial (in L) time algorithm that computes a nonzero multiple of $\lambda(N)/d$ from $\mathbf{a}_1, \dots, \mathbf{a}_s, \mathbf{b}$.*
(ii) *For every $\mathbf{x} \in (\mathbb{Z}_N^*)^k$, the cardinality of the set $\{z \in \mathbb{Z}_N^* \mid \exists \mathbf{y} \in (\mathbb{Z}_N^*)^k\colon \mathbf{y}^{\mathbf{b}} \equiv z \pmod N, \mathbf{y}^{\mathbf{a}_i} \equiv \mathbf{x}^{\mathbf{a}_i} \pmod N$ for $i = 1, \dots, s\}$ is equal to the number of solutions $z \in \mathbb{Z}_N^*$ of $z^d \equiv 1 \pmod N$.*

For instance, if $d = 1$ then from $\mathbf{a}_1, \dots, \mathbf{a}_s, \mathbf{b}$ we can compute in polynomial (in L) time a multiple of $\lambda(N)$ and from that we can compute in probabilistic polynomial (in length (N)) time the factorization of N [7]. In the other extreme situation that $d = \lambda(N)$, if $\mathbf{x}^{\mathbf{a}_1}, \dots, \mathbf{x}^{\mathbf{a}_s}$ are given but $\mathbf{x}$ is unknown, then every number in $\mathbb{Z}_N^*$ is possible for $\mathbf{x}^{\mathbf{b}}$.

3. Proofs

We need some lemmas to prove Theorems 1 and 2.

Lemma 1. *There is a polynomial time algorithm that computes for every $\mathbf{a}_1, \dots, \mathbf{a}_s \in \mathbb{Q}^k$ a basis $\{\mathbf{e}_1, \dots, \mathbf{e}_k\}$ of $\mathbb{Z}^k$ and $d_1, \dots, d_t \in \mathbb{Q}_{>0}$ such that $\{d_1\mathbf{e}_1, \dots, d_t\mathbf{e}_t\}$ is a basis of $\mathbb{Z}\{\mathbf{a}_1, \dots, \mathbf{a}_s\}$.*

Proof. For $\mathbf{a}_1, \dots, \mathbf{a}_s \in \mathbb{Z}^k$ this follows from the result of Kannan and Bachem [6] that we can compute in polynomial time the Smith normal form of an integral matrix. For $\mathbf{a}_1, \dots, \mathbf{a}_s \in \mathbb{Q}^k$, one may first compute $d \in \mathbb{N}$ such that $d\mathbf{a}_1, \dots, d\mathbf{a}_s \in \mathbb{Z}^k$ and then apply the result of Kannan and Bachem. □

Lemma 2. *For $a, b \in \mathbb{Z}$, $a, b \neq 0$, let $(a\backslash b)$ denote the largest positive divisor of a which is not divisible by any prime number dividing b. There is a polynomial time algorithm that computes $(a\backslash b)$ from $a, b \in \mathbb{Z}$, $a, b \neq 0$.*

Proof. Consider the sequence of integers $c_0 = |a|$, $c_1 = c_0/\gcd(c_0, b)$, $c_2 = c_1/\gcd(c_1, b)$, ... There is an i such that $\gcd(c_i, b) = 1$; let i_0 be the smallest such i. Since $c_1 \le c_0/2$, $c_2 \le c_1/2, \dots, c_{i_0} \le c_{i_0-1}/2$, $c_{i_0+1} = c_{i_0}$, we have $i_0 \le \log|a|/\log 2$. Hence it takes polynomial time to compute c_{i_0}. For each prime number p and each $a \in \mathbb{Z}$, $a \neq 0$, let $\mathrm{ord}_p(a)$ be the integer such that $a \cdot p^{-\mathrm{ord}_p(a)}$ is an integer not divisible by p. Obviously, $\mathrm{ord}_p(c_{i_0}) = 0$ for each prime p dividing b. Further, if p is a prime dividing a but not b, then $\mathrm{ord}_p(a) = \mathrm{ord}_p(c_0) = \mathrm{ord}_p(c_1) = \cdots = \mathrm{ord}_p(c_{i_0})$. It follows that $c_{i_0} = (a \backslash b)$. □

Lemma 3. *Let $\mathbf{a}_1, \ldots, \mathbf{a}_s \in \mathbb{Q}^k$, $\mathbf{b} \in \mathbb{Q}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$, and let d be the smallest positive integer such that $d\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$. Then there is a vector $\mathbf{r} \in \mathbb{Q}^k$ such that the denominators of the coordinates of $\mathbf{r}$ are composed of prime numbers dividing d and such that*

$$\langle \mathbf{a}_i, \mathbf{r} \rangle \in \mathbb{Z} \quad \textit{for } i = 1, \ldots, t, \qquad \langle \mathbf{b}, \mathbf{r} \rangle - \frac{1}{d} \in \mathbb{Z}. \tag{3}$$

Further, there is a polynomial time algorithm that computes d and such a vector $\mathbf{r}$ from $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}$.

