

Algorithms for Sharing a Password

Artur Soroczuk and Marian Srebrny

Abstract

In this paper we present a new algorithm for enhanced protection of a password, personal identification number PIN or a sensitive cryptographic secret key by splitting it into shares and storing each share on a distinct computer. A password/key distributed among n computers can be reconstructed by pooling any t shares, for a given threshold $0 < t \leq n$, while knowing less than t of the shares reveals no information about the password/key whatsoever. Our algorithm uses a scheme of double recursion over n and t for generating the shares. It reveals the inductive structure of one of the very few known schemes for sharing secrets; the one based on intractability of the integer factorization. The complexity bounds are presented.

Introduction.

The area of secret sharing was pioneered by Blakley [1] and Shamir [4]. A secret to share could be any object. For us in this paper this is a sensitive cryptographic key or an access password. The reader is referred to [3] for more information on the role of secret sharing and the results known until 1999. We are particularly interested in the algorithms implementing a scheme of sharing a secret password/key introduced in [2] based on intractability of the integer factorization. In this paper we provide a new version of the algorithm of [2]. Its recursive definition is given in the following sections. Section I is devoted to the initial cases of $t = 1, 2$ and 3 , with arbitrary n . Section II takes care of the case $n = t$. In Section III the recursion step is provided. We use a scheme of double recursion over n and t .

Following [2], in the present paper the password/key is represented by a large prime number p . The shares will be defined as the products of p and

some specially arranged auxiliary pairwise distinct large primes, denoted by q with subscripts. We assume that each of those q 's is different than p . The goal is to satisfy the following condition: the greatest common divisor of any set (coalition) of the shares is equal to p if and only if the set has at least t elements.

For convenience we will be using arrays with integer entries for computing the shares. Each share will be the product of all the entries of certain array. For simplicity, we use the integers as p and the q 's here, however all this can be done using elements of any Gaussian ring. (See [3].)

We use standard terminology and notation throughout the paper. $\binom{n}{t}$ denotes Newton's binomial coefficient. By analogy, we use $\begin{bmatrix} n \\ t \end{bmatrix}$ to stand for a matrix of n rows and $k = \binom{n-1}{t-2}$ columns, and with integer entries. a_i is the i -th row of matrix a . $a_i[j]$ - the j -th entry of row a . Whenever convenient, the same can be denoted by $a[i, j]$. Let $v[j]$ denote an array of size $l = \binom{n}{t-1}$, whose entries are the large primes q_1, q_2, \dots, q_l introduced above.

Section I. Algorithm for n shares with threshold 3

1. Idea

In the case of $t = 1$ each share is defined to be just equal to the password p . In the case of $t = 2$ each share is defined as the product of p and q_i , for some $1 \leq i \leq n$. Reconstruction of the original password is executed by calculating the $GCD(pq_i, pq_j)$. In those two cases generating the shares is trivial.

Let us now introduce the scheme of generating n shares with threshold 3. That is, in such a way that knowing one or two of the shares does not allow computational reconstruction of the original password (under the assumption that all the prime parameters are chosen properly, see [2]). The basic idea is that each auxiliary prime q_i will occur as a factor in exactly two shares. We will be inserting it twice to the shares under construction in which we want it to occur. First, by copying it from the input array of the auxiliary primes supplied for use in generating distribution to a given number of shares. Second, by copying it from the shares that has been generated for distribution to a smaller number of shares.

2. Algorithm.

Input: n, p, v

Return: a_1, a_2, \dots, a_n

For $i \leftarrow 1$ to n do $a_i[0] = p$.


```

Procedure distrib1( $n$ )
For  $j \leftarrow 1$  to  $n - 1$  do  $a_1[j] = v[j]$ ;
For  $i \leftarrow 2$  to  $n$  do
  For  $m \leftarrow 1$  to  $i - 1$  do {to array  $a_i$  insert the entries of earlier
     $a_i[m] = a_m[i - 1]$  generated arrays with subscript  $i - 1$ }
  For  $s \leftarrow i$  to  $n - 1$  do {complete array  $a_i$  with uncopied
     $a_i[s] = v \left[ n(i - 1) - \frac{i(i+1)}{2} + s + 1 \right]$  so far entries of array  $v$ }

```

The input data consist of the number n of shares to be obtained, the password p to be shared, and an array of $\binom{n}{2}$ entries each of which is a large prime number. The algorithm returns arrays a_1, a_2, \dots, a_n , each of size n consisting of the factors of the particular shares. First, we plug in p into each share. Then we copy to the first share the first $n - 1$ entries of array v . Each of the following shares we get in such a way that share i consists of the entries which have already been obtained by copying the entries with subscript $i - 1$ and completing share i with uncopied so far entries of array v .

