On the Security of RDSA

Pierre-Alain Fouque and Guillaume Poupard

DCSSI Crypto Lab
51 Boulevard de La Tour-Maubourg
75700 Paris 07 SP, France
Pierre-Alain.Fouque@ens.fr
Guillaume.Poupard@m4x.org

Abstract. A variant of Schnorr's signature scheme called RDSA has been proposed by I. Biehl, J. Buchmann, S. Hamdy and A. Meyer in order to be used in finite abelian groups of unknown order such as the class group of imaginary quadratic orders. We describe in this paper a total break of RDSA under a plain known-message attack for the parameters that were originally proposed. It recovers the secret signature key from the knowledge of less than 10 signatures of known messages, with a very low computational complexity.
We also compare a repaired version of RDSA with GPS scheme, another Schnorr variant with similar properties and we show that GPS should be preferred for most of the applications.
Keywords. Signature scheme, cryptanalysis, DSA variant, known-message attack, lattice reduction, GPS.

1 Introduction

In 1989, C. Schnorr proposed a proof of knowledge of a discrete logarithm in groups of known prime order [13]. Such a zero-knowledge proof can be used as an interactive identification scheme and also be converted into a signature scheme using the Fiat-Shamir paradigm [3]. This scheme has motivated the design of many signature schemes, including the standard DSA [9].

Those variants are intended to achieve additional properties. Firstly, we can use a composite modulus instead of a prime modulus and keep its factorization secret. We can also use various groups with interesting cryptographic properties [2]. As a consequence, the order of the group in which the computations are performed may remain secret. Furthermore, the order of the publicly known bases used in those schemes can also be public or private. In the Schnorr scheme and DSA, both the order of the group and the order of the base are known. Other schemes achieve different combinations.

Two variants allow to use groups of unknown order and bases whose order is also unknown. The first one, GPS, was proposed by Girault [4] and further analyzed by Poupard and Stern [12]. The second one, called RDSA, has been proposed by I. Biehl, J. Buchmann, S. Hamdy and A. Meyer [1] in order to be implemented in finite abelian groups of unknown order such as the class group of imaginary quadratic orders.

Our results

In this paper, we describe a total break of RDSA under a plain known-message attack. It requires the knowledge of a few valid signatures for messages that do not have to be chosen by the attacker. Furthermore, the computational complexity of the attack is very low. As an example, an attacker that observes 10 signatures of known messages can recover the secret signature key using a single computer in about five minutes.

The RDSA scheme is presented in section 2. Then, the attack is described in section 3; some useful tools such as lattice reduction techniques are reminded and a full algorithm is detailed and analyzed. Finally, a comparison between GPS and a repaired version of RDSA shows that GPS should be preferred for most of the applications.

2 The RDSA signature scheme

The RDSA signature scheme is fully described in [1] and [2]. It performs computations in a finite abelian group G, written multiplicatively. The basic idea of RDSA is to transform the Schnorr [14] and the DSA [9] schemes in order to use groups of unknown order.

We remind the RDSA scheme, using the original notations:

1. **Key generation**
   - randomly select an element $\gamma \in G$,
   - randomly select a prime q,
   - let $h(.)$ be a cryptographic hash function with outputs in the range $[0, q-1]$,
   - randomly select an integer $a \in [2, q-1]$,
   - compute $\alpha = \gamma^a$.

   The public key is (G, γ, α, q) and the private key is a.
2. **Signature of a message** M
   - randomly select an integer $k \in [0, q-1]$,
   - compute $\varrho = \gamma^k$, $e = h(M||\varrho)$ and $x = k - a \times e$,
   - compute integers s and ℓ such that $x = \ell \times q + s$ and $0 \leq s < q$,
   - compute $\lambda = \gamma^\ell$.

   The signature of the message M is (s, ϱ, λ).
3. **Verification**
   A triplet $(s, \varrho, \lambda) \in \mathbb{Z} \times G \times G$ is a valid signature of the message M if and only if $0 \leq s < q$ and $\gamma^s \alpha^{h(M||\varrho)} \lambda^q = \varrho$.

Parameters: G a group of unknown order
$\gamma \in G$ and a prime q
Secret key: $a \in [2, q-1]$
Public key: $\alpha = \gamma^a$

Prover		Verifier
choose k in $[0, q-1]$		
$\varrho = \gamma^k \bmod n$	$\xrightarrow{\varrho}$	
check $e \in [0, q-1]$	$\xleftarrow{e}$	choose e in $[0, q-1]$
$x = k - a \times e$, $s = x \bmod q$		
$\ell = (x-s)/q$, $\lambda = \gamma^\ell$	$\xrightarrow{s, \lambda}$	check $\gamma^s \alpha^e \lambda^q \stackrel{?}{=} \varrho$ and $s \in [0, q-1]$

Fig. 1. RDSA identification scheme

This signature scheme follows the Fiat-Shamir paradigm [3] that transforms any 3-move zero-knowledge interactive identification scheme in a signature scheme. This heuristic has been widely used to design signature schemes, even if the resulting security is only guaranteed in the random oracle model (see [10] for general results). Figure 1 describes what we call RDSA identification scheme.
Note: This scheme can be modified in order to make the signatures shorter, without reducing the security nor increasing the computation time. The idea is to output (e, s, λ) instead of (s, ϱ, λ) as the signature and to use the following verification procedure:

3bis. **Verification**
A triplet $(e, s, \lambda) \in \mathbb{Z} \times \mathbb{Z} \times G$ is a valid signature of the message M if and only if $0 \le e < q$, $0 \le s < q$ and $h(M || \gamma^s \alpha^e \lambda^q) = e$.

