On the Security of the Schnorr Scheme using Preprocessing

Peter de Rooij
PTT Research*

Abstract

In this paper, it is shown that the Schnorr scheme with preprocessing as proposed in [4] leaks too much information. An attack based on this information leakage is presented that retrieves the secret key. The complexity of this attack is upper bounded by $2k \cdot k^{3(d-2)}$ steps, and the expected required number of signatures is less than $2k \cdot (\frac{k}{2})^{d-2}$, where k is a security parameter. This complexity is significantly lower than the $k^{k(d-2)}$ steps, conjectured in [4]. For example, for the security parameters that are proposed in [4], the secret key can on average be found in $2^{37.5}$ steps, instead of in 2^{72} steps. This shows that it is inevitable to either modify the preprocessing algorithm, or choose the values of the security parameters larger than proposed in [4].

Finally, we briefly discuss the possibility of averting the proposed attack by modifying the preprocessing algorithm.

1 Introduction

The Schnorr scheme [4] comprises an interactive identification protocol based on the discrete logarithm problem and a related signature scheme. In addition, a preprocessing algorithm that substantially speeds up the calculations of the prover/signer in the scheme is proposed. This preprocessing algorithm can be useful in any protocol where modular exponentiations are used.

In this paper, an attack based on the use of this preprocessing algorithm is presented. This attack shows that the values of the security parameters should be chosen larger than proposed in [4], in order to reach the designated level of security. Alternatively, the preprocessing algorithm itself can be modified.

This paper is organized as follows. In Section 2, the Schnorr scheme and the preprocessing algorithm are briefly introduced. In Section 3, an 'information leak' in the preprocessing is pointed out and an attack based on this leak is presented in Section 4. In Section 5 the possibility of averting the attack by changing the preprocessing algorithm is briefly investigated. Section 6 gives the conclusions.

*P.O. Box 421, 2260 AK Leidschendam, The Netherlands; E-mail P_dRooij@pttrnl.nl

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 71-80, 1991.

2 The scheme and the preprocessing algorithm

The identification protocol in the Schnorr scheme [4] is based on the Chaum-Evertse-van de Graaf-protocol [1]. Essentially, it condenses this protocol into one single round. Furthermore, a signature scheme, based on the identification protocol, is given. We briefly describe both the identification protocol and the signature scheme as far as relevant to this paper. For details, see [4].

2.1 Preliminaries

The following parameters are chosen once and for all, and are known to all users: a large prime p, a prime q that divides $p-1$, a primitive qth root of unity $\alpha \in \mathbb{Z}_p$ and a security parameter t. In [4], it is proposed to take p and q in the order of 512 bits and 140 bits respectively, and $t = 72$.

Each user chooses a secret key $s \in \mathbb{Z}_q^*$. The corresponding public key is $v = \alpha^{-s} \bmod p$. A key authentication center signs, for each user, a string (I, v), consisting of the identity I and the public key v of the user. This signature is used to authenticate the public key, but since this is of no relevance to this paper, we will disregard this aspect from now on.

2.2 The Identification Protocol

Suppose *prover* $\mathcal{A}$ wants to prove his identity to *verifier* $\mathcal{B}$. First, $\mathcal{A}$ picks a random number $r \in \mathbb{Z}_q^*$ and sends the *initial commitment* $x = \alpha^r \bmod p$ to $\mathcal{B}$. Then $\mathcal{B}$ returns a random number $e \in \{0, \ldots, 2^t - 1\}$, called the *challenge*, to $\mathcal{A}$. Finally, $\mathcal{A}$ sends $y = r + se \bmod q$ to $\mathcal{B}$. $\mathcal{B}$ checks $\mathcal{A}$'s proof of identity by calculating $\overline{x} = \alpha^y v^e \bmod p$. $\mathcal{B}$ will accept the proof if and only if $\overline{x} = x$.

In the rest of this paper, all calculations will be modulo q, except where indicated otherwise.

2.3 The Signature Scheme

The signature scheme is an extension of the identification protocol analogous to the extensions of the Fiat-Shamir and Guillou-Quisquater ID protocols [2, 3]. That is, a t-bit hash value of the initial commitment x and the message m to be signed replaces the challenge. The signature consists of this hash value and of y as in the identification protocol.