3. Example: Generating 5 shares with threshold 3.

The input array consists of 10 entries $v[i] = [q_1, q_2, \dots, q_{10}]$. The password p is copied to each share. Then q_1, q_2, q_3, q_4 are copied to the first share. Thus we get $a_1[i] = [p, q_1, q_2, q_3, q_4]$. To the second share array, where we already have p , we copy the entry of v with subscript 1. That is, q_1 . Then we complete this share with the first uncopied so far entries of array v , that is, q_5, q_6, q_7 . This time we get $a_2[i] = [p, q_1, q_5, q_6, q_7]$. The third share array consists by now of the password p , the second entries of arrays a_1 and a_2 , i.e. q_2, q_5 , and the two uncopied before entries of array v , i.e. q_8 and q_9 . Hence, $a_3[i] = [p, q_2, q_5, q_8, q_9]$. Similarly, we get a_4 from the third entries of arrays a_1, a_2, a_3 and q_{10} of array v . Hence, $a_4[i] = [p, q_3, q_6, q_8, q_{10}]$. To the last share array we copy all the fourth entries of the already formed shares. Therefore, a_5 gets the form $a_5[i] = [p, q_4, q_7, q_9, q_{10}]$.

4. Correctness

To show the correctness we have to prove: (1) for each two shares a_i and a_j $GCD(a_i, a_j) = pq_m$, for some m ; and (2) for each three shares a_i, a_j, a_k , $GCD(a_i, a_j, a_k) = p$. Equality (1) follows by the fact that whenever $i < j$ then there exists an entry in array a_i , that was copied to array a_j . Each prime q_i was inserted into arrays a_i , for $1 \leq i \leq n$, exactly twice; first when it was copied from array v , and then when copied from an earlier generated share. Therefore, it is only p that occurs in each of any three shares simultaneously.

5. Complexity

The algorithm consists of the assignments of entries of one array to another and it requires n^2 assignments, since there are n shares, each of size n .

Section II. Algorithm for n shares with threshold n

1. Idea

The algorithm takes the similar input data: the number n of shares to be generated, password p and array v of auxiliary primes. It returns arrays of size n . Each share a_i is constructed from array v by deleting its entry q_{n+1-i} .

2. Algorithm

Input: n, v

Return: a_1, a_2, \dots, a_n (the rows of distribution among n shares with threshold n)

Procedure distrib2(n)

For $i \leftarrow 1$ to n do $a_i \leftarrow v \setminus q_{n+1-i}$
 {assign to a_i all the entries of v except q_{n+1-i} }

3. Example

Generating 5 shares with threshold 5

$$a_1[i] = [q_1, q_2, q_3, q_4]$$

$$a_2[i] = [q_1, q_2, q_3, q_5]$$

$$a_3[i] = [q_1, q_2, q_4, q_5]$$

$$a_4[i] = [q_1, q_3, q_4, q_5]$$

$$a_5[i] = [q_2, q_3, q_4, q_5]$$

4. Correctness

Each entry q_i has been skipped exactly once in the process of generating the 5 shares. Thus no q_i can belong to all the 5 shares simultaneously. Hence $GCD(a_1, a_2, \dots, a_n) = 1$, as required. Elements of any smaller subset of the shares contain at least one common factor q_i , since every two shares differ at one factor, every three - at two factors, *et cetera*, every $n - 1$ shares differ at $n - 2$ factors.

5. Complexity

The algorithm consists of the assignments of one array to the others. It requires n^2 assignments, since we get n shares, each of size n .

Section III. Algorithm for n shares with threshold t

1. Idea

This time the algorithm takes positive integers $n > t > 3$ and an array v of $\binom{n}{t-1}$ entries which are the large primes q with subscripts. These primes are assumed to have been securely generated, as described in [2]. The algorithm returns a matrix of n rows and $\binom{n-1}{t-2}$ columns. Let us denote this matrix by $\begin{bmatrix} n \\ t \end{bmatrix}$. The rows of this matrix will consist exactly of the factors of the shares of the required distribution of password p among n shares with threshold t . Now, we want the algorithm to use recursive calls to the fewer distributions of the same password p among $n-1$ shares with threshold $t-1$ and among $n-1$ shares with threshold t . The initial and final conditions of this recursion are determined by threshold $t=3$ and the threshold equal to the number of shares, as taken care of in sections I and II above.

Let $a[i, j] = \begin{bmatrix} n-1 \\ t-1 \end{bmatrix}$ be a matrix of $n-1$ rows and $\binom{n-2}{t-3}$ columns consisting of the elements q_1, q_2, \dots, q_k where $k = \binom{n-1}{t-2}$, let $b[i, j] = \begin{bmatrix} n-1 \\ t \end{bmatrix}$ be a matrix of $n-1$ rows and $\binom{n-2}{t-2}$ columns consisting of the elements $q_{k+1}, q_{k+2}, \dots, q_l$ where $l = \binom{n}{t-1}$ and let $c[i, j] = \begin{bmatrix} n \\ t \end{bmatrix}$ be a matrix of n rows and $k = \binom{n-1}{t-2}$ columns consisting of the elements q_1, q_2, \dots, q_l . Define a new operation on matrices $\begin{bmatrix} n \\ t \end{bmatrix} = \begin{bmatrix} n-1 \\ t-1 \end{bmatrix} \otimes \begin{bmatrix} n-1 \\ t \end{bmatrix}$ by setting $c[i, j] = a[i, j]$, for $1 \leq i \leq n-1$ and $1 \leq j \leq \binom{n-2}{t-3}$, and setting $c\left[i, j + \binom{n-2}{t-3}\right] = b[i, j]$, for $1 \leq i \leq n-1$ and $1 \leq j \leq \binom{n-2}{t-2}$, and $c[n, m] = q_s$, for $1 \leq m \leq k = \binom{n-1}{t-2}$, where q_s is the j -th auxiliary prime used in the distribution with matrix the matrix $a[i, j]$.

2. Algorithm (generating n shares with threshold t)

Input: $n > t, t > 3, v$

Return: a_1, a_2, \dots, a_n

If $(t > 3 \vee t < n)$ then do $\begin{bmatrix} n \\ t \end{bmatrix} = \begin{bmatrix} n-1 \\ t-1 \end{bmatrix} \otimes \begin{bmatrix} n-1 \\ t \end{bmatrix}$

else

if $(t = 3)$ then distrib1(n)

if $(t = n)$ then distrib2(n)







3. Example

Generating the distribution among 6 shares with threshold 4. The distribution among 6 shares with threshold 4 uses auxiliary primes q_1, q_2, \dots, q_{20} . Let the distribution among 5 shares with threshold 3 use $q_{11}, q_{12}, \dots, q_{16}$ as its auxiliary primes. Finally, let the distribution among 4 shares with threshold 4 use $q_{17}, q_{18}, \dots, q_{20}$.

We get $\begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix} \otimes \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 4 \\ 4 \end{bmatrix} \right)$.

The table below illustrates implementation of this scheme.

q_1	q_2	q_3	q_4	q_{11}	q_{12}	q_{13}	q_{17}	q_{18}	q_{19}
q_1	q_5	q_6	q_7	q_{11}	q_{14}	q_{15}	q_{17}	q_{18}	q_{20}
q_2	q_5	q_8	q_9	q_{12}	q_{14}	q_{16}	q_{17}	q_{19}	q_{20}
q_3	q_6	q_8	q_1	q_{13}	q_{15}	q_{16}	q_{18}	q_{19}	q_{20}
q_4	q_7	q_9	q_1	q_{11}	q_{12}	q_{13}	q_{14}	q_{15}	q_{16}
q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_t

The execution of $\begin{bmatrix} 5 \\ 3 \end{bmatrix}$ is shown in red or  , $\begin{bmatrix} 4 \\ 3 \end{bmatrix}$ in yellow or  , $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$ in blue or  , $\begin{bmatrix} 4 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 4 \\ 4 \end{bmatrix}$ in yellow-blue-green or  -  -  , while $\begin{bmatrix} 5 \\ 3 \end{bmatrix} \otimes \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 4 \\ 4 \end{bmatrix} \right)$ in all the occurring colors.