2.1 Security results on RDSA

The aim of the security analysis or RDSA is to prove that if an attacker has a non negligible probability of success to forge a valid signature, it can be transformed into an efficient algorithm that solves a problem which intractability is widely assumed.

We now define the number theoretical problems used in the analysis of RDSA. We note $< \alpha >$ the subgroup of G generated by the element $\alpha \in G$:

Discrete logarithm problem: given elements $\alpha \in G$ and $\beta \in < \alpha >$, find an integer k with $\alpha^k = \beta$.

Order problem: given an element $\alpha \in G$, find a non zero multiple of the order of α.

Root problem: given a prime number q that does not divide the order of G and an element $\alpha \in G$, find an element $\beta \in G$ with $\beta^q = \alpha$.

Small discrete log existence problem: given an integer q and elements $\alpha \in G$ and $\beta \in <\alpha>$, decide if there exists $k \in [0, q-1]$ such that $\alpha^k = \beta$.

The security analysis of RDSA is based on the forking lemma of Pointcheval and Stern [10]. It is claimed in [1] that RDSA is secure against no-message attack in the random oracle model. Then, using a simulation argument, the following theorem is announced:

Theorem 1 (from [1]). *In the random oracle model, if an existential forgery of a RDSA signature using an adaptively chosen message attack has a non-negligible probability of success, then the root problem or the small discrete log existence problem can be solved in probabilistic polynomial time. Probabilities are taken over random tapes, random oracles and public keys.*

The authors of RDSA also propose a modification of the signature scheme to avoid the assumption of intractability of the small discrete log existence problem. Let L be an approximation of the (unknown) order of the group G. Instead of choosing the random k in $[0, q-1]$, it is chosen in $[0, q \times L^2[$. We will further refer to this variant as RDSA2. The drawback, in comparison with RDSA, is an important degradation of performances.

3 A total break of RDSA under a known-message attack

3.1 Preliminary remarks on RDSA identification scheme

Before describing the attack against RDSA signature scheme, let us first have a look on the related interactive identification scheme described in figure 1. The main observation is that the random number $k \in [0, q-1]$ is used to mask the secret key a in the equation $x = k - a \times e$. However, since there is no modular reduction like in the Schnorr scheme, this random number is too small to hide all the secret. More precisely, after euclidian division of x by q,

$$x = \ell \times q + s \quad \text{with} \quad 0 \leq s < q$$

the quotient ℓ is a very good approximation of $(-a \times e)/q$

$$\ell = \left\lfloor \frac{x}{q} \right\rfloor = \left\lfloor \frac{k - a \times e}{q} \right\rfloor = \left\lfloor \frac{-a \times e}{q} \right\rfloor + \varepsilon \;\text{ with }\; \varepsilon \in \{0, 1\}$$

This first remark can be combined with a second very simple one; the value of ℓ is not disclosed to the verifier but only $\lambda = \gamma^\ell$ is transmitted. We could assume that if the discrete logarithm problem is intractable, the value of ℓ is not disclosed. However, since we have seen that ℓ is approximately equal to $(-a \times e)/q$ and $a \in [0, q-1]$, the size of ℓ is the same as the size of e. Consequently, if e is small enough, the verifier can use an algorithm such as the Pollard rho method [11], or even an exhaustive search, to compute ℓ from λ.

In conclusion, if the verifier sends a small enough challenge e, he can compute ℓ and consequently learn a good approximation of $(a \times e)/q$. Then it is easy to deduce the most significant bits of the secret key a. Indeed, assume e is δ bits long; as we have $s = k - a \times e - \ell \times q$ and $0 \leq s < q$, we obtain $|a \times e + \ell \times q| < q + |k| < 2q$. Therefore $\left|a - \frac{-\ell q}{e}\right| < \frac{2q}{e}$ and the $(\delta - 1)$ most significant bits of $-\ell q/e$ and a are the same.

Furthermore, this attack can be iterated with challenges of increasing size; the most significant bits of a allow to upper and lower bound the discrete logarithm ℓ of λ and a variant of the Pollard rho method enables to compute such discrete logs. Finally, the whole key a is recovered.

