Using Average Case Intractability in Cryptography

CANDELARIA HERNÁNDEZ GOYA AND PINO CABALLERO GIL
Universidad de La Laguna
38271 - La Laguna, Tenerife
SPAIN

Abstract: - One of the most critical questions in Cryptography is referred to the misunderstanding equivalence between using a difficult problem as basis of a cryptographic application and its security verification. Under this erroneous perspective, it is common using problems belonging to NP (according to the worst-case analysis) in the design stage. Afterwards the verification analysis is completely based on the complexity assumptions. However it should be bared in mind that when random generated instances are used, then many times there are fast and efficient algorithms to solve them.

This work includes the description of a new multiparty protocol devoted to the sharing of secrets whose main application is related to key management. The main particularity of this scheme is that it is based on a problem classified as DistNP-Complete under the average-case analysis, the so called Distributional Matrix Representability Problem.

Key-Words: - Cryptographic protocols, secret sharing schemes, average-case complexity, DistNP-complete problems

1 Introduction
This paper put forward the use of problems catalogued as NP-complete from the point of view of the average-case analysis as base of cryptoprotocols, concretely a Secret Sharing Scheme (SSS) with this characteristic is proposed.

When basing a cryptographic application on a determined problem the main characteristic we look for is that fixing one of its instances to find a solution will be computationally impracticable, whereas to generate pairs formed by \((\text{instance}, \text{solution})\) can be efficiently accomplished. In general, it is also sought that the verification procedure of the selected problem for any solution will be as simple as possible.

The previous reasons justify the so extended utilization of problems belonging to the \(NP\) and \(NP\)-complete worst-case classes in the design of cryptographic tools. However, with the development of the Computational Complexity Theory, and concretely thanks to the advances made on the Average-Case analysis, it has been proved that some \(NP\)-complete problems may be efficiently solved when the inputs are randomly generated [1]. One of the most immediate conclusions deduced from this fact is to choose problems whose difficulty is guaranteed by the average-case analysis as base of cryptographic applications. The problem selected for the scheme here described is the designated as Distributional Matrix Representability Problem, which possesses that characteristic [2]. This problem has been used as base problem in the design of a Zero-Knowledge Identification scheme [3] as well. Hence the primary goal of this paper is to show that it is possible to use problems that had been catalogued as difficult according to the average-case analysis as base of different cryptographic protocols.

Two versions of the previously mentioned protocol are described. The first one considers the presence of the combiner figure which is in charge of the recovery stage while in the second one the users recover it by themselves.

The structure of the present work is as follows. The next subsections are devoted to the introduction of some necessary concepts, as well as to the location of the proposal in the existing context of Secret Sharing. In section 2, the proposed Secret Sharing Scheme is described in detail. The section 3 is dedicated to point out some properties associated to the protocol as well as to clarify some of its aspects. Section 4 ends the paper with the presentation of the conclusions and some explanatory commentaries.

1.1 Average-Case Complexity
The average-case analysis is based on the concept of distributional decision problem, which is formed by a decision problem and a probability distribution defined on the set of instances [4].

In such a context the choice of the probability distribution plays an important role since it has a direct influence on the practical complexity of the problem.
In fact, it has been proved that many \textit{NP-complete} problems may be solved in polynomial time when the instances are randomly generated under certain distributions.

The distributional class analogous to \textit{NP} in the hierarchy defined by the average-case analysis is the \textit{DistNP} class. It is formed by pairs \( <D, \mu> \) containing a decision problem \( D \) belonging to the \textit{NP} class, and a probability distributions \( \mu \) that is polynomially computable.

Just as it happens in the worst-case analysis, a distributional problem is said to be average-case \textit{NP-complete} (or complete for \textit{DistNP}) if it it is in \textit{DistNP} and every distributional problem in \textit{DistNP} is reducible to it. The first problem that was catalogued as a member of this class is the distributional tiling problem. Its formal proof of membership may be found in [4]. Later, on other works, descriptions of several new average-case \textit{NP-complete} problems have been published, [2], [5], [6], [7].

