International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

A New Cryptographic Scheme Utilizing the Difficulty of Big Boolean Satisfiability

Waleed Ahmad, Ali Muhammad Ali Rushdi*

Department of Electrical and Computer Engineering, Faculty of Engineering
King Abdulaziz University, P. O. Box 80204, Jeddah 21589, Saudi Arabia
**Corresponding author*: arushdi@kau.edu.sa

(Received March 24, 2017; Accepted June 10, 2017)

Abstract
A search problem may be identified as one, which requires an actual "search" for an answer or a solution. Such a problem may have no obvious method, which could be followed to determine a solution, other than to intelligently search through all candidate or potential solutions, which constitute the search space, until a satisfactory one is found. Typically, we may have an efficient way of determining whether one of the possible solutions is actually correct, but no efficient way of determining how to find a correct solution. There are many such search problems, both theoretically and practically motivated, but they all have these difficulties in common. Consider the example of RSA cryptanalysis, where we are given an integer n which is the product of two large prime numbers a and b, and we need to factor n into its factors a and b. This can be achieved by attempting (according to the sieve of Eratosthenes) to divide n by every prime integer between 2 and $\sqrt{n}$, and hence it is a special kind of a search problem in itself. Several efforts in the past aimed to translate various encryption and hashing schemes into Boolean satisfiability (SAT). The SAT problem is a computationally intractable (NP-hard) problem but relatively efficient SAT-Solvers are built having computational complexity of $2^k(1-\epsilon)$, where $0 < \epsilon < 1$ and thus can prune the search space significantly. Guided by the above concepts, we propose herein a scheme that can encrypt a message by using a 'big' Boolean function, which produces an equation that cannot be solved by the conventional SAT-Solvers and leads to a dramatic increase in the search space from 2^n to $(2^{2^m})^n$ in the worst case. Logical cryptanalysis shows that the proposed scheme is very hard to break, indeed. To the best of our knowledge, the adversary cannot reduce or prune the search space (except for shortening the task needed at every node), and is forced to traverse the whole search space. He might arrive at several candidate solutions, and has to search for clues as to which of them is the correct solution.

Keywords - Cryptography, Boolean satisfiability, Big Boolean algebras, Inverse problems.

1. Introduction

In the last two decades, there has been a remarkable progress achieved in automated solvers of the Boolean satisfiability problem, henceforth referred to as SAT-Solvers (Marques-Silva, 1999; Moskewicz et al., 2001; Eén and Sörensson, 2003). These solvers have very enhanced mathematical algorithms for checking the solution of a SAT problem. An extensive amount of research has been done during the last two decades on the SAT techniques of DLL conflict-driven clause learning and logic simplification to increase the efficiency of SAT-Solvers (Selman et al., 1993; Marques-Silva, 1999; Zhang, 1997; Zhang and Malik, 2002; Eén and Sörensson, 2003; Moskewicz et al. , 2001; Ryan, 2004). The enhanced improvement in the modern SAT-Solvers opens new panoramas for a lot of real world practical applications belonging to the traditional field of artificial intelligence and formal verification that can be successfully solved by the SAT-Solvers (Prasad et al., 2005; Boyan and Moore, 1998; Hughes and Bultan, 2008). Modern SAT-Solvers have done a tremendous job solving practical-application problems of model checking (Biere et al., 1999), scheduling, test-pattern generation in digital systems (Larrabee, 1992), design debugging and diagnosis (Smith et al., 2005), identification of functional dependencies in Boolean functions (Lee et al., 2007), technology-mapping in logic synthesis (Safarpour et al., 2006), circuit delay computation, power optimization (Sagahyroon et al., 2011), FPGA design (Hu et al., 2007), network Intrusion (Kim and

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

Lee, 2007), access control, and cryptography (Mironov and Zhang, 2006). We herein restrict the domain of application of satisfiability to that of cryptography. However, we propose a certain unusual departure from conventional satisfiability, as will be shortly see.

We digress here a little bit to present a quick glimpse of cryptography. It is well know that encryption ciphers certifying privacy and authenticity in computer security are based on practical cryptographic functions. Some features of these functions depend on unverified assumptions about the underlying complexity of the mathematical problems. If these assumptions are found to be inappropriate then a new set of theory and practical attacks are raised that refine the understanding of the problem and improve the progression of cryptography. Therefore, it is very important to study and find the strengths/weaknesses in the techniques that will resist/assist the recovery of secret information (Legendre et al., 2014). There are several cryptanalysis techniques to retrieve secret information such as linear attacks (Matsui and Yamagishi, 1992), extended linearization attacks (Courtois, 2003), Meet-in-Middle attacks (Diffie and Hellman, 1977), differential and side-channel attacks (Ambrose, 2009), as well as the use of SAT-Solvers (Eén and Sörensson, 2003). In fact, SAT-Solvers are more efficient than other techniques because they are faster, do not require additional amount of memory to store the SAT problem and offer what seems to be the best mathematical representation of ciphers. Further, SAT-Solvers have been modified so that they can be better suited for the cryptographic problem. One of such modifications is that the present SAT-Solvers can recognize the XOR operation (Soos et al., 2009), which is one of the main primitives of cryptography. Such enhancements in SAT-Solvers make the work of an adversary easier to decrypt an encrypted message. In response, we try herein to make the work of an adversary more difficult by encoding our message using 'big' Boolean algebras rather than the classical two-valued Boolean algebra (known also as switching or propositional algebra) (Rudeanu, 1974; Brown, 1990; Rushdi, 2001; 2004; 2012; Rushdi and Amashah, 2010; 2011; 2012; Rushdi and Al-Qwasmi, 2015). This increases the hardness of the problem further, as it makes it more difficult to recover the message by using the current SAT-Solvers even after undergoing appropriate modifications. Such modifications are necessary because a conventional SAT-Solver works on the principle of binary Boolean satisfiability while a 'big' Boolean algebra deals with 2^N elements, where N is strictly greater than 1, so N takes the values 4, 8, 16, 32, etc. (Rushdi and Amashah, 2011).

The rest of this paper is organized as follows. Section 2 offers an overview of Boolean Satisfiability and SAT-Solvers. Section 3 describes the basic cryptology techniques that are used. Section 4 describes our method along with an illustrative example. Section 5 concludes the paper.