Let h denote the hash function that is used to compute the hash value. A message m is signed by *signer* $\mathcal{A}$ as follows. First, $\mathcal{A}$ picks a random number $r \in \mathbb{Z}_q^*$ and calculates $x = \alpha^r \bmod p$; from this $e = h(x, m)$ and $y = r + se$ are computed. The signature consists of the pair (y, e).

The signature of $\mathcal{A}$ on m can be checked as follows. Compute $\overline{x} = \alpha^y v^e \bmod p$ and $\overline{e} = h(\overline{x}, m)$. The signature will be accepted if and only if $\overline{e} = e$.

In the sequel, a pair (y, e) is called a signature, even if it is made by the identification protocol. The attack that will be proposed later on, is based on the assumption that p, q, α, t, and a sufficient number of correct signatures are available. The verifier, for example, possesses this information.

2.4 The preprocessing

The aim of the preprocessing is to reduce the computational effort of the prover/signer. This effort is determined by the modular exponentiation $\alpha^r \bmod p$. The preprocessing essentially enables the performance of this exponentiation with the effort of a few multiplications only. This is achieved by taking, instead of a random r, a linear combination of several random numbers $r_i \in \mathbb{Z}_q^*$ for which $x_i = \alpha^{r_i} \bmod p$ are precomputed.

Therefore, each user initially stores a collection of k pairs (r_i, x_i), $0 \leq i < k$, such that $x_i = \alpha^{r_i} \bmod p$. Here k is a security parameter. Furthermore, the r_i's are independently and randomly chosen from $\mathbb{Z}_q^*$. Then, for each signature, the pair (r, x) is chosen as a combination of these pairs (r_i, x_i). Subsequently, the collection of pairs is 'rejuvenated' by replacing one of the pairs (r_i, x_i) by a similar combination. The first time, (r_0, x_0) is replaced by a combination of the pairs (r_i, x_i) for $0 \leq i < k$. This new pair is denoted (r_k, x_k). The next time, (r_{k+1}, x_{k+1}) replaces (r_1, x_1), and so on. A security parameter d determines the number of pairs used in these combinations. As both the new and the original pairs (r_i, x_i) will be used, we will, in contrast to [4], not reduce the indices modulo k. Clearly, this does not alter the preprocessing itself.

Denote the value of r used in the ith initial commitment by r_i^*, and the initial commitment itself by x_i^*. We start numbering from k, for then the index of the initial commitment is the same as the index of the corresponding new r_i. Then this index can be used as a sequential number of the signatures. Denote this sequential number by ν. We now give the preprocessing algorithm in detail.

The Preprocessing Algorithm

0. *Initialization.* Load k pairs (r_i, x_i) as above, $0 \leq i < k$;
   $\nu := k$;
1. Pick random numbers $a(0;\nu), \ldots, a(d-3;\nu) \in \{\nu - k, \ldots, \nu - 1\}$;
   $a(d-2;\nu) := \nu - 1$; $a(d-1;\nu) := \nu - k$; $a(d;\nu) := \nu - 1$;
2. $r_\nu := \sum_{i=0}^{d} 2^i r_{a(i;\nu)} \bmod q$; $\quad x_\nu := \prod_{i=0}^{d} \left(x_{a(i;\nu)}\right)^{2^i} \bmod p$;
   $r_\nu^* := r_{\nu-k} + 2r_{\nu-1} \bmod q$; $\quad x_\nu^* := x_{\nu-k} \cdot x_{\nu-1}^2 \bmod p$;
3. Keep the pair (r_ν^*, x_ν^*) ready for the next signature;
   replace the pair $(r_{\nu-k}, x_{\nu-k})$ by the new pair (r_ν, x_ν);
4. $\nu := \nu + 1$;
   goto 1 for the next signature.

In [4], an algorithm for the computations in step **2** is given that requires d multiplications and d squarings modulo p, d additions modulo q and d bitshifts on q-bit numbers.