The main difficulty that we have found when trying to use problems belonging to this category for practical cryptographic applications is the artificiality of their specifications. So, the principal reason why we have chosen the Distributional Matrix Representability Problem as base of the proposed Secret Shared Scheme is its unsophisticated formulation. The original problem can be defined as follows: given a matrix \( Z_A \) and a set of matrices with same size \( M = [M_1, M_2, ..., M_j] \) it should be decided whether \( Z_A \) can be expressed as a product of matrices belonging to the given set or not. This problem was shown to be indecidable for 6x6 matrices [8] and this result was also obtained later for 4x4 matrices [9]. On the other hand, a bounded version of this problem for 20x20 matrices defined in [7] is used here. In this case the instances consist of a matrix \( Z_A \), a set \( M \) of \( k \) distinct matrices and a positive integer \( n \). All the matrices intervening in the problem are square and with integer entries. Now, the question to answer may be stated as follows: is it possible to express \( Z_A \) as a product belonging to \( M^n \), where \( M^n \) is formed by all the products of \( n \) matrices from \( M \), and \( n \leq k \). The distribution considered to generate the integers \( k \) and \( n \), as well as the integer entries of the matrices is the uniform distribution. This problem may be roughly interpreted as the multiplicative matricial version of the classic knapsack problem based on the additive group of the integers.

Although the proposed scheme uses the search version of the above distributional problem, the difficulty of this version is equivalent to that of the distributional decision problem. This statement is based on the general result stated in [10], according to which, search and decision distributional problems are equivalent from the average-case analysis point of view.

1.2 Secret Sharing Schemes
Secret Sharing protocols solve practical situations in which it is necessary the distribution of a particular secret (\( S \)) among a set of users denoted by \( P = \{P_1, P_2, ..., P_n\} \). Such context may be illustrated with the problem of secret keys protection. So, the main objective of Secret Sharing Schemes is to guarantee that only pre-designated subsets of participants are able to reconstruct the secret by collectively combining their shares of \( S \). The specification of all the subsets of participants which are authorized to recreate the secret is called the access structure of the SSS. It is said to be monotone if any set which contains a subset that can recover the secret can itself recover the secret. A methodology to design SSS for arbitrary monotone access structure was given in [11] and [12]. Such results were not useful for the SSS proposed here because the access structure is not monotone.

The first Secret Sharing Schemes were independently proposed in 1979 [13], [14]. Later it was demonstrated that both proposals can be considered in a most general scheme due to their basis on the same principles of linear algebra, [15]. Many different mathematical structures such as polynomials, geometric configurations, block designs, Reed-Solomon codes, vector spaces, matroids, complete multipartite graphs, orthogonal arrays and Latin squares have been used to model secret sharing schemes.

The scheme here described is based on the generation of a secret matrix \( Z_A \) as product of matrices with the same size \( \{M_j\}_{j=1}^{n}, M_j \in M_{20}(Z), i = 1,2,...,n \) and that can be interpreted as an \( (n,n) \) threshold scheme since in this case it is indispensable the cooperation of all the participants to recover the secret.

A common model of SSS is based on two phases. In the initialization phase, a third trusted party called the dealer \( (D) \) is in charge of distributing the shares of the secret to authorized participants through a secure channel. In the reconstruction phase, the authorized participants pool their shares to reconstruct the secret, either by themselves or through another third trusted party named the combiner \( (C) \). In the initialization of the SSS proposed here the dealer will publish all the shares and the only secret information that is revealed to each participant is a pointer to a particular share jointly with the identifiers of the other parties in the same subset of the access structure. An important concept in secret sharing schemes is what is called perfectness, a SSS is said perfect when it does not
reveal any information about the shared secret to unauthorized individuals. So, from the description of the protocol it can be inferred that the SSS proposed here is perfect and its security is unconditional.

2 Describing the protocol

It is possible to establish two variants of the proposed protocol depending on whether the participation of a combiner (C) is considered or not. If its intervention is decided then it will be responsible for the reconstruction of the secret, maintaining it protected from the rest of participants.

In order to facilitate the general description of the protocol, the following four stages will be considered: set-up, distribution, verification and recovery stages.

2.1 Set-up stage

This stage consists in the generation of the secret, task that is equivalent to the generation of an instance of the underlying problem. It should be carried out under privacy by D.

This stage starts with the random generation of two integers \( k \) and \( n \) verifying the inequality \( n \leq k \).