Proof. Compute a basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ *of* $\mathbb{Z}^k$ and $d_1, \ldots, d_t \in \mathbb{Q}_{>0}$ as in Lemma 1 from $\mathbf{a}_1, \ldots, \mathbf{a}_s$. Further, compute $\xi_1, \ldots, \xi_t \in \mathbb{Q}$ with $\mathbf{b} = \sum_{i=1}^{t} \xi_i d_i \mathbf{e}_i$, e.g., by Gauss elimination. Then d is the smallest positive integer such that $d\xi_1, \ldots, d\xi_t \in \mathbb{Z}$ and so it can be computed in polynomial time. Let $u_1, \ldots, u_t$ be the numerators of $d_1, \ldots, d_t$, respectively. Compute $(u_1 \backslash d), \ldots, (u_t \backslash d)$. Note that $\gcd(d, \xi_1 d(u_1 \backslash d), \ldots, \xi_t d(u_t \backslash d)) = 1$. Now compute $s_1, \ldots, s_t \in \mathbb{Z}$ satisfying

$$\sum_{i=1}^{t} \xi_i d(u_i \backslash d) s_i \equiv 1 \pmod{d}, \tag{4}$$

with Euclid's algorithm. Finally, compute $\mathbf{r} \in \mathbb{Q}^k$, e.g., by Gauss elimination, with

$$\langle \mathbf{e}_i, \mathbf{r} \rangle = \begin{cases} \dfrac{s_i(u_i \backslash d)}{d_i} & \text{for } i = 1, \ldots, t. \\ 0 & \text{for } i = t+1, \ldots, k. \end{cases} \tag{5}$$

Note that the denominators of $s_i(u_i \backslash d)/d_i$ are composed of primes dividing d for $i = 1, \ldots, t$. Since $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ is a basis of $\mathbb{Z}^k$, this implies that the denominators of the coordinates of $\mathbf{r}$ are also composed of primes dividing d. Further, from (5) and $\mathbf{a}_i \in \mathbb{Z}\{d_1\mathbf{e}_1, \ldots, d_t\mathbf{e}_t\}$ it follows that $\langle \mathbf{a}_i, \mathbf{r} \rangle \in \mathbb{Z}$ for $i = 1, \ldots, t$. Finally, from (4), (5), and $\mathbf{b} = \sum_{i=1}^{t} \xi_i d_i \mathbf{e}_i$, it follows that $\langle \mathbf{b}, \mathbf{r} \rangle - 1/d \in \mathbb{Z}$. It is easy to verify that all the computations mentioned above cost polynomial time. This proves Lemma 3. □

Proof of Theorem 2. We assume that $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in \mathbb{Z}^k$ which is no restriction. Indeed, if $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$, then we can compute in polynomial (in L) time $m \in \mathbb{N}$ with $\gcd(m, \lambda(N)) = 1$ such that $\mathbf{a}'_i := m\mathbf{a}_i$ $(i = 1, \ldots, s)$, $\mathbf{b}' := m\mathbf{b} \in \mathbb{Z}^k$, and we can proceed further with $\mathbf{a}'_1, \ldots, \mathbf{a}'_s, \mathbf{b}'$. The integer d is also the smallest positive integer x for which $x\mathbf{b}' \equiv \sum_{i=1}^{s} \xi_i \mathbf{a}'_i \pmod{\lambda(N)}$ is solvable in $\xi_1, \ldots, \xi_t \in \mathbb{Z}$ and $x \mapsto x^m$ is 1–1 on $\mathbb{Z}_N^*$. Hence $\{z \in \mathbb{Z}_N^* | \exists \mathbf{y} \in (\mathbb{Z}_N^*)^k\colon \mathbf{y}^{\mathbf{b}'} \equiv z, \mathbf{y}^{\mathbf{a}'_i} \equiv \mathbf{x}^{\mathbf{a}'_i} \pmod{N} \text{ for } i = 1, \ldots, s\}$ has the same cardinality as $\{z \in \mathbb{Z}_N^* | \exists \mathbf{y} \in (\mathbb{Z}_N^*)^k\colon \mathbf{y}^{\mathbf{b}} \equiv z, \mathbf{y}^{\mathbf{a}_i} \equiv \mathbf{x}^{\mathbf{a}_i} \pmod{N} \text{ for } i = 1, \ldots, s\}$.