In the sequel, it is assumed that the $a(j;\nu)$ in the preprocessing algorithm are chosen independently from a uniform distribution. It seems reasonable to assume this, as any dependence or non-uniformity can be exploited by an attack.

3 The preprocessing leaks information

In [4], a number of possible attacks on the preprocessing in the Schnorr scheme are considered. Subsequently, values for the security parameters k and d are suggested that make those attacks infeasible. In this section, we will look at the preprocessing from another point of view. Instead of considering all possibilities for the $a(i;\nu)$ or the most likely ones, as is done in [4], we consider a special case only. This special case has a relatively low probability of occurrence, but it provides a much higher amount of information about the secret key.

The special case we will consider is the event 'for all i, $0 \leq i \leq d$, $a(i;\nu)$ takes on one of the values $\nu - 1$ and $\nu - k$'. If this happens, r_ν is a linear combination of $r_{\nu-1}$ and $r_{\nu-k}$ only. Now, in some cases, only three occurrences of this special case are needed to find the secret key with high probability, as the following lemma shows.

Lemma 3.1 *Let $0 < a < b$. Suppose that $a(j;\nu) = \nu - 1$ or $a(j;\nu) = \nu - k$ for the three values $\nu = i$, $\nu = i + a(k-1)$ and $\nu = i + b(k-1)$, and for $0 \leq j \leq d$.*

If the signatures with indices i, $i+k-1$, ..., $i+(b-1)(k-1)$, and $i+k$, $i+2k-1$, ..., $i+(b-1)(k-1)+1$ are available, then the secret key s can be determined with high probability by solving a system of three equations in three unknowns.

Proof: Note that r_i, $r_{i+a\cdot(k-1)}$ and $r_{i+b\cdot(k-1)}$, can be written as

$$r_i = \lambda r_{i-1} + \mu r_{i-k}, \tag{1}$$

$$r_{i+a\cdot(k-1)} = \lambda' r_{i-1+a\cdot(k-1)} + \mu' r_{i-1+(a-1)\cdot(k-1)}, \tag{2}$$

$$r_{i+b\cdot(k-1)} = \lambda'' r_{i-1+b\cdot(k-1)} + \mu'' r_{i-1+(b-1)\cdot(k-1)}, \tag{3}$$

yielding three equations in nine unknowns (the r_j's). We will collect a number of additional equations that link those r_j's and s. The *linking equations* will be provided by the signatures as follows. Since, by definition, $r_j^* = r_{j-k} + 2r_{j-1}$ and $y_j = r_j^* + se_j$ for all j, it follows that

$$r_{j-1} = 2^{-1} \cdot (y_j - se_j - r_{j-k}). \tag{4}$$

Repeated use of these linking equations enables us to write an arbitrary $r_{j+c(k-1)}$, $c \in \mathbb{N}$, as a linear combination of y_j, se_j and r_j. More precisely,

$$\begin{aligned} r_{j+c(k-1)} &= 2^{-1}\left(y_{j+c(k-1)+1} - se_{j+c(k-1)+1} - r_{j+(c-1)(k-1)}\right) \\ &= 2^{-1}\Big(y_{j+c(k-1)+1} - se_{j+c(k-1)+1} + \\ &\qquad -2^{-1}\left(y_{j+(c-1)(k-1)+1} - se_{j+(c-1)(k-1)+1} - r_{j+(c-2)(k-1)}\right)\Big) \\ &\;\;\vdots \\ &= (-1)^c 2^{-c} r_j + \sum_{l=1}^{c} (-1)^{l-c} 2^{l-c-1} \left(y_{j+l(k-1)+1} - se_{j+l(k-1)+1}\right) \end{aligned} \tag{5}$$

for all positive integers c.

With this equality, we can rewrite the Equations (1)–(3). The right-hand sides can be rewritten to linear combinations of s, r_{i-k} and known constants, provided the signatures (y_j, e_j) are available for $j = i, i+k-1, \ldots, i+b(k-1)$. Furthermore, if the signatures (y_j, e_j) for $j = i+k, \ldots, i+b(k-1)+1$ are available, the lefthand sides can be rewritten to linear combinations of s, r_i and known constants. Then we find three equations in the three unknowns s, r_i and r_{i-k}. These equations are independent with high probability, see Appendix A. □

4 An attack

From the previous section, it can be concluded that a small number of signatures yield sufficient information to recover the secret key s, provided some of the corresponding r_i's satisfy certain conditions. These conditions are stated concisely by the following definition.