The flaw in the RDSA identification scheme is that the zero-knowledge property is not satisfied; a malicious verifier can learn information during authentication. Of course, such an adversary does not really follow the protocol since it does not randomly choose the challenges. We could hope for the protocol to be at least "honest verifier zero-knowledge". This condition would be sufficient to apply the forking lemma. However, we will see in the sequel that even if verifiers are honest, the protocol is not zero-knowledge. This means that it is not secure, even against passive adversaries that just eavesdrop communications. A consequence is that the forking lemma cannot be used and that the proof of theorem 1 collapses.

Let us now turn to the analysis of RDSA signature scheme. The main difference with the identification scheme is that the challenges are no longer chosen by the verifier but computed as the hash value of the commitment $\varrho = \gamma^k$ and of the message M to be signed: $e = h(M||\gamma^k)$. The consequence is that it is no longer possible for an attacker to control the size of e. At first sight, this seems to defeat the attack against the identification scheme we have just described. However, we explain in the following how to combine known valid signatures in order to obtain some "pseudo-signatures" with small e.

3.2 Generic algorithms for computing discrete logarithms

In any group G, the computation of discrete logarithms in base $\gamma \in G$ of order n can be performed in time $O(\sqrt{n})$ group multiplications with the baby-step giant-step algorithm. This algorithm is deterministic but requires storage for $O(\sqrt{n})$ elements so it is usually advised to use Pollard's rho algorithm [11] which has, heuristically, a similar running time but requires only a negligible amount of memory. Shoup proved in [15] that those algorithms are optimal for computing discrete logarithms in any group, i.e. without trying to take advantage of any additional algebraic structure.

Furthermore, if it is known that the discrete logarithm lies within a restricted interval of width w, another algorithm of Pollard [11, 16] called the lambda method (or the *method for catching kangaroos*) finds the discrete log in time $O(\sqrt{w})$ and space for $O(\log w)$ group elements.

3.3 The LLL toolbox

The lattice reduction algorithm of Lenstra, Lenstra and Lovász [7] has been widely used in cryptanalysis to break many kinds of cryptosystems. Details on lattice reduction techniques are out of the scope of this paper so we refer to [6] for details and extensive bibliography.

It should be noted that the LLL algorithm has already been used by Howgrave-Graham and Smart [5] and then by Nguyen and Shparlinski [8] to attack DSA if ephemeral keys, the equivalent of the k parameter in RDSA, are partially known.

We use LLL to solve the following problem: given $(e_1, ...e_n) \in [0, q-1]^n$, find integer coefficients $(c_1, ...c_n)$ as small as possible such that the linear combination $\sum_{j=1}^{n} c_j \times e_j$ has a prescribed bit size.

Consider the matrix M where the n rows are seen as $(n+1)$-dimensional vectors that define a lattice in $\mathbb{Z}^{n+1}$:

$$M = \begin{pmatrix} e_1 & \Lambda & 0 & \cdots & 0 \\ e_2 & 0 & \Lambda & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ e_n & 0 & \cdots & 0 & \Lambda \end{pmatrix}$$

The aim of a reduction algorithm, such as LLL, is to compute a "reduced" basis, i.e. a basis of the same lattice with short vectors that are "nearly" orthogonal. Let us consider a vector V of such a basis; it is a linear combination of the rows of M so all its coordinates, except the first one, are multiples of Λ:

$$V = \sum_{j=1}^{n} c_j \times (e_j, \overbrace{0, ..., 0}^{j-1}, \Lambda, \overbrace{0, ..., 0}^{n-j})$$

$$= \left(\left(\sum_{j=1}^{n} c_j \times e_j \right), \Lambda \times c_1, ..., \Lambda \times c_n \right)$$

In order to estimate the size of the coordinates of V, let us compute the determinant Δ and then the volume of the lattice:

$$\Delta = \det(M \times {}^tM) = \begin{vmatrix} e_1^2 + \Lambda^2 & e_1 \times e_2 & e_1 \times e_3 & \cdots & e_1 \times e_n \\ e_2 \times e_1 & e_2^2 + \Lambda^2 & e_2 \times e_3 & \cdots & e_2 \times e_n \\ e_3 \times e_1 & e_3 \times e_2 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & e_{n-1} \times e_n \\ e_n \times e_1 & \cdots & \cdots & e_n \times e_{n-1} & e_n^2 + \Lambda^2 \end{vmatrix}$$

If we assume that E is an order of magnitude of e_i, we can compute an approximation of Δ:

$$\Delta \approx \begin{vmatrix} E^2+\Lambda^2 & E^2 & \cdots & E^2 \\ E^2 & E^2+\Lambda^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & E^2 \\ E^2 & \cdots & E^2 & E^2+\Lambda^2 \end{vmatrix} = \Lambda^{2(n-1)}\left(\Lambda^2+nE^2\right)$$

The LLL lattice reduction algorithm is expected to output a short vector V of the lattice:

$$V = (\widetilde{e}, \Lambda \times c_1, ...\Lambda \times c_n) \quad \text{with} \quad \widetilde{e} = \sum_{j=1}^{n} c_j \times e_j$$

Even if we cannot prove it, we assume that the LLL algorithm outputs a reduced basis with vectors of about the same norm. Of course, this is just heuristic but it is a well known fact that the results of LLL are much better than what can be proved. Furthermore, since we use LLL for cryptanalysis, only the result matters and we will see in section 3.6 that this assumption is validated by the success of the attack in practice.