(i) Compute a basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ of $\mathbb{Z}^k$ and $d_1, \ldots, d_t$ (which are now positive integers) such that $\{d_1\mathbf{e}_1, \ldots, d_t\mathbf{e}_t\}$ is a basis of $\mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$. Further, compute integers $\beta_1, \ldots, \beta_k$ such that $\mathbf{b} = \sum_{i=1}^{k} \beta_i \mathbf{e}_i$. Since $\mathbf{b} \notin \mathbb{Q}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$, at least one of the integers $\beta_{t+1}, \ldots, \beta_k$, β_{t+1} say, is nonzero. There are integers $\eta_1, \ldots, \eta_t$ such that

$d\mathbf{b} \equiv \sum_{i=1}^{t} \eta_i d_i \mathbf{e}_i \pmod{\lambda(N)}$. This implies that $d\beta_{t+1} \equiv 0 \pmod{\lambda(N)}$. Hence β_{t+1} is a nonzero multiple of $\lambda(N)/d$. All operations mentioned above can be done in polynomial (in L) time and so β_{t+1} can be computed in polynomial (in L) time. This proves (i).

(ii) Let $S_1 = \{z \in \mathbb{Z}_N^* | \exists \mathbf{y} \in (\mathbb{Z}_N^*)^k \colon \mathbf{y}^{\mathbf{b}} \equiv z, \mathbf{y}^{\mathbf{a}_i} \equiv 1 \pmod{N} \text{ for } i = 1, \ldots, s\}$. Then $\{z \in \mathbb{Z}_N^* | \exists \mathbf{y} \in (\mathbb{Z}_N^*)^k \colon \mathbf{y}^{\mathbf{b}} \equiv z, \mathbf{y}^{\mathbf{a}_i} \equiv \mathbf{x}^{\mathbf{a}_i} \pmod{N} \text{ for } i = 1, \ldots, s\} = \{z \cdot \mathbf{x}^{\mathbf{b}} | z \in S_1\}$. Hence it suffices to show, that S_1 is equal to $S_2 := \{z \in \mathbb{Z}_N^* | z^d \equiv 1 \pmod{N}\}$.

First take $z \in S_1$. There are $\xi_1, \ldots, \xi_t \in \mathbb{Z}$ such that $d\mathbf{b} \equiv \sum_{i=1}^{s} \xi_i \mathbf{a}_i \pmod{\lambda(N)}$. Together with the fact that $a^{\lambda(N)} \equiv 1 \pmod{N}$ for every $a \in \mathbb{Z}_N^*$, this implies that for some $\mathbf{y} \in (\mathbb{Z}_N^*)^k$: $z^d \equiv \mathbf{y}^{d\mathbf{b}} \equiv \prod_{i=1}^{s} (\mathbf{y}^{\mathbf{a}_i})^{\xi_i} \equiv 1 \pmod{N}$. Hence $z \in S_2$. It follows that $S_1 \subseteq S_2$.

Now take $z \in S_2$. We can factor N as $N = p_1^{k_1} \cdots p_t^{k_t}$ where $p_1, \ldots, p_t$ are odd primes and $k_1, \ldots, k_t \in \mathbb{N}$. Put $\delta_i = \gcd(d, \varphi(p_i^{k_i}))$ for $i = 1, \ldots, t$. Then δ_i is the smallest positive integer x such that $x\mathbf{b} \equiv \sum_{i=1}^{s} \xi_i \mathbf{a}_i \pmod{\varphi_i}$ has a solution in $\xi_1, \ldots, \xi_s \in \mathbb{Z}$, where $\varphi_i = p_i^{k_i - 1}(p_i - 1)$, i.e., the smallest positive integer x for which $x\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s, \varphi_i \mathbf{e}_1, \ldots, \varphi_i \mathbf{e}_k\}$, where $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ is any basis of $\mathbb{Z}^k$. By Lemma 3, there is a vector $\mathbf{r}$, with

$$\begin{aligned} &\langle \mathbf{a}_j, \mathbf{r} \rangle \in \mathbb{Z} &&\text{for} \quad j = 1, \ldots, s, \\ &\langle \varphi_i \mathbf{e}_j, \mathbf{r} \rangle \in \mathbb{Z} &&\text{for} \quad j = 1, \ldots, k, \\ &\langle \mathbf{b}, \mathbf{r} \rangle - \frac{1}{\delta_i} \in \mathbb{Z}. \end{aligned}$$

Put $\mathbf{v} := \varphi_i \mathbf{r}$. Then $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{Z}^k$ and

$$\left.\begin{aligned} &\langle \mathbf{a}_j, \mathbf{v} \rangle \equiv 0 \pmod{\varphi_i} \qquad \text{for} \quad j = 1, \ldots, s, \\ &\langle \mathbf{b}, \mathbf{v} \rangle \equiv \frac{\varphi_i}{\delta_i} \pmod{\varphi_i}. \end{aligned}\right\} \tag{6}$$

Since $z \in S_2$, we have $z^{\delta_i} \equiv 1 \pmod{p_i^{k_i}}$. Further, the group of residue classes mod $p_i^{k_i}$ coprime to p_i is cyclic of order φ_i. Hence there is a residue class $w_i \pmod{p_i^{k_i}}$ with $w_i^{\varphi_i/\delta_i} \equiv z \pmod{p_i^{k_i}}$. Put $\mathbf{y}_i = (w_i^{v_1}, \ldots, w_i^{v_k})$. Then (6) implies that $\mathbf{y}_i^{\mathbf{a}_j} \equiv w_i^{\langle \mathbf{a}_j, \mathbf{v} \rangle} \equiv 1 \pmod{p_i^{k_i}}$ for $j = 1, \ldots, s$ and $\mathbf{y}_i^{\mathbf{b}} \equiv w_i^{\langle \mathbf{b}, \mathbf{v} \rangle} \equiv w_i^{\varphi_i/\delta_i} \equiv z \pmod{p_i^{k_i}}$. By the Chinese remainder theorem, there is a $\mathbf{y} \in (\mathbb{Z}_N^*)^k$ with $\mathbf{y} \equiv \mathbf{y}_i \pmod{p_i^{k_i}}$ for $i = 1, \ldots, t$. This $\mathbf{y}$ satisfies $\mathbf{y}^{\mathbf{a}_j} \equiv 1 \pmod{N}$ for $j = 1, \ldots, s$, and $\mathbf{y}^{\mathbf{b}} \equiv z \pmod{N}$. Hence $z \in S_1$. We conclude that also $S_2 \subseteq S_1$. Therefore $S_2 = S_1$ and part (ii) of Theorem 2 has been proved. □

Proof of Theorem 1. Assume we are given N and $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$ satisfying (1).

(i) Assume there is a probabilistic algorithm AL which from $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$ computes $\mathbf{x}^{\mathbf{b}}$ in time $\leq T_{\mathrm{AL}}(L)$ with probability of success $\geq \varepsilon_{\mathrm{AL}}(L)$ for randomly chosen $\mathbf{x} \in (\mathbb{Z}_N^*)^k$. Fix $u \in \mathbb{Z}_N^*$. We describe a probabilistic algorithm $\overline{\mathrm{AL}}$ to compute $u^{1/d}$. The idea is to apply AL to the vector $\mathbf{u} = (u^{t_1}, \ldots, u^{t_k})$ for an appropriate vector $\mathbf{t} \in \tilde{\mathbb{Q}}_N$. However, this $\mathbf{u}$ is not a random vector in $(\mathbb{Z}_N^*)^k$, all its coordinates being a power of the same residue class, and so we do not know anything about the probability of success when AL is applied to $\mathbf{u}$. We use the well-known trick of applying AL instead to a vector of the form $\mathbf{x} = (u^{t_1} r_1^m, \ldots, u^{t_k} r_k^m)$, where

$\mathbf{r} = (r_1, \ldots, r_k)$ is randomly chosen from $(\mathbb{Z}_N^*)^k$ and m is such that $m\mathbf{a}_1, \ldots, m\mathbf{a}_s$, $m\mathbf{b} \in (\mathbb{Z}_N^*)^k$. Since $\mathbb{Z}_N^*$ is a multiplicative group and the mapping $x \to x^m$ on $\mathbb{Z}_N^*$ is 1–1 in view of $\gcd(m, \varphi(N)) = 1$, this vector $\mathbf{x}$ is uniformly distributed on $(\mathbb{Z}_N^*)^k$.

Below we describe the algorithm $\overline{\mathrm{AL}}$ (all congruences are mod N):

Step 1. Compute $\mathbf{t} = (t_1, \ldots, t_k) \in \mathbb{Q}^k$ and $\alpha_1, \ldots, \alpha_k, \beta \in \mathbb{Z}$ such that $\langle \mathbf{a}_i, \mathbf{t}\rangle = \alpha_i$ for $i = 1, \ldots, s$, $\langle \mathbf{b}, \mathbf{t}\rangle + \beta = 1/d$ and the denominators of $t_1, \ldots, t_k$ are composed of primes dividing d. Since $\gcd(d, \varphi(N)) = 1$, we have $\mathbf{t} \in (\tilde{\mathbb{Q}}_N)^k$. Compute m such that $m\mathbf{a}_i$ $(i = 1, \ldots, s)$, $m\mathbf{b} \in \mathbb{Z}^k$.
Step 2. Choose $\mathbf{r} = (r_1, \ldots, r_k)$ from $(\mathbb{Z}_N^*)^k$.
Step 3. Compute $u^{\alpha_i}\mathbf{r}^{m\mathbf{a}_i} \equiv u^{\langle \mathbf{a}_i, \mathbf{t}\rangle} \cdot \mathbf{r}^{m\mathbf{a}_i} \equiv \mathbf{x}^{\mathbf{a}_i}$ for $i = 1, \ldots, s$, where $\mathbf{x} \equiv (u^{t_1}r_1^m, \ldots, u^{t_k}r_k^m)$. This computation is easy since $\alpha_i \in \mathbb{Z}$, $m\mathbf{a}_i \in \mathbb{Z}^k$ for $i = 1, \ldots, s$. We remark that it need not be feasible to compute $\mathbf{x}$.
Step 4. Apply AL to $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$.
Step 5. If AL outputs $\mathbf{x}^{\mathbf{b}}$, then compute $\mathbf{x}^{\mathbf{b}}\mathbf{r}^{-m\mathbf{b}}u^{\beta} \equiv u^{\langle \mathbf{b}, \mathbf{t}\rangle + \beta} \equiv u^{1/d}$. This is possible since $\beta \in \mathbb{Z}$ and $m\mathbf{b} \in \mathbb{Z}^k$.

We did not yet specify the way $\mathbf{r}$ is chosen from $(\mathbb{Z}_N^*)^k$ in Step 2. There is no known polynomial time method to simulate a perfect uniform choice from $\mathbb{Z}_N^*$ with a probabilistic algorithm whose only possible nondeterministic operations are coin tosses. But we can proceed as follows. Compute the integer K with $N \le 2^K < 2N$. Choose r at random from $\{1, \ldots, 2^K\}$ by doing K coin tosses. Check if $r \in \{1, \ldots, N-1\}$ and $\gcd(r, N) = 1$. If so, take the residue class $r \pmod N$. Thus, we get an element of $\mathbb{Z}_N^*$ with probability of success $\varphi(N)/2^K \ge 1/(12 \log\log N)$, in view of the inequality $\varphi(N) \ge N/(6 \log\log N)$ for $N > 3$ [10]. There is a constant $c > 0$ with $1 - (1 - 1/(12 \log\log N))^{c \log k \cdot \log\log N} \ge (\frac{1}{2})^{1/k}$. Hence, after at most $ck \log k \cdot K \log\log N$ coin tosses we find a vector $\mathbf{r} \in (\mathbb{Z}_N^*)^k$ with probability of success $\ge \frac{1}{2}$. Moreover, the conditional probability distribution of $\mathbf{r}$ given success is uniform on $(\mathbb{Z}_N^*)^k$.

Steps 1, 2, 3, and 5 of algorithm $\overline{\mathrm{AL}}$ described above have running time $L^{O(1)}$. Further, Step 4 has running time T_{AL}. Hence the running time of $\overline{\mathrm{AL}}$ is $\le T_{\mathrm{AL}} + L^{O(1)}$. Step 2 has probability of success $\ge \frac{1}{2}$. Given success in Step 2, the conditional probability distribution of $\mathbf{x}$ is uniform on $(\mathbb{Z}_N^*)^k$ and hence the conditional probability of success in Step 4 is $\ge \varepsilon_{\mathrm{AL}}$. Therefore, the unconditional probability of success of $\overline{\mathrm{AL}}$ is $\ge \frac{1}{2}\varepsilon_{\mathrm{AL}}$. This proves (i).

(ii) Assume we are given a probabilistic algorithm AL which from randomly chosen $u \in \mathbb{Z}_N^*$ computes $u^{1/d}$ in time $\le T_{\mathrm{AL}}$ and probability of success $\ge \varepsilon_{\mathrm{AL}}(L)$. We construct the following algorithm $\overline{\mathrm{AL}}$ (the congruences are mod N):

Step 1. Compute $\xi_1, \ldots, \xi_s \in \mathbb{Z}$ such that $d\mathbf{b} = \sum \xi_i \mathbf{a}_i$.
Step 2. Choose $r \in \mathbb{Z}_N^*$.
Step 3. Compute $u \equiv r^d \cdot \prod_i (\mathbf{x}^{\mathbf{a}_i})^{\xi_i}$.
Step 4. Apply AL to u.
Step 5. If AL outputs $u^{1/d}$, then compute $r^{-1}u^{1/d} \equiv \prod_i (\mathbf{x}^{\mathbf{a}_i})^{\xi_i/d} \equiv (\mathbf{x}^{d\mathbf{b}})^{1/d} \equiv \mathbf{x}^{\mathbf{b}}$.

If we choose r in Step 2 as described above, then with essentially the same argument as above, it follows that $\overline{\mathrm{AL}}$ has running time $\le T_{\mathrm{AL}} + L^{O(1)}$ and probability of success $\ge \frac{1}{2}\varepsilon_{\mathrm{AL}}$ for arbitrary $\mathbf{x} \in (\mathbb{Z}_N^*)^k$. This proves (ii). □

4. Some Practical Applications

We go back to the protocols of Section 1. Let N be a composite modulus and let $\mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b} \in (\tilde{\mathbb{Q}}_N)^k$. Signature authority $\mathscr{Z}$ chooses at random $\mathbf{x} \in (\mathbb{Z}_N^*)^k$ and issues the signatures $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$ to the individual $\mathscr{A}$. $\mathscr{A}$ wants to compute $\mathbf{x}^{\mathbf{b}}$. Is he able to do this? Of course, we assume that for $\mathscr{A}$ it is infeasible to compute RSA-roots modulo N, since otherwise he could forge all signatures. Theorem 1 implies the following.

Corollary. *Assume there is an integer d with $(d, \varphi(N)) = 1$ and $d\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$. Then it is feasible for $\mathscr{A}$ to compute $\mathbf{x}^{\mathbf{b}}$ from $\{N, \mathbf{a}_1, \ldots, \mathbf{a}_s, \mathbf{b}, \mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}\}$ for uniformly chosen $\mathbf{x} \in (\mathbb{Z}_N^*)^k$ if and only if $\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$.*

(We do not want to make precise the notion of "computational infeasibility.") If $\mathbf{b} \in \mathbb{Z}\{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$, then $\mathbf{x}^{\mathbf{b}}$ can be computed from $\mathbf{x}^{\mathbf{a}_1}, \ldots, \mathbf{x}^{\mathbf{a}_s}$ simply by multiplying and dividing (mod N): if $\xi_1, \ldots, \xi_s \in \mathbb{Z}$ are such that $\mathbf{b} = \xi_1\mathbf{a}_1 + \cdots + \xi_s\mathbf{a}_s$, then $\mathbf{x}^{\mathbf{b}} \equiv \prod_{i=1}^s (\mathbf{x}^{\mathbf{a}_i})^{\xi_i}$ (mod N). Hence the corollary means that $\mathscr{A}$ cannot compute RSA-signatures from other ones, unless he is able to do this using only the obvious operations on RSA-signatures: multiplying and dividing (mod N). This corollary can also be used in situations where $\mathscr{Z}$ issues also $\mathbf{x}$ or in which $\mathscr{A}$ receives signatures $\mathbf{x}_1^{\mathbf{a}_1}, \ldots, \mathbf{x}_s^{\mathbf{a}_s}$ on distinct vectors $\mathbf{x}_1, \ldots, \mathbf{x}_s$ (see Remark 2 of Section 2).

We now give two examples related to coin systems to illustrate the corollary.

Example 1. In [2], a user-anonymous off-line check system is introduced. Here we discuss a special attack by the user on this system. We consider only a simplified version of the check system (not providing user anonymity) which, however, makes no difference for the attack we discuss.

The bank chooses a composite modulus N and two one-way functions f, g and makes N, f, g public. Assume the user wants a check of \$1023 = \$($2^0 + 2^1 + \cdots + 2^9$) from the bank. To this end, he chooses numbers b_i, r_i, s_i and computes $M_i \equiv f(g(b_i, r_i))$, $m_i \equiv f(g(b_i, s_i))$, $\alpha_i \equiv M_i^{3^{10}} \cdot m_i^{17}$ $(i = 0, \ldots, 9)$ (all congruences are mod N), and sends $\alpha_1, \ldots, \alpha_{10}$ to the bank. (In the original user-anonymous protocol, the user includes so-called blinding factors in the α_i's to hide the M_i's and m_i's from the bank (see also [3]) and uses a cut-and-choose protocol to convince the bank that he formed the α_i's as described in the protocol). The bank withdraws \$1023 from the user's account and sends back the signature $D \equiv \prod_{i=0}^{9} \alpha_i^{1/(17\cdot 3^{10-i})}$. Then the user can compute (note that $6947 \cdot 17 - 2 \cdot 3^{10} = 1$)

$$C \equiv D^{-2\cdot 3^{10}} \cdot \left(\prod_{i=0}^{9} M_i^{3^i}\right)^{6947} \cdot \left(\prod_{i=0}^{9} m_i^{3^i}\right)^{2} \equiv \prod_{i=0}^{9} M_i^{3^i/17},$$

$$C' \equiv D^{6947\cdot 3^{10}} \cdot \left(\prod_{i=0}^{9} M_i^{3^i}\right)^{-6947} \cdot \left(\prod_{i=0}^{9} m_i^{3^i}\right)^{-2} \equiv \prod_{i=0}^{9} m_i^{3^{i-10}}.$$