Definition 4.1 *A set $C = \{i, i+a(k-1), i+b(k-1)\}$ of indices will be called* a candidate. *A candidate C will be said to* fit *if for $\nu \in C$ we have that $a(j;\nu) \in \{\nu-1, \nu-k\}$ for all j, $0 \le j \le d$. Likewise, a signature with sequential number ν will be said to fit if $a(j;\nu) \in \{\nu-1, \nu-k\}$ for all j, $0 \le j \le d$.*

So, if a fitting candidate C and signatures as indicated in Lemma 3.1 are available, then the secret key s can be calculated with high probability. Note that there are several possible values of λ, λ', λ'', μ, μ' and μ'' in (1)–(3), corresponding to different choices $a(j;\nu) = \nu - 1$ or $\nu - k$ for all j and for $\nu \in C$. Obviously one must know or guess these values in order to perform the calculations.

This leads us to the following idea for an attack. Consider a (large) number of candidates. Then, for all possible ways in which they can fit, do the following. First, calculate the solution $\hat{s}$ of the corresponding set of equations (1)–(3) using the appropriate linking equations, and subsequently check this solution by calculating $\alpha^{-\hat{s}} \bmod p$. We have found the secret key if $v = \alpha^{-\hat{s}}$.

The proposed attack

```
for (all possible candidates) do {
   for (all possible fittings for this candidate) do {
      calculate the corresponding solution ŝ;
      v̂ = α^(-ŝ) mod p;
      if (v̂ = v)
         stop;     /* we have found the secret key now!  */
   }
}
```

Note that the order in which the candidates are tried does not matter, for each candidate has the same probability of fitting. In the proofs in the sequel it is assumed for simplicity that they are tried in order of their largest index.

Call the calculation of one estimate s plus the corresponding v a *try*. The calculations for one try comprise a few modular multiplications and additions modulo q for the calculation of s, the calculation of $\hat{v}$ requires a full exponentiation modulo p, albeit with a $\log q$-bit exponent only (which is in the order of $\log p / \log q$ times as fast as one with a $\log p$-bit exponent). Therefore, this exponentiation determines the workload of one try.

The following lemma gives the probability that an arbitrary candidate fits.

Lemma 4.2 *The probability that a single arbitrary signature fits is* $(\frac{2}{k})^{d-2}$. *The probability that an arbitrary candidate* $\{i, i + a(k-1), i + b(k-1)\}$ *fits, is*

$$\mathrm{P}(\mathit{success}) = \left(\frac{2}{k}\right)^{3\cdot(d-2)}.$$

Proof: There are $d-2$ values of j for which the $a(j;\nu)$ are chosen at random. Each of these $a(j;\nu)$'s must take on one of two values out of the k possible values, in order to provide a fitting signature. The result follows from the fact that the $a(j;\nu)$ are chosen independently from a uniform distribution. □

With the aid of the above lemma, the required number of signatures can be calculated.

Lemma 4.3 *The expected number of signatures* N *required to obtain exactly one fitting candidate is less than* $(2k-1)\cdot(\frac{k}{2})^{d-2}$.

Proof: We prove this lemma in two steps. First, the expected number of signatures needed to have l fitting signatures is calculated, for general l. Then the probability that the lth fitting signature completes the first fitting candidate is calculated.