2. Boolean Satisfiability and SAT-Solvers

Boolean satisfiability is the problem of deciding whether a propositional logic formula can be satisfied given suitable value assignments to the variables of the formula. Generally, the problem is represented by a propositional formula consisting of a conjunction (ANDing) of clauses (alterms), each of which comprising a disjunction (ORing) of literals, where a literal is a variable in un-complemented form X_i or in a complemented form $\bar{X}_i$. This representation of the propositional formula is called a Conjunctive Normal Form (CNF). A CNF is also known as a product-of-sums (pos) expression. A literal X_i is said to be true if a value 1 is assigned to it, while a literal $\bar{X}_i$ is said to be true if a value of 0 is assigned to X_i. So, a clause which is the disjunction of literals is said to be satisfied if any one of its literals is assigned to a true value. If each of the literals in a clause is assigned to a false value then the clause is said to be unsatisfiable and hence the whole formula is said to be unsatisfiable.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

Many SAT-Solvers have been developed to solve the Boolean satisfiability problem. Today most modern SAT-Solvers (Moskewicz et al., 2001; Eén and Sörensson, 2003; Pipatsrisawat and Darwiche, 2007a; 2007b) are based on the original DPLL algorithm (Davis and Putnam, 1960; Davis et al., 1962), which is composed of branching, unit propagation (a recursive from for what is known as Boolean constraint propagation) and backtrack searching. The DPLL algorithm performs a search process that traverses the space of variable assignments until a satisfying assignment is found, or the search space is exhausted without finding any satisfying assignment. The first major enhancement to DPLL was introduced in the GRASP solver (Marques-Silva and Sakallah, 1997; 1999), which introduced a new learning mechanism from conflicting assignments. The GRASP solver performs non-chronological backtracking by learning clauses with conflicting assignments and attaching the class of learned clauses to the formula. The second main enhancement in DPLL SAT-Solvers was introduced by Gomes et al. (1998) by implementing a restart in the search algorithm. It restarts the search space to the root level when a certain amount of conflict has been reached. The limit on the number of conflicts varies in different SAT-Solvers. The most common restarting policies include the optimal speedup policy (Luby et al., 1993) and the adaptive restart strategy (Biere, 2008). The third main enhancement was the efficient implementation of Boolean Constraint Propagation (BCP) as one of the main features of DPLL. The Chaff solver implements two-literal watching, which efficiently reduces the overhead of the BCP (Moskewicz et al., 2001). Additional enhancements which further improve the performance of modern SAT-Solvers include Conflict-Based Adaptive Branching such as Dynamic Largest Individual Sum (DLIS) (Marques-Silva, 1999) and Variable State Independent Decaying Sum (VSIDS) (Pipatsrisawat and Darwiche, 2007a; 2007b).

3. Cryptography

A cryptosystem is one of the most fundamental cryptographic protocols used in data security (Talbot and Welsh, 2006). The sender has a message text (Mesg) and an encryption function that can be used to encrypt the message into a cryptogram or cipher text (Ciph), as depicted by the equation

$$Ciph = Encr(Mesg) \tag{1}$$

The encrypted message is then sent on the (insecure) communication channel to the receiver. The receiver collects the message and decrypts it by using a decryption function to recover the message, namely

$$Mesg = Decr(Ciph) \tag{2}$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

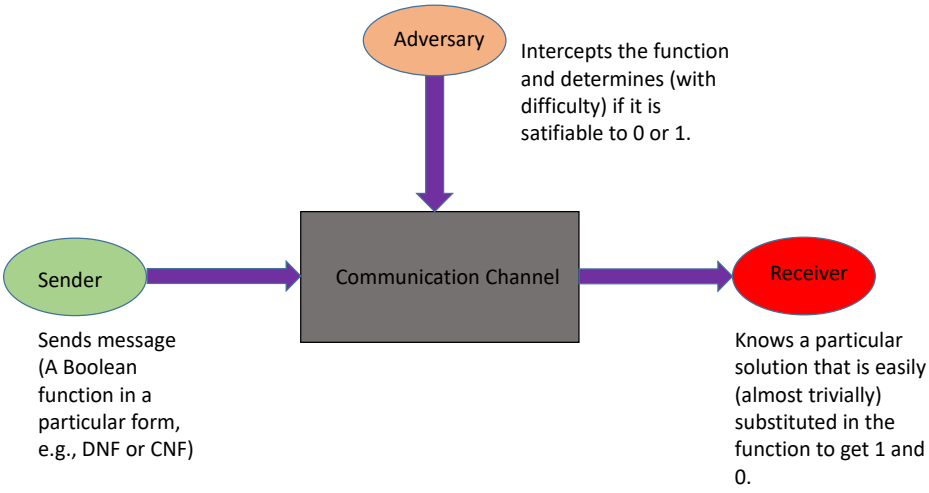

Fig. 1. Roles of the sender, receiver and adversary in an insecure communication channel with SAT employed as a cryptosystem

Fig. 1 shows a typical insecure communication channel with three players: a sender, a receiver, and an adversary. The channel employs a SAT cryptosystem. The essence of this system is that deciphering the encrypted message is very easy (even trivial) for the intended receiver (just straightforward substitution), while the same task is terribly difficult, costly and time-consuming for the adversary (amounting to solving an NP-hard SAT problem).

Usually, cryptosystems can be categorized into two types, symmetric (private-key) cryptosystems and asymmetric (public-key) cryptosystems. In a symmetric cryptosystem an otherwise secret key is shared between the sender and the receiver. The encryption function and the decryption function both depend on this shared key. The set of possible quintuples for this cryptosystem are below.

$$\{M, K, C, Encr(M, K,), Decr(C, K)\} \tag{3}$$

where M is set of all possible messages, K is the set of all possible keys, C is the set of all possible cipher texts, the $Encr$ function maps the message into a cipher text while the $Decr$ function maps the cipher text back into the original message. These two functions are described by:

$$Encr: M \times K \to C \tag{4}$$
$$Decr: C \times K \to M \tag{5}$$

The techniques of Crypto-1, DES, and AES are the most popular cryptographic techniques which work on the principle of symmetric cryptography (Afianti and Barmawi, 2015). All of these cryptosystems can be attacked by using algebraic cryptanalysis (Kamal and Youssef, 2010) and an adversary can retrieve the secret key by solving the polynomial equation through the use of linearization, Gröbner bases or SAT-Solvers. For a SAT-Solver the equations can be encoded to CNF form and the adversary must use an efficient SAT-Solver so as to decrypt the message, a very difficult task, indeed, especially when required in real time.

On the other hand, the public-key cryptosystem does not have any secret key that is shared solely between the sender and the receiver. Instead, the key is public and the cryptosystem employs the

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

concept of a one-way function, *i.e.*, a function which is very easy to compute in the forward direction used for encryption and signature verification, but very hard to compute in the reverse direction used for decryption and signature generation. The larger the size of the (public) key is, the greater is the difference between the computations required in the forward direction and in the reverse direction.

Multiplication and factorization, discrete exponential and logarithm, and cryptographic hash functions such as SHA-1 and MD4/5 are well-known examples of one-way functions used in public-key cryptographic systems. Although all these functions used in cryptosystems are said to be one-way functions, there is no theoretical proof that any of them offers perfect security or unbreakable encryption. Hence, it is possible to discover algorithms, which can attack a cryptosystem based on any of them, and hence retrieve the secret information form it. For example, one might use logical cryptanalysis, in which a cryptosystem based on factorization, discrete logarithm, SHA-1 and MD4/5 hash functions (Jovanović and Janičić, 2005; Legendre et al., 2014) is encoded to a SAT problem and then, with the help of a SAT-Solver, is decrypted by an adversary (Faizullin et al., 2009).