If the vectors of the reduced basis have similar norms and if the basis is "nearly" orthogonal, we obtain, using the volume $\sqrt{\Delta}$ of the lattice M, that

$$\sqrt{\widetilde{e}^2 + \Lambda^2 \times \sum_{j=1}^{n} c_j^2} \approx \left(\sqrt{\Delta}\right)^{\frac{1}{n}}$$

Furthermore, since V is a short vector of the lattice, we assume that all its coordinates have the same order of magnitude ($\forall j \;\; \widetilde{e} \approx \Lambda \times c_j$). We obtain that

$$\sqrt{n+1} \times \widetilde{e} \approx \left(\sqrt{\Delta}\right)^{\frac{1}{n}}$$

so

$$\widetilde{e} \approx \frac{1}{\sqrt{n+1}} \times \Lambda^{\frac{n-1}{n}} \left(\sqrt{\Lambda^2+nE^2}\right)^{\frac{1}{n}} \approx \Lambda \times \left(\frac{E}{\Lambda}\right)^{\frac{1}{n}} \times \frac{n^{\frac{1}{2n}}}{\sqrt{n+1}}$$

and

$$\sum_{j=1}^{n} |c_j| \approx \frac{n\widetilde{e}}{\Lambda} \approx \left(\frac{E}{\Lambda}\right)^{\frac{1}{n}} \times \frac{n^{1+\frac{1}{2n}}}{\sqrt{n+1}}$$

With $E = q$ and assuming $n \ll \Lambda$ and $n \ll q$, we finally obtain first order approximations on the bit size of the integer $\widetilde{e}$ and the sum of the coefficients c_j:

$$\log \widetilde{e} \approx \log \Lambda + \frac{1}{n} \log q - \frac{1}{n} \log \Lambda + \frac{1}{2n} \log n - \frac{1}{2} \log(n+1)$$
$$\approx \frac{\log q}{n} + \left(1 - \frac{1}{n}\right) \times \log \Lambda$$
$$\log\left(\sum_{j=1}^{n} |c_j|\right) \approx \log n + \log \widetilde{e} - \log \Lambda \approx \frac{\log q}{n} - \frac{\log \Lambda}{n}$$

3.4 Definition of "pseudo-signatures"

Let us assume that we know valid signatures $(s_j, \varrho_j, \lambda_j)$ of n messages M_j. This means that, for all $j \in [1, n]$, $0 \leq s_j < q$ and $\gamma^{s_j} \alpha^{h(M_j||\varrho_j)} \lambda_j^q = \varrho_j$. We note $e_j = h(M_j||\varrho_j)$.

We first notice that signatures can be combined in order to obtain what we call "pseudo-signatures", i.e. quadruples $(\widetilde{e}, \widetilde{s}, \widetilde{\varrho}, \widetilde{\lambda})$ that fit the verification equations $0 \leq \widetilde{s} < q$ and $\gamma^{\widetilde{s}} \alpha^{\widetilde{e}} \left(\widetilde{\lambda}\right)^q = \widetilde{\varrho}$ but that are not necessarily associated to any known message. Let $c_1, ...c_n$ be n integers. The following formulas allow to combine n signatures into a pseudo-signature $(\widetilde{e}, \widetilde{s}, \widetilde{\varrho}, \widetilde{\lambda})$:

$$\widetilde{e} = \sum_{j=1}^{n} c_j \times e_j \qquad \widetilde{\lambda} = \left(\prod_{j=1}^{n} \lambda_j^{c_j}\right) \times \gamma^{(\widetilde{s}_0 - \widetilde{s})/q}$$
$$\widetilde{s}_0 = \sum_{j=1}^{n} c_j \times s_j \qquad \widetilde{\varrho} = \prod_{j=1}^{n} \varrho_j^{c_j}$$
$$\widetilde{s} = \widetilde{s}_0 \bmod q$$

We verify that $(\widetilde{e}, \widetilde{s}, \widetilde{\varrho}, \widetilde{\lambda})$ is a pseudo-signature because $0 \leq \widetilde{s} < q$ and

$$\gamma^{\widetilde{s}} \alpha^{\widetilde{e}} \left(\widetilde{\lambda}\right)^q = \gamma^{\widetilde{s}} \times \alpha^{\sum_{j=1}^{n} c_j \times e_j} \times \left(\prod_{j=1}^{n} \lambda_j^{c_j}\right)^q \times \gamma^{q \times \frac{\widetilde{s}_0 - \widetilde{s}}{q}}$$
$$= \gamma^{\widetilde{s}_0} \times \prod_{j=1}^{n} \left(\alpha^{e_j}\right)^{c_j} \times \prod_{j=1}^{n} \left(\lambda_j^q\right)^{c_j} = \prod_{j=1}^{n} \left(\gamma^{s_j} \alpha^{e_j} \lambda_j^q\right)^{c_j} = \prod_{j=1}^{n} \varrho_j^{c_j} = \widetilde{\varrho}$$

If we further note $\widetilde{k}$ the discrete log of $\widetilde{\varrho}$ in base γ and $\widetilde{\ell}$ the discrete log of $\widetilde{\lambda}$, we obtain the equation

$$\log_\gamma \left(\gamma^{\widetilde{s}} \alpha^{\widetilde{e}} \left(\widetilde{\lambda}\right)^q\right) = \widetilde{s} + \widetilde{e} \times a + q \times \widetilde{\ell} = \log_\gamma(\widetilde{\varrho}) = \widetilde{k}$$

so

$$q \times \widetilde{\ell} + \widetilde{s} = \widetilde{k} - a \times \widetilde{e}$$

where q, $\widetilde{s}$ and $\widetilde{e}$ are known, a and $\widetilde{k}$ are unknown and $\widetilde{\ell}$ can be computed if it lies in a small enough range to make the Pollard-lambda algorithm practical on $\widetilde{\lambda}$.

3.5 The attack

The attack is basically a loop where some bits of the secret key a are found at each round, from the most significant ones to the least significant. Let us assume that, after i rounds, the β_i most significant bits of the secret key a are known. We note $a = a_i + a'_i$, where a_i is known and a'_i is bounded by $\underline{A}_i \leq a'_i \leq \overline{A}_i$. Initially, $\underline{A}_0 = 2$, $\overline{A}_0 = q - 1$ and $a_0 = 0$.

Let us further compute a pseudo-signature $(\widetilde{e}, \widetilde{s}, \widetilde{\varrho}, \widetilde{\lambda})$ using the coefficients $c_1, ...c_n$ output by the LLL-based algorithm of section 3.3 with a parameter Λ_i that will be precised below.

Firstly, $\widetilde{k}$ is equal to the linear combination $\sum_{j=1}^{n} c_j \times k_j$ of the (unknown) integers k_j. Since $0 \leq k_j < q$, we have

$$\underline{K}_i \leq \widetilde{k} \leq \overline{K}_i \quad \text{with} \quad \underline{K}_i = (q-1) \sum_{c_j < 0} c_j \quad \text{and} \quad \overline{K}_i = (q-1) \sum_{c_j > 0} c_j$$

We immediately see that $\widetilde{k} - \widetilde{e} \times a$ is bounded by

$$\underline{K}_i - \widetilde{e} \times a_i - \widetilde{e} \times \overline{A}_i \leq \widetilde{k} - \widetilde{e} \times (a_i + a'_i) \leq \overline{K}_i - \widetilde{e} \times a_i - \widetilde{e} \times \underline{A}_i$$

and consequently that

$$\frac{\underline{K}_i - \widetilde{e} \times \overline{A}_i}{q} - \frac{\widetilde{e} \times a_i}{q} - 1 < \left\lfloor \frac{\widetilde{k} - \widetilde{e} \times (a_i + a'_i)}{q} \right\rfloor \leq \frac{\overline{K}_i - \widetilde{e} \times \underline{A}_i}{q} - \frac{\widetilde{e} \times a_i}{q}$$

We observe that $\widetilde{\ell} = \lfloor (\widetilde{k} - \widetilde{e} \times a)/q \rfloor$ so we obtain bounds for $\widetilde{\ell}$:

$$\widetilde{\ell} = \left\lfloor -\frac{\widetilde{e} \times a_i}{q} \right\rfloor + \delta \;\; \text{with} \;\; \delta \in \left\{ \left\lfloor \frac{\underline{K}_i - \widetilde{e} \times \overline{A}_i}{q} \right\rfloor, ..., \left\lfloor \frac{\overline{K}_i - \widetilde{e} \times \underline{A}_i}{q} \right\rfloor + 1 \right\}$$

If the range of δ is not too large, $\widetilde{\ell}$ can be computed from $\widetilde{\lambda}$ using an algorithm described in section 3.2. Since $\widetilde{k} - \widetilde{e} \times a = q \times \widetilde{\ell} + \widetilde{s}$, we obtain

$$a = \frac{\widetilde{k} - (q \times \widetilde{\ell} + \widetilde{s})}{\widetilde{e}}$$

and consequently, using the bounds on $\widetilde{k}$,

$$\frac{\underline{K}_i}{\widetilde{e}} - \frac{q \times \widetilde{\ell} + \widetilde{s}}{\widetilde{e}} \leq a \leq \frac{\overline{K}_i}{\widetilde{e}} - \frac{q \times \widetilde{\ell} + \widetilde{s}}{\widetilde{e}}$$

We have obtained a new approximation a_{i+1} of the secret key a:

$$a_{i+1} = \left\lfloor -\frac{q \times \widetilde{\ell} + \widetilde{s}}{\widetilde{e}} \right\rfloor \quad \text{with} \quad \left\lfloor \frac{\underline{K}_i}{\widetilde{e}} \right\rfloor \leq a - a_{i+1} \leq \left\lfloor \frac{\overline{K}_i}{\widetilde{e}} \right\rfloor + 1$$