Suppose a large set of consecutive signatures is available. The probability that the mth element in the sequence is the lth fitting one, is (cf. Lemma 4.2)

$$p(m,l) = \binom{m-1}{l-1}\left(\left(\frac{2}{k}\right)^{d-2}\right)^{l}\left(1-\left(\frac{2}{k}\right)^{d-2}\right)^{m-l}.$$

Therefore, the expected position in the sequence of the lth fitting signature is

$$\sum_{m=l}^{\infty} m\cdot p(m,l) = l\sum_{m=l}^{\infty}\binom{m}{l}\left(\left(\frac{2}{k}\right)^{d-2}\right)^{l}\left(1-\left(\frac{2}{k}\right)^{d-2}\right)^{m-l} = l\cdot\left(\frac{k}{2}\right)^{d-2}.$$

Denote the probability that the lth fitting signature completes the first candidate in the sequence by $\mathrm{P}(l=l)$. Since the the probability that a signature fits is independent of its index, the fitting signatures are uniformly distributed over the indices modulo $k-1$. We see that $\mathrm{P}(l=l)$ is the probability that the lth signature

that fits is the third one for its index mod $k-1$, and no other index mod $k-1$ has occurred three times yet. Therefore, trivially $\mathrm{P}(l=l)=0$ if $l<3$ or $l>2k-1$. In general, the related expectation $\mathrm{E}_l l$ is hard to evaluate, but obviously $\mathrm{E}_l l < 2k-1$.

It is easy to see that the expected required number of signatures N, necessary to obtain exactly one fitting candidate, equals $N = (\frac{k}{2})^{d-2} \cdot \mathrm{E}_l l$, from which the result follows. □

From this, the number of signatures required for an attack can be calculated. This is done in the next theorem.

Theorem 4.4 *If $k < 2^{2(d-2)}$, then the expected number of consecutive signatures N', necessary for a successful attack is less than $2k \cdot (\frac{k}{2})^{d-2}$.*

Proof: N' is smaller than N times the expected number of fitting candidates that provides an independent system of equations, for this is the required number if we would start all over in case of a dependent system. (In that case we 'loose' all fitting signatures encountered so far.) That is, $N' < N/P$, where $P = \mathrm{P}(\text{independent})$, see Appendix A. The result follows. □

This theorem, finally, enables the calculation of the workload.

Theorem 4.5 *The expected number of steps of the proposed attack is less than*

$$\frac{k^{3(d-2)}}{(k-1)^2} \cdot \mathrm{E}_l \binom{l+2}{3} < 2k \cdot k^{3(d-2)}$$

for $k \geq 6$.

Proof: We calculate the number of steps in the attack, given that n' signatures are required to complete the attack. The set of the n' signatures used in the attack is partitioned into $k-1$ subsets according to the index of the signature mod $k-1$. These subsets have size approximately $n'/(k-1)$. For each triple of signatures in each of these subsets, we have checked whether this triple is a fitting candidate. So the number of candidates we tried before we had a fitting signature is

$$(k-1) \cdot \binom{\frac{n'}{k-1}}{3} \lessapprox \frac{n'^3}{6(k-1)^2},$$

see Theorem 4.4. Hence the expected number T of candidates tried before the attack is successful is upper bounded by

$$\begin{aligned} T &= \sum_{n'=3}^{\infty} \frac{n'^3}{6(k-1)^2} \sum_{l=3}^{2k-1} p(n',l) \cdot \mathrm{P}(l=l) \\ &= \frac{1}{6(k-1)^2} \sum_{l=3}^{2k-1} \mathrm{P}(l=l) \cdot \sum_{n'=3}^{\infty} n'^3 p(n',l) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{(k-1)^2} \sum_{l=3}^{2k-1} \binom{l+2}{3} \mathrm{P}(l=l) \cdot \sum_{n'=l}^{\infty} \binom{n'+2}{l+2} \left(\left(\tfrac{2}{k}\right)^{d-2}\right)^{l} \left(1-\left(\tfrac{2}{k}\right)^{d-2}\right)^{n'-l} \\
&= \frac{1}{(k-1)^2} \left(\tfrac{k}{2}\right)^{3(d-2)} \sum_{l=3}^{2k-1} \binom{l+2}{3} \mathrm{P}(l=l) \\
&= \frac{1}{(k-1)^2} \left(\tfrac{k}{2}\right)^{3(d-2)} \mathrm{E}_l \binom{l+2}{3} \\
&< 2k \cdot \left(\frac{k}{2}\right)^{3(d-2)},
\end{aligned}
$$

for $k \leq 6$. (Use $\mathrm{P}(l=l) = 0$ for $l > 2k-1$.)