Assume that the user wants to pay \$$a$ = \$($\sum_{i=0}^{9} a_i 2^{10-i}$) at the shop with this check, where $a_i \in \{0, 1\}$ for $i = 0, \ldots, 9$. To this end, the user gives the number C to the shop, as well as the numbers $u_i := g(b_i, r_i)$ for those i with $a_i = 0$, and b_i, r_i for those

i with $a_i = 1$. The shop checks that C is correct, i.e., $C \equiv \prod_{a_i=0} u_i^{3^i/17} \prod_{a_i=1} g(b_i, r_i)^{3^i/17}$. After some time, the shop sends C and the revealed u_i, b_i, r_i to the bank (since the system is off-line, the shop does not send the numbers he received from a user at once, but first collects the numbers from several users). The bank checks that C is correct and that he did not receive the b_i's before, and stores the b_i's.

The user gets back from the bank the amount from this check which he did not spend (i.e., $\$(\sum_{i=0}^{9}(1 - a_i)2^{10-i})$) as follows: he gives C' to the bank together with the numbers $g(b_i, r_i)$ for those i with $a_i = 1$, and b_i, r_i for those i with $a_i = 0$. The bank checks that C' is correct, checks that he did not receive the revealed b_i's before and if so, refunds the user the money.

Assume that the user tries to cheat, i.e., tries to spend $\$a$ at the shop and to get back from the bank $\$b$, where $a + b > 1023$. Note that the user cannot show the same b_i to both the shop (when buying) and the bank (when asking for refund), otherwise he is caught cheating. The user could try to make a new signature $C_\sigma \equiv \prod_{i=0}^{9} M_i^{3^{\sigma(i)}/17}$ or $C'_\sigma \equiv \prod_{i=0}^{9} m_i^{17\cdot 3^{\sigma(i)-10}}$, where σ is a nonidentical permutation of $(0, \dots, 9)$. Then the user could cheat as follows: he looks for i_1 such that $\sigma(i_1) = i_2 > i_1$; he spends $\$(\sum_{i \neq i_1} 2^i)$ at the shop by using C (as explained above); and when asking for refund, he shows C'_σ and b_{i_1}, r_{i_1} to the bank, in order to get $\$(2^{i_2})$ as refund in stead of $\$(2^{i_1})$. The user could cheat with signature C_σ as follows: he looks for j_1 such that $\sigma(j_1) = j_2 < j_1$; he shows C_σ to the shop and spends $\$(\sum_{i \neq j_2} 2^i)$ using only $\$(\sum_{i \neq j_1} 2^i)$ from his check; and gets back from the bank $\$(2^{j_1})$ by showing it C'.

Note that in order to compute C_σ or C'_σ, the user may use $M_0, \dots, M_9$ $m_0, \dots, m_9, D$. Put $\mathbf{e}_1 = (1, 0, \dots, 0), \dots, \mathbf{e}_{20} = (0, \dots, 0, 1)$, $\mathbf{c}_\sigma = \sum_{i=1}^{10} (3^{\sigma(i-1)}/17)\mathbf{e}_i$, $\mathbf{c}'_\sigma = \sum_{i=1}^{10} 3^{\sigma(i)-10}\mathbf{e}_{i+10}$, $\mathbf{d} = \sum_{i=1}^{10} (17\cdot 3^{11-i})^{-1}(3^{10}\mathbf{e}_i + 17\mathbf{e}_{i+10})$, $\mathbf{m} = (M_0, \dots, M_9, m_0, \dots, m_9)$. Thus, $M_i = \mathbf{m}^{\mathbf{e}_{i+1}}$ $(i = 0, \dots, 9)$, $m_i = \mathbf{m}^{\mathbf{e}_{i+11}}$ $(i = 0, \dots, 9)$, $D = \mathbf{m}^{\mathbf{d}}$, $C_\sigma = \mathbf{m}^{\mathbf{c}_\sigma}$, $C'_\sigma = \mathbf{m}^{\mathbf{c}'_\sigma}$. Assume that the user cannot compute RSA-roots (mod N) of random numbers, and that f, g are good pseudo-random functions, so that we may consider $\mathbf{m}$ as a randomly chosen vector. Then the corollary implies that the user can compute C_σ or C'_σ from $M_0, \dots, M_9, m_0, \dots, m_9, D$ if and only if $\mathbf{c}_\sigma$ or $\mathbf{c}'_\sigma$ belongs to $A = \mathbb{Z}\{\mathbf{e}_1, \dots, \mathbf{e}_{20}, \mathbf{d}\}$. But it is easy to show, using that 3 is a primitive root (mod 17), that none of the vectors $\mathbf{c}_\sigma$, $\mathbf{c}'_\sigma$ belongs to A unless σ is the identity. Hence the kind of attack described above fails at this check-system.