The (Boolean/propositional) satisfiability problem is known to be NP-hard. In fact, it is the first problem ever proven to be so (Rushdi and Ahmad, 2016). In the worst case it takes a traversal of 2^n values to solve the problem. Now, we propose a scheme for symmetric-key cryptography, in which we encrypt the message by using a 'big' Boolean algebra rather than the two-valued one. The advantage of this scheme is that it increases the difficulty of the problem and it cannot be solved by a conventional SAT-Solver without introducing appropriate modifications to it.

4. Methodology

As usual, our cryptosystem consists of two parts: encryption and decryption. In this scheme, a k-bit message which consists of 1's and 0's can be encrypted by set of m big Boolean functions of value 0, and a similar set of values 1. These Boolean functions are in the following form.

$$f_i(\boldsymbol{X}) = 0, \qquad 0 \le i \le (m-1) \tag{6}$$
$$g_j(\mathbf{X}) = 1, \qquad 0 \le j \le (m-1) \tag{7}$$

Where m represents the number of each of the f-type and g-type functions, and $[X_1\, X_2 \ldots . X_l]^T$ is an l-tuple with X_i being a variable belonging to a 'big' Boolean algebra B_{2^N}, where $N > 1$. Such an algebra is atomic with N atoms and has 2^N partially-ordered elements forming a complemented distributive lattice. Fig. 2 shows the lattice structure of B_{16}, which is the free Boolean algebra $FB(a, b)$ created by two generators a and b. Note that the elements of B_{16} are therefore all the switching functions of the two "variables" a and b. Therefore, a message of 0 is encoded by randomly choosing a function from the set of the f-type functions $f_i(\mathbf{X})$ and a message of 1 can be encoded by randomly choosing a function from the set of the g-type functions $g_j(\mathbf{X})$. For example, a 4-bit message 0010 can be encoded by the sequence $f_3(\mathbf{X})\ f_2(\mathbf{X})\ g_4(\mathbf{X})\ f_3(\mathbf{X})$, where the functions $f_i(\mathbf{X})$ and $g_j(\mathbf{X})$ are arbitrarily or randomly selected from an available pool of 0-valued and 1-valued functions.

The sent message now consists of a sequence of k functions encrypting the original bits of the form (6) and (7). Each equation in the pool of available equations (6) and (7) has a number of particular solutions, which we deliberately make arbitrarily large. However, we make sure that there is a

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

single common particular solution $\mathbf{X}_c$ for all the equations involving the 0-valued functions $f_i(\mathbf{X})$ and the 1-valued functions $g_j(\mathbf{X})$. This common solution $\mathbf{X}_c$ is entrusted securely to the intended receiver. The job of the receiver is to trivially substitute this $\mathbf{X}_c$ into the sequence of functions received, thereby converting it into the original sequence of 1's and 0's. However, it is a totally different story for the adversary, who needs to intercept the ciphered message in the first place and then try to evaluate the functions received at an arbitrarily selected value of $\mathbf{X}$. Actually the adversary will possibly have to substitute every $\mathbf{X}$ in $B_{2^N}^k$ to evaluate k functional values each belonging to B_{2^N}, and identify those values among them belonging to $\{0,1\}^k$ as candidate answers. If the adversary obtains a unique such value, it is the correct answer. Otherwise, a choice among the obtained values should be based on other consideration such as plausibility and context. Anyhow, the search space traversed always consists of $k\,2^N$ points. Typically, $N = 2^n$ ($n = 1,2,3, \ldots.$), where n is the number of generators used to generate the 'big' Boolean algebra as the free Boolean algebra $FB(a_1, a_2, a_3, \ldots. a_n)$. The search space encountered by the adversary is a much larger than that in a conventional SAT problem.

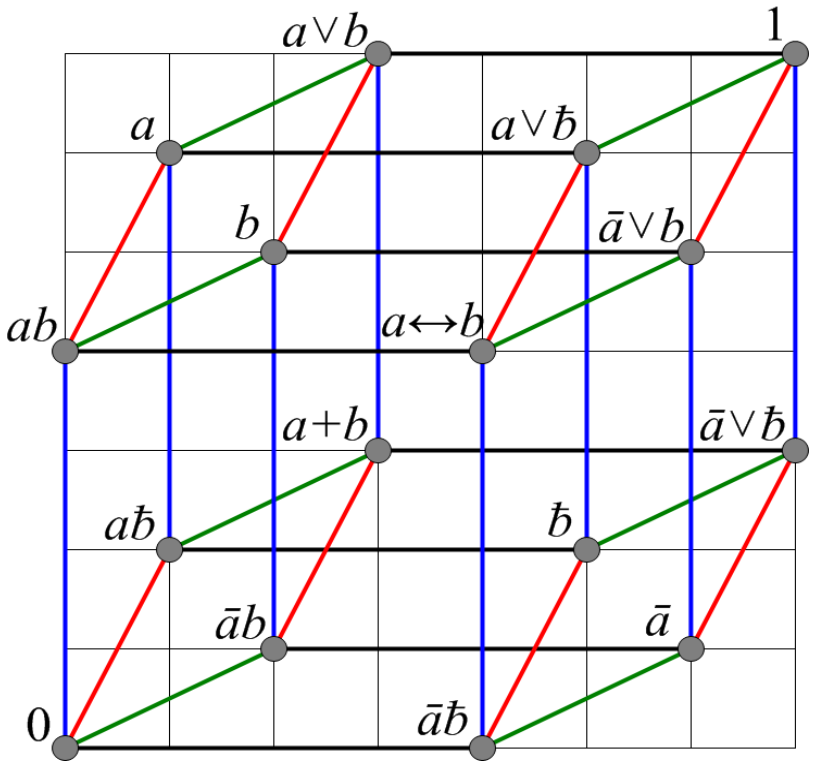

Fig. 2. A hypercube lattice indicating the partial ordering among the 16 elements of the 4-atom Boolean algebra $B_{16} = FB(a, b)$

5. Computational Illustration

We construct a toy cryptosystem having a pool of four f-type and four g-type functions $f_i(\boldsymbol{X})$: $B_2^{16} \rightarrow B_{16}$ and $g_i(\boldsymbol{X})$: $B_2^{16} \rightarrow B_{16}$ where $\boldsymbol{X} = [X_1\ X_2]^T$ and $B_{16} = FB(a, b)$, with all eight functions sharing the single common solution $\boldsymbol{X_c} = [a\ \bar{b}]^T$. Table 1 shows these eight functions, characterized by their complete sets of particular solutions, which are of variable cardinalities and

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

have an intersection equal to $\{\boldsymbol{X_c}\}$. The contents of Table 1 are kept totally secret and not entrusted to anybody other than the sender. The explicit expressions or formulas of the functions are obtained as shown in Table 2, and a sequence comprising a few of these are sent on the (insecure) communication channel so to encode a specific message. Along with these, a key $\boldsymbol{X_c}$ is shared between the sender and receiver (once and for all) *via* some sort of secure channel. Now, we need to explore the jobs required of the sender, receiver, and adversary.