There are $2^{3(d-2)}$ possible ways for a candidate to fit. At least 2^{d-2} of those possibilities provide a dependent system of equations, see Appendix A and Lemma 3.1. Therefore, we perform $2^{3(d-2)} - 2^{d-2}$ tries per candidate in the attack. From this the result follows. □

The proposed attack requires an expected number of signatures less than $2k \cdot \left(\frac{k}{2}\right)^{d-2}$ signatures, and an expected number of steps less than $2k \cdot k^{3(d-2)}$. For the parameters proposed in [4], these quantities can be calculated explicitly.

Corollary 4.6 *For $k = 8$, $d = 6$, the parameters proposed in [4], the number of signatures required to have one fitting candidate is $N \approx 2034$, the expected number of signatures in the proposed attack is $N' < 2043$ and the expected number of steps $\approx 2^{37.5} = 2.0 \cdot 10^{11}$.*

Proof: The result follows from the fact that $\mathrm{E}_l l = 7.95\cdots$ and $\mathrm{E}_l \binom{l+2}{3} \approx 143$ for $k = 8$ (the expectations can be evaluated explicitly by hand) and the proofs of Lemma 4.3 and Theorem 4.4 and 4.5. □

Hence, this attack shows that these parameters are not sufficient to reach the designated level of security: in [4] it is conjectured that an attack will take $\mathcal{O}(2^{72})$ with the suggested parameters, or in general $\mathcal{O}(k^{k(d-1)})$ steps.

Remark Comparing the proposed attack with those considered in [4], one might say that the low probability of a fitting candidate ($(2/k)^{d-2}$ instead of 1) is 'cancelled out' by a proportionally lower number of tries per candidate (2^{d-2} instead of k^{d-2}). The gain of the proposed attack is a consequence of the fact that only three fitting signatures are needed, instead of k or $k+1$.

5 Conclusions

The idea of preprocessing [4] is interesting, because by using a preprocessing algorithm, an exponentiation can be performed with the effort of a few multiplications only. However, one must be careful using preprocessing algorithms, for these leak

information. When used in the Schnorr scheme, the preprocessing algorithm, as proposed in [4], leaks too much information, for the attack presented in this paper enables retrieving the secret key from on average two thousand signatures in on average $2^{37.5}$ steps. Since the calculations for this attack can be performed in parallel to a large extent, the attack does not seem infeasible.

The proposed attack exploits the following properties of the preprocessing algorithm from [4], and seems to depend on them.

- There exists a kind of dependence —the fitting candidates— that occurs with small but nonnegligible probability.
- The required linking equations are provided by the signatures, and have nonnegligible probability of occurrence (in fact they occur with probability 1).

This suggests that the attack will become infeasible, if one of the mentioned probabilities is made substantially smaller. It seems possible to do so, even if one takes the security and efficiency requirements from [4] into account. The design of such an algorithm falls outside the scope of this paper, however —the aim of this discussion is to indicate the possibility of doing so.

We conclude that more research is needed before one can safely use this kind of preprocessing algorithm. It seems possible to design efficient preprocessing algorithms for the Schnorr scheme that avert the attack.

Acknowledgements

The author would like to thank Jean-Paul Boly and Johan van Tilburg for their constructive comments on earlier versions of this paper.

References

[1] D. Chaum, J. H. Evertse and J. van de Graaf, 'An improved protocol for demonstration possession of discrete logarithms and some generalizations', *Proc. Eurocrypt'87*, Lecture Notes in Computer Science vol. **304**, pp. 127–141, Springer Verlag, Berlin, 1988.

[2] U. Feige, A. Fiat and A. Shamir, 'Zero knowledge proofs of identity', *Proc. of STOC 1987*, pp. 210–217.

[3] J. J. Quisquater and L. S. Guillou, 'A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory', *Proc. Eurocrypt'88*, Lecture Notes in Computer Science vol. **330**, pp. 123–128, Springer Verlag, Berlin, 1988.

[4] C. P. Schnorr, 'Efficient identification and signatures for smart cards', *Proc. CRYPTO'89*, Lecture Notes in Computer Science vol. **435**, pp. 239–251, Springer Verlag, Berlin, 1990.