Consequently, we define $\underline{A}_{i+1} = \left\lfloor \frac{\underline{K}_i}{\widetilde{e}} \right\rfloor$ and $\overline{A}_{i+1} = \left\lfloor \frac{\overline{K}_i}{\widetilde{e}} \right\rfloor + 1$

We now consider $\beta_i = \log q - \log(\overline{A}_i - \underline{A}_i)$, i.e. β_i is an estimation of the number of bits of the secret key that have been learned after i rounds of attack. We estimate the value of β_{i+1}, i.e. the number of bits that have been learned after round $i+1$, using the results of section 3.3:

$$\begin{aligned}
\beta_{i+1} &= \log q - \log(\overline{A}_{i+1} - \underline{A}_{i+1}) \approx \log q - \log\left(\frac{\overline{K}_i - \underline{K}_i}{\widetilde{e}}\right) \\
&\approx \log q - \left(\log q + \log\left(\sum_{j=1}^{n} |c_j|\right) - \log \widetilde{e}\right) \\
&\approx \log q - \left(\left(1 + \frac{1}{n}\right)\log q - \frac{1}{n}\log \Lambda_i - \frac{1}{n} \times \log q - \left(1 - \frac{1}{n}\right)\log \Lambda_i\right) \\
&\approx \log \Lambda_i
\end{aligned}$$

Furthermore, the number of group operations needed to find $\widetilde{\ell}$ is about

$$\sqrt{\frac{\overline{K}_i - \underline{K}_i + \widetilde{e}(\overline{A}_i - \underline{A}_i)}{q}}$$

Always using the results of section 3.3, we can write

$$\log\left(\frac{\overline{K}_i - \underline{K}_i}{q}\right) \approx \left(\log \sum_{j=1}^{n} |c_j|\right) \approx \frac{1}{n}\log q - \frac{1}{n}\log \Lambda_i$$

$$\log\left(\frac{\widetilde{e}(\overline{A}_i - \underline{A}_i)}{q}\right) \approx \log(\widetilde{e}) + \log\left(\frac{\overline{A}_i - \underline{A}_i}{q}\right) \approx \frac{1}{n}\log q + \left(1 - \frac{1}{n}\right) \times \log \Lambda_i - \beta_i$$

If we assume that $\log \Lambda_i = \beta_{i+1}$ is greater than β_i, i.e. that the number of known bits of a increases, and if we note T the number of group operations that can be performed to compute $\widetilde{\ell}$, we obtain the estimation

$$\left(1 - \frac{1}{n}\right)(\log \Lambda_i - \log q) + \log q - \beta_i \approx \log(T^2)$$

and finally

$$\beta_{i+1} \approx \log \Lambda_i \approx \frac{2\log T + \beta_i - \frac{1}{n}\log q}{1 - \frac{1}{n}}$$

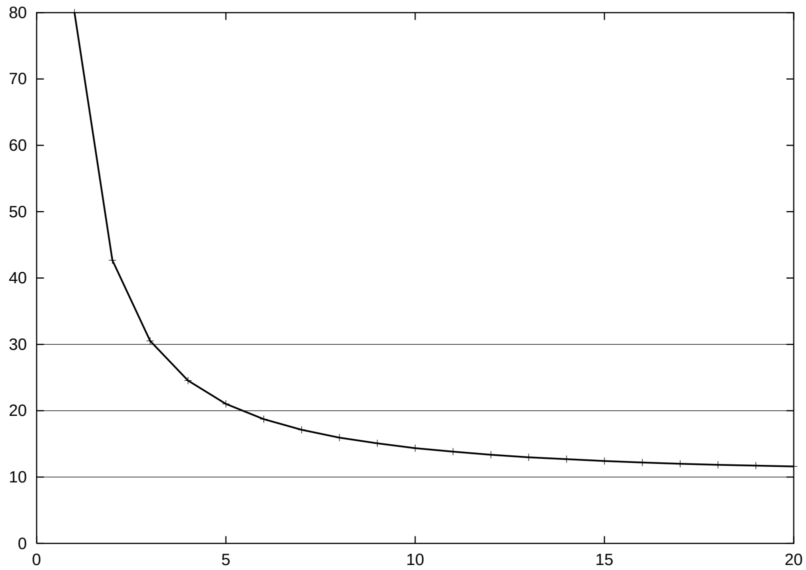

Fig. 2. Practical computation time of the attack ($\log_2$ of the number of group operation) according to the number n of available valid signatures ($|q| = 160$ bits).

This enables to calibrate the size of Λ_i used for lattice reduction in the algorithm of section 3.3.

We finally obtain the algorithm of appendix A where the inputs are the public key (G, γ, α, q), the number T of group operations that can be done in G for each computation of a discrete log in a small range, and n signatures $(s_j, \varrho_j, \lambda_j)$ of messages M_j. The output is the secret key a.

3.6 Practical efficiency of the attack

An actual implementation of the attack confirms the validity of the previous analysis. It shows that the LLL complexity is small and, consequently, that most of the computation time is used to compute discrete logarithms that lie in known ranges. Figure 2 shows the total number of group operation required to find the secret signature key according to the number of known signatures, for a 160-bit long q parameter.

If we assume that a group operation is performed in 10 milliseconds, the knowledge of 10 signatures of known messages allows to find the secret key in 4 minutes.

As an example, using the randomly chosen 160-bit secret key

$$a = 783747269568486438745024665497732424427783872122$$

and 10 signatures of known randomly chosen messages, the algorithm outputs the following list of approximations of a :

$$
\begin{aligned}
a_1 &= 699385134427555275284339311004987510346869139626 \\
a_2 &= \mathbf{78}2479287601512066256106326496551372637480929902 \\
a_3 &= \mathbf{783}821065851910137404362049317488060299468649498 \\
a_4 &= \mathbf{78374}5856938901542621342549531086555421197152107 \\
a_5 &= \mathbf{783747}195027730332973546090431994377868006041180 \\
a_6 &= \mathbf{783747269}878977094681369544174970381613011805054 \\
a_7 &= \mathbf{78374726956}7797050637945617507444265545395300956 \\
a_8 &= \mathbf{783747269568}502213048617608799840610517322561625 \\
a_9 &= \mathbf{7837472695684864}54929541646754729663729190704664 \\
a_{10} &= \mathbf{78374726956848643}9419860587327869299275855671372 \\
a_{11} &= \mathbf{783747269568486438745 0}31810665073621629868186026 \\
a_{12} &= \mathbf{7837472695684864387450 2}4667262630368177995127031 \\
a_{13} &= \mathbf{7837472695684864387450246654 97}085963603094329441 \\
a_{14} &= \mathbf{78374726956848643874502466549773 24}32534250967186 \\
a_{15} &= \mathbf{7837472695684864387450246654977324244277}91638474 \\
a_{16} &= \mathbf{78374726956848643874502466549773242442777838 7 21}34
\end{aligned}
$$

We observe that after 16 applications of the LLL algorithm, all the digits of the secret signature key are known, excepted the 2 least significant ones.

4 Comparison between RDSA2 and GPS

The attack we have just described shows that the original RDSA scheme cannot be used. An obvious reparation is to use random masking parameters k chosen in a larger range. The RDSA2 scheme, described in section 2.1, uses $k \in [0, qL^2[$ where L is an estimation of the size of group G. The consequence is an important increasing of the computational complexity for both signers and verifiers.

The GPS scheme was first described in [4]. It is reminded in figure 3, using the RDSA notations in order to highlight the similarities between the two signature schemes. The main difference is the treatment of $x = k - a \times h(M||\varrho)$. In GPS, the information x is a part of the signature while it enables to compute s and λ in RDSA2.

GPS is provably secure [12] if $k \in [0, A[$ where A is much larger than q^2. We can consider that $1/2^{80}$ is negligible and consequently we choose $A = q^2 \times 2^{80}$. A comparison between RDSA2 and GPS shows that:

	RDSA2	GPS
key generation	choose a group G with unknown order choose an element $\gamma \in G$ choose a 160-bit prime number q choose a secret key $a \in [2, q-1]$ compute the public key $\alpha = \gamma^a$	
signature	choose $k \in [2, qL^2[$ $\varrho = \gamma^k$ $x = k - a \times h(M\|\|\varrho)$ $s = x \bmod q$ $\lambda = \gamma^{\frac{x-s}{q}}$ signature (s, ϱ, λ)	choose $k \in [0, q^2 \times 2^{80}[$ $\varrho = \gamma^k$ $x = k + a \times h(M\|\|\varrho)$ signature (ϱ, x)
verification	$s \in [0, q-1]$ $\gamma^s \lambda^q \alpha^{h(M\|\|\varrho)} = \varrho$	$x \in [0, q^2 \times \left(2^{80}+1\right)[$ $\gamma^x \alpha^{h(M\|\|\varrho)} = \varrho$

Fig. 3. Compared description of RDSA2 and GPS

– the security of GPS is based on the intractability of the discrete log problem while the security of RDSA2 is based on a stronger assumption, the intractability of the root problem,
– GPS signatures are shorter than RDSA2 signatures if the group G has more than 2^{240} elements,
– the random k is larger for RDSA2 than for GPS so precomputation of ϱ takes more time and requires more random bits for RDSA2,
– on-line computation for the signer requires an exponentiation in RDSA2 to compute λ while on-line computation of x in GPS takes a negligible amount of time, even using low cost smart cards,
– verification of an RDSA2 signature requires the computation of exponentiations with shorter exponents so verification is 16% to 32% longer in GPS than in RDSA2.

In conclusion, RDSA is not secure and RDSA2 has no advantage on GPS, whatever the criterion may be, except for verification time.

References

1. I. Biehl, J. Buchmann, S. Hamdy, and A. Meyer. A Signature Scheme Based on the Intractability of Computing Roots. *Designs, Codes and Cryptography*, 25(3):223–236, March 2002.
2. J. Buchmann and S. Hamdy. A Survey on IQ Cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15. Walter de Gruyter, 2001.

3. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – proceedings of CRYPTO '86*, Lecture Notes in Computer Science volume 263, pages 186–194. Springer-Verlag, 1987.
4. M. Girault. Self-Certified Public Keys. In *Advances in Cryptology – proceedings of EUROCRYPT '91*, Lecture Notes in Computer Science volume 547, pages 490–497. Springer-Verlag, 1992.
5. N. Howgrave-Graham and N.P. Smart. Lattice attacks on digital signature schemes. *Design, Codes and Cryptography*, 23:283 – 290, 2001.
6. A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.
7. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261, 1982.
8. P.Q. Nguyen and I.E. Shparlinski. The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. *Journal of Cryptology*, 15(3):151 – 176, 2002.
9. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186–2, february 2000.
10. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
11. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
12. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Advances in Cryptology – proceedings of EUROCRYPT '98*, Lecture Notes in Computer Science volume 1403, pages 422–436. Springer-Verlag, 1998.
13. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology – proceedings of CRYPTO '89*, Lecture Notes in Computer Science volume 435, pages 235–251. Springer-Verlag, 1990.
14. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
15. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology – proceedings of EUROCRYPT '97*, Lecture Notes in Computer Science volume 1233, pages 256–266. Springer-Verlag, 1997.
16. P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Advances in Cryptology – proceedings of EUROCRYPT '96*, Lecture Notes in Computer Science volume 1070, pages 332–343. Springer-Verlag, 1996.

A Cryptanalysis of RDSA [1] : a detailed algorithm

Algorithm 1.1 BREAK-RDSA$(G,\gamma,\alpha,q,T,\{(M_i,(s_i,\varrho_i,\lambda_i)), i \in [1,n]\})$

$a_0 \leftarrow 0$, $\underline{A}_0 \leftarrow 2$, $\overline{A}_0 \leftarrow q-1$, $\beta_0 \leftarrow 0$, $i \leftarrow 0$

while $\overline{A}_i - \underline{A}_i > T^2$ **do**

$\quad \beta_{i+1} \leftarrow \frac{2\log T + \beta_i - \frac{\log q}{n}}{1-\frac{1}{n}}$

$\quad \Lambda_i \leftarrow 2^{\beta_{i+1}}$

$\quad M \leftarrow \begin{pmatrix} e_1 & \Lambda_i & 0 & \cdots & 0 \\ e_2 & 0 & \Lambda_i & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ e_n & 0 & \cdots & 0 & \Lambda_i \end{pmatrix}$ where the n rows are viewed as the vectors of a lattice

$\quad$ apply the LLL algorithm to M; let $V = (\widetilde{e}, \Lambda_i \times c_1, \ldots \Lambda_i \times c_n)$ be the shortest vector of the reduced basis

$\quad \widetilde{s}_0 \leftarrow \sum_{j=1}^{n} c_j \times s_j$

$\quad \widetilde{s} \leftarrow \widetilde{s}_0 \bmod q$

$\quad \widetilde{\varrho} \leftarrow \prod_{j=1}^{n} \varrho_j^{c_j}$

$\quad \widetilde{\lambda} \leftarrow \prod_{j=1}^{n} \lambda_j^{c_j} \times \gamma^{(\widetilde{s}_0 - \widetilde{s})/q}$

$\quad \underline{K}_i \leftarrow (q-1)\sum_{c_j<0} c_j$

$\quad \overline{K}_i \leftarrow (q-1)\sum_{c_j>0} c_j$

$\quad$ compute the discrete log $\widetilde{\ell}$ of $\widetilde{\lambda}$ in base γ; $\widetilde{\ell}$ is known to be in the range $\left[\left\lfloor \frac{\underline{K}_i - \widetilde{e} \times \overline{A}_i}{q} - \frac{\widetilde{e} \times a_i}{q} \right\rfloor, \left\lfloor \frac{\overline{K}_i - \widetilde{e} \times \underline{A}_i}{q} - \frac{\widetilde{e} \times a_i}{q} \right\rfloor\right]$

$\quad a_{i+1} \leftarrow \left\lfloor -\frac{q\widetilde{\ell} + \widetilde{s}}{\widetilde{e}} \right\rfloor$

$\quad \underline{A}_{i+1} \leftarrow \left\lfloor \frac{\underline{K}_i}{\widetilde{e}} \right\rfloor$, $\overline{A}_{i+1} \leftarrow \left\lfloor \frac{\overline{K}_i}{\widetilde{e}} \right\rfloor + 1$

$\quad i \leftarrow i+1$

compute the discrete log a of α in base γ; a is known to be in the range $[a_i + \underline{A}_i, a_i + \overline{A}_i]$

output the secret key a