5.1 Job Required of the Sender

The sender needs to replace Table 1 by Table 2, *i.e.*, to solve the inverse problem of Boolean equations (Rushdi and Albarakati, 2014). We now demonstrate this task by considering one of the f-type functions and one of the g-type functions. Let us consider the f-type Boolean equation

$$f_1(\boldsymbol{X}) = 0 \tag{8}$$

where the function f_1 is specified according to Table 1 by the set of particular solutions of (6), namely

$$\{(a,\bar{b}), (ab, 0), (ab, a\bar{b}), (ab, \bar{a}\bar{b}), (ab, \bar{b}), (a, 0), (a, a\bar{b}), (a, \bar{a}\bar{b})\} \tag{9}$$

The consistency condition associated with these particular solutions is assumed to be the identity (0=0). The function f_1 is expressed via (Rushdi and Albarakati, 2014):

$$f_1(X_1, X_2) = 0 \vee [((X_1 \oplus a) \vee (X_2 \oplus \bar{b})) \wedge ((X_1 \oplus ab) \vee (X_2 \oplus 0)) \wedge ((X_1 \oplus ab) \vee (X_2 \oplus a\bar{b})) \wedge ((X_1 \oplus ab) \vee (X_2 \oplus \bar{a}\bar{b})) \wedge ((X_1 \oplus ab) \vee (X_2 \oplus \bar{b})) \wedge ((X_1 \oplus a) \vee (X_2 \oplus 0)) \wedge ((X_1 \oplus a) \vee (X_2 \oplus a\bar{b})) \wedge ((X_1 \oplus a) \vee (X_2 \oplus \bar{a}\bar{b}))] \tag{10}$$

which is simplified as shown by sequence of natural maps (VEKMs) in Figs 3(a) – 3(c).

The final expression for f_1 could be its minimal sum

$$f_1(X_1, X_2) = ab\,\overline{X_1} \vee \bar{a}\,X_1 \vee b\,X_2 \tag{11}$$

but we deliberately write it using the (more involved but easier to transmit) minterm expansion

$$f_1(X_1, X_2) = ab\,\overline{X_1}\,\overline{X_2} \vee b\,\overline{X_1}\,X_2 \vee \bar{a}\,X_1\,\overline{X_2} \vee (\bar{a} \vee b)\,X_1\,X_2 \tag{12}$$

Now let us the consider g-type Boolean equation

$$g_1(X_1, X_2) = 1 \tag{13}$$

where the function g_1 is specified by the set of particular solutions of (7), namely

$$\{(a, \bar{b}), (1, \bar{b}), (a \vee \bar{b}, \bar{b}), (a \vee b, \bar{b}), (1, \bar{a} \vee \bar{b}), (a, \bar{a} \vee \bar{b}), (a \vee b, \bar{a} \vee \bar{b}), (a \vee \bar{b}, \bar{a} \vee \bar{b})\} \tag{14}$$

Again, under the assumption of a (0=0) consistency condition, the function g_1 is expressed via (Rushdi and Albarakati, 2014):

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

$$g_1(X_1,X_2) = [((X_1 \odot a) \wedge (X_2 \odot \bar{b})) \vee ((X_1 \odot 1) \wedge (X_2 \odot \bar{b})) \vee ((X_1 \odot a \vee \bar{b}) \wedge (X_2 \odot \bar{b})) \vee ((X_1 \odot a \vee b) \wedge (X_2 \odot \bar{b})) \vee ((X_1 \odot 1) \wedge (X_2 \odot \bar{a} \vee \bar{b})) \vee ((X_1 \odot a) \wedge (X_2 \odot \bar{a} \vee \bar{b})) \vee ((X_1 \odot a \vee b) \wedge (X_2 \odot \bar{a} \vee \bar{b})) \vee ((X_1 \odot a \vee \bar{b}) \wedge (X_2 \odot \bar{a} \vee \bar{b}))] \quad (15)$$

	X_1	
X_2	$(ab) \wedge ((ab) \vee a\bar{b}) \wedge ((ab) \vee (\bar{a}\bar{b})) \wedge ((ab) \vee \bar{b}) \wedge (a) \wedge ((a) \vee (a\bar{b})) \wedge ((a) \vee (\bar{a}\bar{b})) \wedge ((a) \vee \bar{b})$	$(\bar{a} \vee \bar{b}) \wedge ((\bar{a} \vee \bar{b}) \vee a\bar{b}) \wedge ((\bar{a} \vee \bar{b}) \vee (\bar{a}\bar{b})) \wedge ((\bar{a} \vee \bar{b}) \vee \bar{b}) \wedge (\bar{a}) \wedge ((\bar{a}) \vee (a\bar{b})) \wedge ((\bar{a}) \vee (\bar{a}\bar{b})) \wedge ((\bar{a}) \vee \bar{b})$
	$(1) \wedge ((ab) \vee \bar{a} \vee b) \wedge ((ab) \vee a \vee b) \wedge ((ab) \vee b) \wedge (1) \wedge ((a) \vee \bar{a} \vee b) \wedge ((a) \vee a \vee b) \wedge ((a) \vee b)$	$(1) \wedge ((\bar{a} \vee \bar{b}) \vee \bar{a} \vee b) \wedge ((\bar{a} \vee \bar{b}) \vee a \vee b) \wedge ((\bar{a} \vee \bar{b}) \vee b) \wedge 1 \wedge ((\bar{a}) \vee \bar{a} \vee b) \wedge ((\bar{a}) \vee a \vee b) \wedge ((\bar{a}) \vee b)$

a. $f_1(X_1, X_2)$

	X_1	
X_2	$(ab) \wedge a \wedge ((ab \vee \bar{a}\bar{b})) \wedge (a \vee \bar{b}) \wedge (a) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{b})$	$(\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{b}) \vee (\bar{a} \vee \bar{b}) \wedge (\bar{a}) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$
	$(1) \wedge (\bar{a} \vee b) \wedge (a \vee b) \wedge (a \vee b) \wedge (1) \wedge 1 \wedge (a \vee b) \wedge (a \vee b)$	$(1) \wedge 1 \wedge 1 \wedge 1 \wedge (\bar{a}) \wedge 1 \vee (\bar{a} \vee b) \wedge 1 \wedge (\bar{a} \vee b)$