A Independence of equations in the attack

The attack presented in this paper is based on the possibility of solving the set of equations we obtain from Equations (1)–(3) by means of the linking equations, see Lemma 3.1. By tedious but elementary calculations, the set of equations given below is found.

$$\begin{cases} \frac{1}{2}\lambda y_i &= (\frac{1}{2}\lambda - \mu) r_{i-k} &+ r_i &+ \frac{1}{2}\lambda e_i s \\ C_a &= (\frac{1}{2}\lambda' - \mu')(-2)^{-a} r_{i-k} &+ (-2)^{-a} r_i &+ E_a s \\ C_b &= (\frac{1}{2}\lambda'' - \mu'')(-2)^{-b} r_{i-k} &+ (-2)^{-b} r_i &+ E_b s \end{cases} \tag{6}$$

where

$$C_a = \tfrac{1}{2}\lambda' y_{i+a(k-1)} + (\tfrac{1}{2}\lambda' - \mu') \sum_{l=1}^{a} (-2)^{l-a-1} y_{i+(l-1)(k-1)} + \sum_{l=1}^{a} (-2)^{l-a-1} y_{i+l(k-1)+1},$$

$$C_b = \tfrac{1}{2}\lambda'' y_{i+b(k-1)} + (\tfrac{1}{2}\lambda'' - \mu'') \sum_{l=1}^{b} (-2)^{l-b-1} y_{i+(l-1)(k-1)} + \sum_{l=1}^{b} (-2)^{l-b-1} y_{i+l(k-1)+1},$$

and

$$E_a = \tfrac{1}{2}\lambda' e_{i+a(k-1)} + (\tfrac{1}{2}\lambda' - \mu') \sum_{l=1}^{a} (-2)^{l-a-1} e_{i+(l-1)(k-1)},$$

$$E_b = \{\tfrac{1}{2}\lambda'' e_{i+b(k-1)} + (\tfrac{1}{2}\lambda'' - \mu'') \sum_{l=1}^{b} (-2)^{l-b-1} e_{i+(l-1)(k-1)}.$$

The probability that these equations are independent is the probability that the first two columns are different times the probability that the third column is not a linear combination of the first two columns, given that those are different. This latter probability follows from the fact that for any choice of λ, E_a, λ'', μ'' and $e_{i+c(k-1)}$ for $c < b$ there is exactly one $e_{i+b(k-1)} \in \mathbb{Z}_q^*$ that makes the third column a linear combination of the first two columns (for there is one value of E_b that does so). Since we also have $e_{i+b(k-1)} < 2^t$, the probability that this happens is certainly less than 2^{-t}.

Hence we have

$$P \doteq \mathrm{P}(\text{independence}) > \left(1 - \mathrm{P}(\tfrac{1}{2}\lambda - \mu = \tfrac{1}{2}\lambda' - \mu' = \tfrac{1}{2}\lambda'' - \mu'')\right) \cdot (1 - 2^{-t})$$

and

$$P < 1 - \mathrm{P}(\tfrac{1}{2}\lambda - \mu = \tfrac{1}{2}\lambda' - \mu' = \tfrac{1}{2}\lambda'' - \mu'').$$

Now λ is the sum of the powers 2^j over those j for which $a(j;i) = i - 1$, and μ is the sum of the other powers 2^j with $0 \le j \le d$. Therefore, it is not hard to see that every choice of the $a(j;i)$'s yields a unique value of $\frac{1}{2}\lambda - \mu$. Thus it holds that $\frac{1}{2}\lambda - \mu = \frac{1}{2}\lambda' - \mu' = \frac{1}{2}\lambda'' - \mu''$ if and only if $a(j;i) = a(j;i + a(k-1)) = a(j;i + b(k-1))$ (which happens with probability one quarter) for all j. From this it follows that

$$(1 - (\tfrac{1}{2})^{2(d-2)}) \cdot (1 - 2^{-t}) < P < (1 - (\tfrac{1}{2})^{2(d-2)}).$$

For $d = 6$ and $t = 72$, we have $P \approx \frac{255}{256}$.