Example 2. Consider the user-anonymous off-line coin system of [8]. In this system, the bank uses a signature scheme which we do not specify here. The user makes RSA-signatures using his own modulus N whose factorization he keeps secret; so here the user plays the role of a signature authority. Let L be a fixed integer, and define $I \equiv (\mathrm{ID}_{\mathrm{user}} \| R)^L$ mod N, where R is a number chosen at random by the user. In Fig. 2 the basic idea of the withdrawal (in which the user is able to blind and the bank to sign messages, see [8]) and spending protocol of a coin is given. Later on, the shop sends the numbers that it received to the bank and the bank verifies that these numbers have not been used before.

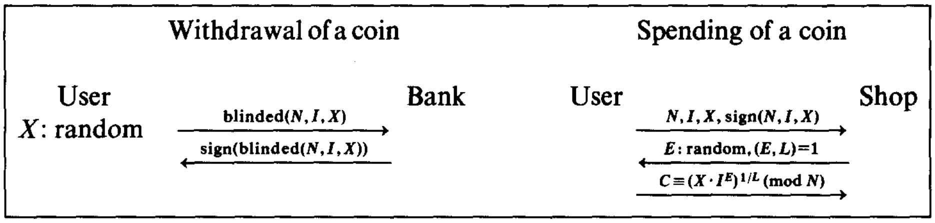

Fig. 2. The (simplified) off-line coin system of [8].

From the corollary it follows that it is not feasible for the shop/bank to compute the identity of the user (i.e., $I^{1/L} \bmod N$) from N, I, X, E, L and $C = X^{1/L} \cdot I^{E/L}$. But if the user spends the same coin at two shops, then the bank receives N, I, X, L, sign(N, I, X), two integers E_1, E_2 that are coprime with L, and the signatures $(X \cdot I^{E_1})^{1/L} \pmod N$ and $(X \cdot I^{E_2})^{1/L} \pmod N$. From the corollary it follows that the bank can compute $I^{1/L} \bmod N$ from this (and hence the user's identity) if and only if $\gcd(E_1 - E_2, L) = 1$. Hence the probability that a double spender is caught by the bank is approximately $\varphi(L)/L$ (because E_1, E_2 are randomly chosen). This probability is close to 1 if L is a large prime, and close to 0 if L is the product of many small primes. Therefore, it is not wise to let the user choose L freely himself (which was the original suggestion), but better to fix L as a large prime.

Acknowledgment

We would like to thank the referees for their valuable suggestions for improving the paper and shortening the proofs.

References

[1] S. Akl and P. Taylor, Cryptographic solution to a problem of access control in a hierarchy, *ACM Trans. Comput. Systems* **1** (1983), pp. 239–248.

[2] D. Chaum, H. den Boer, E. van Heyst, S. Mjølsnes, and A. Steenbeek, Efficient offline electronic checks, *Advances in Cryptology—Eurocrypt '89* (J.-J. Quisquater and J. Vandewalle, eds.), Lecture Notes in Computer Science, vol. 434, Springer-Verlag, Berlin, pp. 294–301.

[3] D. Chaum and J.-H. Evertse, A secure and privacy-protecting protocol for transmitting personal information between organizations, *Advances in Cryptology—Crypto '86* (A. M. Odlyzko, ed.), Lecture Notes in Computer Science, vol. 263, Springer-Verlag, Berlin, pp. 118–167.

[4] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977), pp. 675–695.

[5] J. Hastad, On using RSA with low exponent in a public key network, *Advances in Cryptology—Crypto '85* (H. C. Williams, ed.), Lecture Notes in Computer Science, vol. 218, Springer-Verlag, Berlin, pp. 403–408.

[6] R. Kannan and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8** (1979), pp. 499–507.

[7] J. C. P. Miller, On factorization, with a suggested new approach, *Math. Comp.* **29** (1975), pp. 155–172.

[8] T. Okamoto and K. Ohta, Disposable zero-knowledge authentications and their applications to untraceable electronic cash, *Advances in Cryptology—Crypto '89* (G. Brassard, ed.), Lecture Notes in Computer Science, vol. 435, Springer-Verlag, Berlin, pp. 481–497.

[9] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, *Comm. ACM* **21** (1978), pp. 120–126.

[10] J. Rosser and L. Schoenfeld, Approximate formulas for some functions of prime numbers, *Illinois J. Math.* **6** (1962), pp. 64–94.

[11] A. Shamir, On the generation of cryptographically strong pseudorandom sequences, *ACM Trans. Computer Systems* **1** (1983), pp. 38–44.