b. $f_1(X_1, X_2)$

	X_1	
X_2	ab	$\bar{a}$
	b	$\bar{a} \vee b$

c. $f_1(X_1, X_2)$

Fig. 3. Three consecutive instances of the natural maps (VEKMs) of f_1

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

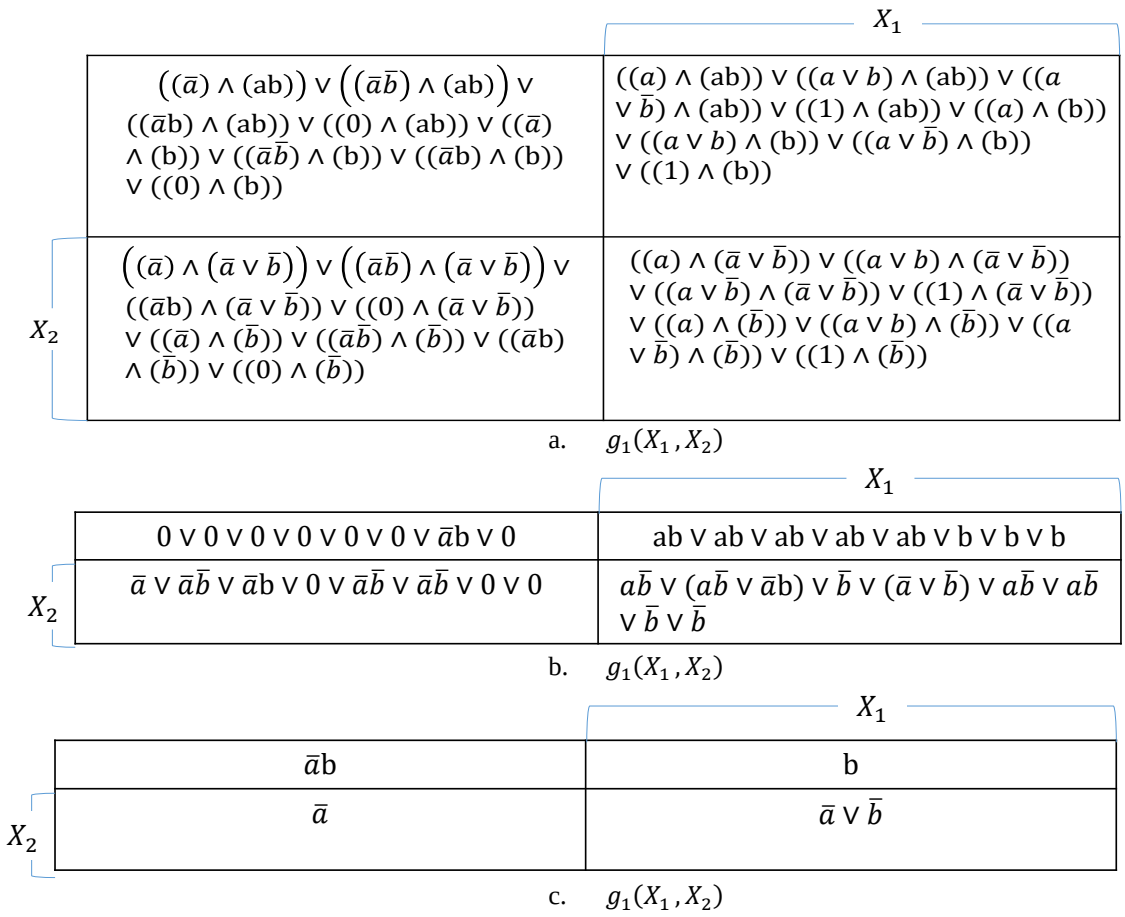

Fig. 4. Three consecutive instances of the natural maps (VEKMs) of g_1

which is simplified via the sequence of natural maps (VEKMs) in Figs. 4(a) – 4(c). The final minterm expression of g_1 is

$$g_1(X_1, X_2) = \bar{a}\, b\overline{X_1}\ \overline{X_2} \vee \bar{a}\overline{X_1}\ X_2 \vee\ b - \overline{X_2} \vee (\bar{a} \vee \bar{b})X_1\, X_2 \tag{16}$$

To encrypt the message 0010, the sender uses the vector

$$\begin{bmatrix} f_3(X_1, X_2) \\ f_2(X_1, X_2) \\ g_4(X_1, X_2) \\ f_3(X_1, X_2) \end{bmatrix} = \begin{bmatrix} a\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}\, X_2 \vee\ a\bar{b}X_1\ \overline{X_2} \vee bX_1\, X_2 \\ (a \vee \bar{b})\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}\, X_2 \vee\ \bar{a}\,\bar{b}X_1\ \overline{X_2} \vee bX_1\, X_2 \\ \bar{a}\,\overline{X_1}\ \overline{X_2} \vee \bar{b}\overline{X_1}\, X_2 \vee\ abX_1\ \overline{X_2} \vee a\bar{b}X_1\, X_2 \\ a\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}\, X_2 \vee\ a\bar{b}X_1\ \overline{X_2} \vee bX_1\, X_2 \end{bmatrix} \tag{17}$$

which might be abbreviated as

$$\begin{bmatrix} a & (a \vee b) & a\bar{b} & b \\ (a \vee \bar{b}) & (a \vee b) & \bar{a}\,\bar{b} & b \\ \bar{a} & \bar{b} & a\bar{b} & a\bar{b} \\ a & (a \vee b) & a\bar{b} & b \end{bmatrix} \tag{18}$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

5.2 Job Required of the Receiver

The receiver knows the value $\boldsymbol{X_c} = [a\ \bar{b}\ ]$ and he receives the set of functions (17) (possibly abbreviated as in (18) over the insecure channel. For him, decryption is just a matter of trivial single substitution of $\boldsymbol{X_c}$ in (17) that enables him to recover the original message 0010.

$$\begin{bmatrix} a\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1} X_2 \vee\ a\bar{b}X_1\ \overline{X_2} \vee bX_1\ X_2 \\ (a \vee \bar{b})\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1} X_2 \vee\ \bar{a}\bar{b}X_1\ \overline{X_2} \vee bX_1\ X_2 \\ \bar{a}\ \overline{X_1}\ \overline{X_2} \vee \bar{b}\overline{X_1} X_2 \vee\ abX_1\ \overline{X_2} \vee a\bar{b}X_1\ X_2 \\ a\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1} X_2 \vee\ a\bar{b}X_1\ \overline{X_2} \vee bX_1\ X_2 \end{bmatrix} = \begin{bmatrix} a\bar{a}b \vee (a \vee b)\bar{a}\bar{b} \vee\ a\bar{b}ab \vee ba\bar{b} \\ (a \vee \bar{b})\bar{a}b \vee (a \vee b)\bar{a}\bar{b} \vee\ \bar{a}\bar{b}ab \vee ba\bar{b} \\ \bar{a}\bar{a}b \vee \bar{b}\bar{a}\bar{b} \vee\ abab \vee a\bar{b}a\bar{b} \\ a\bar{a}b \vee (a \vee b)\bar{a}\bar{b} \vee\ a\bar{b}ab \vee ba\bar{b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{19}$$

5.3 Job Required of the Adversary

The adversary might try to substitute each of the 256 elements of B_2^{16}. He/she will obtain at least a single 4-tuple of only 0's and 1's. Table 3 shows all the 256 4-tuples to be obtained by the adversary. The table has 16 column indices depicting the 16 possible values for X_1 and 16 row indices specifying the 16 possible values for X_2. The 256 entries of the table indicate the result of substituting the X_1 and X_2 values in the 4-tuple $[f_3 \quad f_2 \quad g_4 \quad f_3]^T$. Of course, the adversary does not have to complete Table 3 as such, as there is no necessity to complete the computation of a vector of which one of the early elements is neither a 0 nor a 1. The adversary needs a maximum of 256 substitutions (an average of 128 substitutions) to arrive at a value $\boldsymbol{X}$ consisting solely of 0's and 1's. If he knows that this $\boldsymbol{X}$ might be wrong, he will insist on completing the 256 substitutions so as to get as many $\boldsymbol{X}$'s comprised of 0's and 1's as possible. He might be able to pick up the correct one if there is one value only that makes sense to him in a particular context. This last issue is an important feature of the proposed method, since the adversary is unable to reduce or prune the search space but is rather forced to traverse it exhaustively. As Table 3 indicates, the adversary finds himself/herself lucky as there is only a single 4-tuple of only 0's and 1's. However, as a second thought, the sender might attempt to change the used pool of functions so as to complicate matters further for adversary.

$f_1(X_1, X_2) = 0$	$(a,\bar{b})$	$(ab, 0)$	$(ab, a\bar{b})$	$(ab, \bar{a}\bar{b})$	$(a, 0)$	$(ab, \bar{b})$	$(a, a\bar{b})$	$(a, \bar{a}\bar{b})$
$f_2(X_1, X_2) = 0$	$(a,\bar{b})$	$(1, \bar{b})$	$(a \vee \bar{b}, \bar{b})$	$(a \vee b, \bar{b})$	$(a, \bar{a}\bar{b})$	$(a \vee b, \bar{a}\bar{b})$	$(a \vee \bar{b}, \bar{a}\bar{b})$	$(1, \bar{a}\bar{b})$
$f_3(X_1, X_2) = 0$	$(a, \bar{b})$	$(a, a\bar{b})$	$(ab, a\bar{b})$	$(a \vee \bar{b}, a\bar{b})$	$(1, a\bar{b})$	$(1, \bar{b})$	$(a \vee \bar{b}, \bar{b})$	$(a \vee b, \bar{b})$
$f_4(X_1, X_2) = 0$	$(a, \bar{b})$	$(a\bar{b}, \bar{b})$						
$g_1(X_1, X_2) = 1$	$(a, \bar{b})$	$(1, \bar{b})$	$(a \vee \bar{b}, \bar{b})$	$(a \vee b, \bar{b})$	$(1, \bar{a} \vee \bar{b})$	$(a, \bar{a} \vee \bar{b})$	$(ab, \bar{a}\bar{b})$	$(a\bar{b}, \bar{a}\bar{b})$
$g_2(X_1, X_2) = 1$	$(a, \bar{b})$	$(a\bar{b}, \bar{b})$	$(a, a\bar{b})$	$(a\bar{b}, a\bar{b})$				
$g_3(X_1, X_2) = 1$	$(a, \bar{b})$	$(a, \bar{a}\bar{b})$	$(ab, \bar{a}\bar{b})$	$(a \vee \bar{b}, \bar{a}\bar{b})$	$(1, \bar{a}\bar{b})$	$(1, \bar{b})$	$(a \vee \bar{b}, \bar{b})$	$(ab, \bar{b})$
$g_4(X_1, X_2) = 1$	$(a, \bar{b})$	$(ab, a\bar{b})$	$(ab, \bar{b})$	$(ab, \bar{b})$				

Table 1. A small set of Boolean functions over B_{16} that evaluate either to 0 or 1

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 1, 47–61, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.1-005

$f_1(X_1, X_2) = ab\overline{X_1}\ \overline{X_2} \vee b\overline{X_1}X_2 \vee\ \bar{a}X_1\overline{X_2} \vee (\bar{a} \vee b)X_1X_2$
$f_2(X_1, X_2) = (a \vee \bar{b})\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}X_2 \vee\ \bar{a}\bar{b}X_1\overline{X_2} \vee bX_1X_2$
$f_3(X_1, X_2) = a\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}X_2 \vee\ a\bar{b}X_1\overline{X_2} \vee bX_1X_2$
$f_4(X_1, X_2) = \bar{b}\overline{X_1}\ \overline{X_2} \vee (a \vee b)\overline{X_1}X_2 \vee\ (\bar{a} \vee \bar{b})X_1\overline{X_2} \vee (\bar{a} \vee b)X_1X_2$
$g_1(X_1, X_2) = \bar{a}\,b\overline{X_1}\ \overline{X_2} \vee \bar{a}\overline{X_1}X_2 \vee\ bX_1\overline{X_2} \vee (\bar{a} \vee \bar{b})X_1X_2$
$g_2(X_1, X_2) = (\bar{a} \vee b)\overline{X_1}\ \overline{X_2} \vee \bar{a}\bar{b}\overline{X_1}X_2 \vee\ abX_1\overline{X_2} \vee a\bar{b}X_1X_2$
$g_3(X_1, X_2) = \bar{a}b\overline{X_1}\ \overline{X_2} \vee \bar{a}\bar{b}\overline{X_1}X_2 \vee\ (a \vee b)X_1\overline{X_2} \vee \bar{b}X_1X_2$
$g_4(X_1, X_2) = \bar{a}\overline{X_1}\ \overline{X_2} \vee \bar{b}\overline{X_1}X_2 \vee\ abX_1\overline{X_2} \vee a\bar{b}X_1X_2$