Once done, the \( k \) matrices \( M_i \), \( i = 1, 2, \ldots, k \) with integer entries and size 20x20 are randomly generated such that two products of \( n \) matrices coincide. All the matrices form the set denoted by \( M \) and are identified through an index determined by its position in this set. The two subsets of \( n \) matrices constitute the access structure and the product of the \( n \) matrices in each subset is considered the secret information \( Z_A \).

1. D selects at random \( k, n \in Z : n \leq k \).
2. The matrices set \( M = \{ M_i, M_i \in M_{20}(Z) \}, i = 1, 2, \ldots, k \) is generated by \( D \) selecting at random the entries of each matrix.
3. The secret matrix \( Z_A \) is determined by the product of \( n \) matrices of set \( M \).

The technical infrastructure needed consists of an indexed directory, with only reading permission enabled, used to store the set \( M \). Each authorized participant can have access to this directory, but he/she only owns a pointer to his/her share.

2.2 Distribution stage

This phase requires either the existence of a secure communication channel or the use of some cipher. It is also developed by D.

The usual main problem of this stage is the bandwidth necessary to transfer the shadows. However, such as difficulty is here avoided by sending to each authorized participant user the index assigned to the corresponding matrix in the access structure.

1. \( D \rightarrow P_i : j \in M, i = 1, 2, \ldots, n \)

2.3 Verification stage

This stage allows to detect the presence of cheaters among the shadow holders and to guarantee the correctness of the secret reconstruction. The probabilistic algorithm described by Freivalds [16] and designed to obtain the verification of the product of two matrices is here used to achieve the fraud detection process. One of the characteristic of the algorithm described by Freivalds is its error probability, such a probability is bounded by \( 2^t \), where \( t \) is the number of iterations to be performed. In order to enhance the performance of the protocol, all the necessary matrices products will be carried out using the algorithm proposed in [17].

The first step of the verification stage is the generation of a random binary vector \( U \) with as many components as the dimension of the matrices intervening in the protocol. It may be done either by \( D \), who holds the secret, or through a random public generator. The generated vector \( U \) is multiplied by the secret matrix \( Z_A \) obtaining \( U' \), that is a new vector containing random linear combinations of the rows of the secret matrix. This vector \( U' \) and the random vector \( U \) will be also located in the previously mentioned directory.

At this point, a permutation of the participants set \( \{ P_{(1)}, P_{(2)}, \ldots, P_{(n)} \} \) is randomly chosen. Such a permutation establishes the order in which the verification is developed. The participant designated in first place \( (P_{(1)}) \) computes the product of his or her shadow by the random binary vector obtaining \( M_{(1)}U \) under privacy, and sends this result to the next
participant \( P(2) \). Now \( P(2) \) computes the product determined by his or her shadow and the vector provided by \( P(i), (M(2)M(i)U) \), and so on. Only if all the participants have been honest, \( P(n) \) obtains \( U' \), result that will be communicated to the others.

According to the previous process, if any of the participants forge his or her shadows then it will be detected with probability strictly greater than 1/2. If a better security level is desired, then the complete verification process may be repeated a sufficient number of times.

1. A random binary vector \( U \in Z_{20} \) is generated with a public generator.
2. D publishes \( U' = U \cdot Z_{\lambda} \)
3. A random permutation on the participants set \( \{P(1), P(2), \ldots, P(n)\} \) is publicly generated.
4. \( P(1) \rightarrow P(2): M(1) \cdot U \)
5. \( P(j) \rightarrow P(j+1): M_{(j)} \cdot [M_{(j-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot U], j = 2,3,\ldots,n \).