4. Correctness

We need to prove equation $\begin{bmatrix} n \\ t \end{bmatrix} = \begin{bmatrix} n-1 \\ t-1 \end{bmatrix} \otimes \begin{bmatrix} n-1 \\ t \end{bmatrix}$. To this end, we show that the right-hand side matrix is a matrix of the distribution $\begin{bmatrix} n \\ t \end{bmatrix}$. Suppose $\begin{bmatrix} n-1 \\ t-1 \end{bmatrix}$ and $\begin{bmatrix} n-1 \\ t \end{bmatrix}$ are distribution matrices with auxiliary elements q_1, q_2, \dots, q_k and $q_{k+1}, q_{k+2}, \dots, q_l$ where $k = \binom{n-1}{t-2}$ and $l = \binom{n}{t-1}$. Due to the property of Newton binomial coefficient that $\binom{n-1}{t-2} + \binom{n-1}{t-1} = \binom{n}{t-1}$ one can notice that the same number of the auxiliary primes is used in both right and left hand side matrices. Following the definition of operation \otimes we construct a new matrix by inserting $\begin{bmatrix} n-1 \\ t-1 \end{bmatrix}$ and $\begin{bmatrix} n-1 \\ t \end{bmatrix}$ into its first $n-1$ rows of the distribution and the auxiliary primes of the distribution $\begin{bmatrix} n-1 \\ t-1 \end{bmatrix}$ into its n -th row. This makes the new matrix of $\begin{bmatrix} n-1 \\ t-1 \end{bmatrix} \otimes \begin{bmatrix} n-1 \\ t \end{bmatrix}$ to consist of n rows. It remains to show the treshold of this new distribution is t ; i.e., the GCD of any t rows of this new matrix is different than 1 and each t of its rows are relatively prime. The first follows by the definition of a treshold and by the fact that the constructed matrix contains a matrix of distribution $\begin{bmatrix} n-1 \\ t \end{bmatrix}$. In order to show the second we have to consider two cases. In the case when the n -th row is not among our chosen t rows, the GCD of the chosen rows is 1 since the distributions $\begin{bmatrix} n-1 \\ t-1 \end{bmatrix}$ and $\begin{bmatrix} n-1 \\ t \end{bmatrix}$ use different auxiliary prime factors and the treshold of each of them is $\leq t$. Now, consider the case when the n -th row is among our chosen t rows. Since each auxiliary prime occurring in the n -th row is different than the primes of distribution $\begin{bmatrix} n-1 \\ t \end{bmatrix}$

(which has threshold t), it suffices to show that the GCD of any $t - 1$ rows of distribution $\binom{n-1}{t-1}$ and the n -th row is equal to 1. If in this case $\text{GCD} > 1$ then the threshold of the considered distribution would have been greater than $t - 1$, since in the n -th row there are all the prime factors of distribution $\binom{n-1}{t-1}$. This would contradict the assumption that this threshold is equal to $t - 1$.

5. Complexity

The number of recursive calls in this algorithm is $\binom{n-3}{t-3}$. Notice that

$$\binom{2m}{m} = \frac{4^m}{\sqrt{\pi m}} \left(1 - \frac{1}{8m} + \frac{1}{128m^2} + \frac{5}{1024m^3} - \frac{21}{32768m^4} + O(m^{-5}) \right).$$

In the other words, $\binom{m}{\frac{m}{2}} = \Theta(2^m)$. Therefore, the worst case run time of these recursive calls gives the bound which is exponential in the number of shares to be generated. However, in the case of $\binom{n}{2}$ we get that the execution time of this algorithm is linear in n . Similarly, in the case of $\binom{n}{n-2}$. Roughly, the algorithm is feasible in all the cases with either very small or very big threshold.

References

- [1] G. R. Blakley, *Safeguarding Cryptographic Keys*, AFIPS Conference Proceedings 48(1979), pp. 313-317
- [2] W. Dobrzański, A. Soroczuk, M. Srebrny and M. A. Srebrny Jr, *Distributed Password*, Technical Report, Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland, December 1999.
- [3] A. Menezes, P. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1999.
- [4] A. Shamir, *How to share a secret*, Communications of the ACM 22 (1979), pp. 612-613.

Artur Soroczuk
 Institute of Mathematics
 and Computer Science
 Pedagogical University
 Częstochowa, Poland
 a.soroczuk@wsp.czyst.pl

and Marian Srebrny
 Institute of Computer Science
 Polish Academy of Sciences
 Warsaw, Poland
 marians@ipipan.waw.pl