Table 2. Formulas for the pertinent functions in the eight equations used

	0	$\bar{a}\bar{b}$	$\bar{a}b$	$a\bar{b}$	ab	$a \oplus b$	a	b	$\bar{b}$	$\bar{a}$	$a \odot b$	$\bar{a} \vee \bar{b}$	$\bar{a} \vee b$	$a \vee \bar{b}$	$a \vee b$	1
0	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a \odot b \\ \bar{a} \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{b} \\ \bar{a} \vee b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ \bar{a} \vee b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{b} \\ a \odot b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{b} \\ b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{b} \\ ab \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ a \odot b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ ab \\ a\bar{b} \end{bmatrix}$
$\bar{a}\bar{b}$	$\begin{bmatrix} a \\ a \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ ab \\ \bar{a} \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ \bar{a} \vee b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ 0 \\ \bar{a} \vee b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ a \odot b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ ab \\ b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ a\bar{b} \\ ab \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ 0 \\ b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ 0 \\ a \odot b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ 0 \\ ab \\ a\bar{b} \end{bmatrix}$
$\bar{a}b$	$\begin{bmatrix} a \vee b \\ 1 \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee \bar{b} \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee \bar{b} \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee \bar{b} \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee \bar{b} \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee \bar{b} \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \\ ab \\ a \oplus b \end{bmatrix}$
$a\bar{b}$	$\begin{bmatrix} a \\ \bar{a} \vee b \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ \bar{a} \vee b \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ \bar{a} \vee b \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{a} \\ 1 \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} 0 \\ \bar{a} \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} ab \\ a\bar{b} \\ a \vee \bar{b} \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} ab \\ \bar{a} \vee b \\ a \oplus b \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a\bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ \bar{a} \\ a \vee b \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a\bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a\bar{b} \\ \bar{a}\bar{b} \\ a \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} 0 \\ \bar{a} \\ a \vee b \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ a\bar{b} \\ a \vee \bar{b} \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ a\bar{b} \\ a \\ 0 \end{bmatrix}$
ab	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \odot b \\ 0 \\ a \end{bmatrix}$
$a \oplus b$	$\begin{bmatrix} a \vee b \\ 1 \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a} \\ a \vee \bar{b} \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ ab \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee \bar{b} \\ a \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ ab \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a} \vee \bar{b} \\ a \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a} \\ a \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a} \\ a \vee \bar{b} \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a} \\ a \\ \bar{a}b \end{bmatrix}$
a	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a \oplus b \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a\bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \vee \bar{b} \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a \oplus b \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a \odot b \\ a\bar{b} \\ ab \end{bmatrix}$
b	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ \bar{a} \vee b \\ 0 \\ a \vee b \end{bmatrix}$
$\bar{b}$	$\begin{bmatrix} a \\ a \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a\bar{b} \\ 1 \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} ab \\ a\bar{b} \\ a \vee \bar{b} \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ a \oplus b \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ a\bar{b} \\ a \vee b \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ ab \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ a\bar{b} \\ a \\ a\bar{b} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ a \vee b \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ a \vee \bar{b} \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ a \\ 0 \end{bmatrix}$
$\bar{a}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a}b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a}b \\ ab \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a}b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ \bar{a}b \\ ab \\ a \oplus b \end{bmatrix}$
$a \odot b$	$\begin{bmatrix} a \\ a \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ 0 \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a\bar{b} \\ \bar{a}\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ 0 \\ a \end{bmatrix}$
$\bar{a} \vee \bar{b}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \vee b \\ \bar{b} \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ a \odot b \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a}b \\ a \vee \bar{b} \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} b \\ a \oplus b \\ a \odot b \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ b \\ a\bar{b} \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ a \vee b \\ a\bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \oplus b \\ a \\ a \oplus b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ a\bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \oplus b \\ a \\ b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a}b \\ a \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a}b \\ a \vee \bar{b} \\ \bar{a}b \end{bmatrix}$	$\begin{bmatrix} \bar{a}b \\ \bar{a}b \\ a \\ \bar{a}b \end{bmatrix}$
$\bar{a} \vee b$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{a}\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ 0 \\ a \vee b \end{bmatrix}$
$a \vee \bar{b}$	$\begin{bmatrix} ab \\ a \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{a} \vee \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{a} \vee \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ \bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ a \oplus b \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ a\bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} a \\ a \\ a\bar{b} \\ a \end{bmatrix}$	$\begin{bmatrix} a \\ ab \\ a \oplus b \\ a \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ \bar{b} \\ ab \end{bmatrix}$	$\begin{bmatrix} ab \\ ab \\ a\bar{b} \\ ab \end{bmatrix}$
$a \vee b$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ ab \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ 1 \\ a\bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ ab \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ 1 \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ ab \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ \bar{a} \vee b \\ a\bar{b} \\ b \end{bmatrix}$
1	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ \bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ a \vee b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ a\bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} a \vee b \\ a \vee b \\ a\bar{b} \\ a \vee b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ a\bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ \bar{b} \\ b \end{bmatrix}$	$\begin{bmatrix} b \\ b \\ a\bar{b} \\ b \end{bmatrix}$

Table 3. All the 256 4-tuples to be obtained by the adversary

6. Discussion and Conclusions

Cryptography is the science of encrypting and decrypting data so as to allow secure transfer of information over space (transmission over a communication channel) or over time (storage within a computer memory). This paper serves as a first step towards an automated novel cryptosystem that is based on the utilization of a 'big' Boolean algebra, i.e., a finite (atomic) Boolean algebra other than the conventional two-valued one. The basic idea is to dramatically extend the search space needed in SAT-based cryptography. The adversary will not only be obliged to traverse a search space (that can be arbitrarily huge), but might end up with several candidate answers, all of which are wrong except one.

The paper demonstrated the feasibility of the proposed cryptosystem as well as the relative difficulty of breaking it. An automated version of the proposed cryptosystem is currently being built, and will be subsequently tested by subjecting it to various sorts of attacks. A forthcoming sequel of this paper will report the implementation of the proposed cryptosystem in arbitrarily big Boolean algebras, as well as techniques and results of cryptanalysis of this system.

Acknowledgement
The authors are greatly indebted to Dr. Ahmad Ali Rushdi of the University of California, Davis, California, USA for the technical help he offered in the preparation of this manuscript.

References

Afianti, F., & Barmawi, A. M. (2015). Strengthening Crypto-1 cipher against algebraic attacks. *Journal of ICT Research and Applications*, *9*(1), 88-110.

Ambrose, J. A. (2009). *Power analysis side channel attacks: the processor design-level context* (Doctoral dissertation, University of New South Wales).

Biere, A. (2008, May). Adaptive restart strategies for conflict driven SAT solvers. In *International Conference on Theory and Applications of Satisfiability Testing* (pp. 28-33). Springer Berlin Heidelberg.

Biere, A., Cimatti, A., Clarke, E. M., Fujita, M., & Zhu, Y. (1999, June). Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference* (pp. 317-320). ACM.

Boyan, J. A., & Moore, A. W. (1998, July). Learning evaluation functions for global optimization and Boolean satisfiability. In Innovative Applications of Artificial Intelligence Conference, *AAAI/IAAI* (pp. 3-10).

Brown, F. M. (1990). *Boolean Reasoning: The logic of Boolean equations*. Kluwer Academic Publishers, Boston, MA, USA.

Courtois, N. T. (2003, August). Fast algebraic attacks on stream ciphers with linear feedback. In *Annual International Cryptology Conference* (pp. 176-194). Springer Berlin Heidelberg.

Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM (JACM)*, *7*(3), 201-215.

Davis, M., LogMeIn, G., & Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, *5*(7), 394-397.

Diffie, W., & Hellman, M. E. (1977). Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, *10*(6), 74-84.

Eén, N., & Sörensson, N. (2003, May). An extensible SAT-solver. In *International conference on theory and applications of satisfiability testing* (pp. 502-518). Springer Berlin Heidelberg.

Faizullin, R. T., Khnykin, I. G. E., & Dylkeyt, V. I. (2009). The SAT solving method as applied to cryptographic analysis of asymmetric ciphers. *arXiv preprint arXiv:0907.1755*.

Gomes, C. P., Selman, B., & Kautz, H. (1998). *Boosting combinatorial search through randomization*. AAAI/IAAI, 98, 431-437.

Hu, Y., Shih, V., Majumdar, R., & He, L. (2007) Efficient SAT-based Boolean matching for heterogeneous FPGA technology mapping. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'07),* IEEE Press.

Hughes, G., & Bultan, T. (2008). Automated verification of access control policies using a SAT solver. *International Journal on Software Tools for Technology Transfer (STTT)*, *10*(6), 503-520.

Jovanović, D., & Janičić, P. (2005, September). Logical analysis of hash functions. In *International Workshop on Frontiers of Combining Systems* (pp. 200-215). Springer Berlin Heidelberg.

Kamal, A. A., & Youssef, A. M. (2010, July). Applications of SAT solvers to AES key recovery from decayed key schedule images. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on* (pp. 216-220). IEEE.

Kim, S., & Lee, J. Y. (2007). A system architecture for high-speed deep packet inspection in signature-based network intrusion prevention. *Journal of Systems Architecture*, *53*(5), 310-320.

Larrabee, T. (1992). Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *11*(1), 4-15.

Lee, C. C., Jiang, J. H. R., Huang, C. Y. R., & Mishchenko, A. (2007, November). Scalable exploration of functional dependency by interpolation and incremental SAT solving. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'07),* (pp. 227-233). IEEE Press.

Legendre, F., Dequen, G., & Krajecki, M. (2014). Logical reasoning to detect weaknesses about SHA-1 and MD4/5. *IACR Cryptology ePrint Archive*, *2014*, 239.

Luby, M., Sinclair, A., & Zuckerman, D. (1993). Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, *47*(4), 173-180.

Marques-Silva, J. P., & Sakallah, K. A. (1997, January). GRASP—a new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design* (pp. 220-227). IEEE Computer Society.

Marques-Silva, J. P., & Sakallah, K. A. (1999). GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, *48*(5), 506-521.

Marques-Silva, J. P., (1999, September). The impact of branching heuristics in propositional satisfiability algorithms. In *Portuguese Conference on Artificial Intelligence* (pp. 62-74). Springer Berlin Heidelberg.

Matsui, M., & Yamagishi, A. (1992, May). A new method for known plaintext attack of FEAL cipher. In *Workshop on the Theory and Application of Cryptographic Techniques* (pp. 81-91). Springer Berlin Heidelberg.

Mironov, I., & Zhang, L. (2006, August). Applications of SAT solvers to cryptanalysis of hash functions. In *International Conference on Theory and Applications of Satisfiability Testing* (pp. 102-115). Springer Berlin Heidelberg.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001, June). Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference* (pp. 530-535). ACM.

Pipatsrisawat, K., & Darwiche, A. (2007a). RSAT 2.0: SAT solver description. *SAT competition*, *7*.

Pipatsrisawat, K., & Darwiche, A. (2007b). A lightweight component caching scheme for satisfiability solvers. In *International conference on theory and applications of satisfiability testing* (pp. 294-299). Springer Berlin Heidelberg.

Prasad, M. R., Biere, A., & Gupta, A. (2005). A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer*, *7*(2), 156-173.

Rudeanu, S. (1974). *Boolean functions and equations*, North-Holland Publishing Company & American Elsevier, Amsterdam, the Netherlands.

Rushdi, A. M. (2001). Using variable-entered Karnaugh maps to solve Boolean equations. *International Journal of Computer Mathematics*, *78*(1), 23-38.

Rushdi, A. M. (2012). A comparison of algebraic and map methods for solving general Boolean equations. *Journal of Qassim University: Engineering and Computer Sciences*, *5(*2), 147-173.

Rushdi, A. M. A. (2004). Efficient solution of Boolean equations using variable-entered Karnaugh maps. *Journal of King Abdulaziz University: Engineering Sciences*, *15*(1), 115-131.

Rushdi, A. M. A., & Ahmad, W. (2016). Finding all solutions of the Boolean satisfiability problem, if any, via Boolean-equation solving. *Journal of King Abdulaziz University: Engineering Sciences*, 27(1), 19-34.

Rushdi, A. M. A., & Albarakati, H. M. (2014). Prominent classes of the most general subsumptive solutions of Boolean equations. *Information Sciences*, *281*, 53-65.

Rushdi, A. M. A., & Al-Qwasmi, M. A. (2015). Formal derivation of a particular input of a single AND (OR) gate in terms of its output and other inputs. *Journal of King Abdulaziz University: Engineering Sciences*, *26*(2), 51-62.

Rushdi, A. M. A., & Amashah, M. H. (2010). Parametric general solutions of Boolean equations via variable–entered Karnaugh maps. *Journal of Qassim University: Engineering and Computer Sciences*, *3*(1) 59-71.

Rushdi, A. M. A., & Amashah, M. H. (2012). Purely-algebraic versus VEKM methods for solving big Boolean equations. *Journal of King Abdulaziz University: Engineering Sciences*, *23*(2), 75-85.

Rushdi, A. M., & Amashah, M. H. (2011). Using variable-entered Karnaugh maps to produce compact parametric general solutions of Boolean equations. *International Journal of Computer Mathematics*, *88*(15), 3136-3149.

Ryan, L. (2004). *Efficient algorithms for clause-learning SAT solvers* (Doctoral dissertation, Theses (School of Computing Science)/Simon Fraser University).

Safarpour, S., Veneris, A., Baeckler, G., & Yuan, R. (2006, July). Efficient SAT-based Boolean matching for FPGA technology mapping. In *Design Automation Conference, 2006 43rd ACM/IEEE* (pp. 466-471). IEEE.

Sagahyroon, A., Aloul, F. A., & Sudnitson, A. (2011). Using SAT-based techniques in low power state assignment. *Journal of Circuits, Systems, and Computers*, *20*(08), 1605-1618.

Selman, B., Kautz, H. A., & Cohen, B. (1993). Local search strategies for satisfiability testing. *Cliques, Coloring, and Satisfiability*, *26*, 521-532.

Smith, A., Veneris, A., Ali, M. F., & Viglas, A. (2005). Fault diagnosis and logic debugging using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *24*(10), 1606-1621.

Soos, M., Nohl, K., & Castelluccia, C. (2009, June). Extending SAT solvers to cryptographic problems. In *International Conference on Theory and Applications of Satisfiability Testing* (pp. 244-257). Springer Berlin Heidelberg.

Talbot, J., & Welsh, D. J. A. (2006). *Complexity and cryptography: an introduction* (Vol. 13). Cambridge University Press.

Zhang, H. (1997, July). SATO: An efficient propositional prover. In *International Conference on Automated Deduction* (pp. 272-275). Springer Berlin Heidelberg.

Zhang, L., & Malik, S. (2002, July). The quest for efficient Boolean satisfiability solvers. In *International Conference on Computer Aided Verification* (pp. 17-36). Springer Berlin Heidelberg.