**Fig. 3: Algorithm SSS-DMR: Verification stage**

### 2.4 Recovery stage

In order to recover the secret every participant should access to the directory where the set \( M \) is available to obtain his or her shadow. Afterwards, and depending on the existence of a combiner \(( C)\), there are two possible actions:

- **C’s intervention is considered:** In this case, each participant sends his or her shadow to \( C \), who will reconstruct the secret.
- **C is omitted:** Then, the first participant in the permutation selects a random integer number \((x \in Z)\) and reveals the product of his or her shadow by \( x \ (M_{(1)} \cdot x) \) to the following user. The receiver multiplies his or her shadow by the information transferred and so on (taking into account that the intermediate products should be sent using secure means). Then the last user \( P(n) \) will send to \( P(n) \) the result \( Z_{\lambda} \cdot x \). The only thing that last is the recovery of \( Z_{\lambda} \), multiplying by \( x^{-1} \), action that only \( P(n) \) can accomplish. Once the secret has been recovered \( P(n) \) will broadcast it to the rest of participants. The idea behind this way to proceed is carrying out a secure multiparty computation of the secret information. In this way, the equitable distribution of computation complexity among all the participants is achieved. The parameters used should be chosen in a way that the security of the system remains guaranteed i.e. the integer \( x \) should be randomly selected and the number of matrices available in \( M \) \((k)\) must warranted the impossibility of a successful exhaustive search attack.

- **C participates:**
  1. \( P(i) \rightarrow C: M_{(i)}, i = 1,2,\ldots,n \).
- **C is omitted:**
  1. \( P(1) \rightarrow P(2): M_{(1)} \cdot x, x \in Z \).
  2. \( P(j) \rightarrow P(j+1): M_{(j)} \cdot [M_{(j-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot x], j = 2,3,\ldots,n-1 \).
  3. \( P(n) \rightarrow P(1): M_{(n)} \cdot [M_{(n-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot x]. \)
  4. \( P(1) \rightarrow P(0): [M_{(n)} \cdot M_{(n-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot x] \cdot x^{-1}, j = 2,3,\ldots,n \).

**Fig. 4: Algorithm SSS-DMR: Recovery stage**

### 3 Important considerations

Another scheme based also in matrices was proposed in [18], but there the secret is a solution to a system of linear equations. In the generation of the vector \( U \) the binary vectors with Hamming weight \( i \) should be discarded, because otherwise the vector \( U' \) will coincide with a column of the secret matrix \( Z_{\lambda} \). Thus, the cardinality of the set of possible binary vectors is \( 2^{20} \cdot 20 \).

An advantage of this SSS is that it is feasible to maintain the secret protected even from the participants in any time, so the scheme allows the reusability of the secret and shadows.

A SSS is perfect if for any subset of participants \( A \ (A \subseteq P) \) the entropy of the secret follows the expression:

\[
H(Z_{\lambda} \setminus S_{A}) = \begin{cases} 
H(Z_{\lambda}) & \text{if } A \notin \Gamma \\
0 & \text{otherwise}
\end{cases}
\]

where \( S_{A} \) represents the random variable associated to the shares owned by \( A \) and \( \Gamma \) denotes the access structure. The following theorem establishes the perfecteness of the described scheme.

**Theorem:**

The \((n,n)\) SSS-DMR threshold scheme is perfect.

**Proof**

The dealer selects randomly and independently \( n \) matrices belonging to \( M \) to determine the secret \( Z_{\lambda} \). Let’s suppose that \( n-1 \) shares have been revealed. These shares verify the following equality
\[ M(1) \cdot M(2) \cdot M(3) \cdot \ldots \cdot M(n) = Z_k \] but the secret will be obtained only if the share \( M(n) \) is successfully determined. It will be occurred with probability \( 1 / |M| = 1/k \). Hence, the secret entropy is not reduced.

As we mentioned before, a concept extensively used in SSS is the unconditional security of perfect schemes. A SSS is considered unconditionally secure against cheaters if the probability of successful cheats does not depend on the computational abilities of the cheaters. In this sense and thanks to the described verification procedure, the SSS included in this paper may be considered as unconditionally secure.

3 Conclusions

In this work a secret sharing scheme based on an average-case NP-complete problem has been proposed. The proposal does not reveal any information about the shared secret matrix to unauthorized parties, and the size of each share equals the size of the secret, so the scheme is ideal. Although it is not the first secret sharing scheme connected with combinatorial structures, the main advantage of the proposed scheme is that its security is guaranteed by its average-case complexity. The study of concrete constructions of difficult instances of the problem that are adequate according to the design of the scheme is part of a work in progress, so we hope that a forthcoming version of this work will include it.

Also a complete analysis of the security of the scheme and a comparison with other known schemes are questions that deserve further research.

References:


