The Design of Cryptographic S-Boxes Using CSPs

Venkatesh Ramamoorthy[1], Marius C. Silaghi[1], Toshihiro Matsui[2], Katsutoshi Hirayama[3], and Makoto Yokoo[4]

[1] Florida Institute of Technology, Melbourne, FL 32901, United States of America
vramamoo@my.fit.edu, msilaghi@cs.fit.edu
[2] Nagoya Institute of Technology, Nagoya, Aichi, 466-8555, Japan
matsui.t@nitech.ac.jp
[3] Kobe University, Kobe, 657-8501, Japan
hirayama@maritime.kobe-u.ac.jp
[4] Kyushu University, Hakozaki 6-10-1, Higashi-ku, Fukuoka, 812-8581, Japan
yokoo@is.kyushu-u.ac.jp

Abstract. We use the Constraint Satisfaction Problem (CSP) framework to model and solve the problem of designing substitution functions for substitution-permutation (SP) networks as proposed by Shannon for the architecture of ciphers. Many ciphers are designed using the SP pattern, and differ mainly by two parametrized functions: substitution and permutation. The most difficult of the two is the substitution function, which has to be nonlinear (a requirement that was difficult to define and quantify). Over time, researchers such as Nyberg, Pieprzyk and Matsui have proposed various metrics of nonlinearity that make the function robust to modern attacks. Before us, people have attempted various ways to design functions that respect these metrics. In the past people hand-picked substitution tables (S-boxes) by trying various values. Recently they use difficult to analyze constructs (such as Bent functions, spectral inversion, inverses in Galois fields) whose outputs are tested for nonlinearity. While efficient, such techniques are neither exhaustive (optimal), nor did they manage to generate better substitutions than the ones hand-picked in the past.

We show that Matsui's nonlinearity requirement can be naturally modelled using CSPs. Based on a combination of existing CSP techniques and some new filtering operators that we designed specially for the new types of constraints, we manage to obtain better S-boxes than any previously published ones. The simplicity of the CSP framework and availability of general CSP solvers like ours, makes it easy for more people to design their own ciphers with easy to understand security parameters. Here we report on this new application of CSPs.

Keywords: CSP Model, S-Boxes, DES, 3DES, Nonlinearity, Linear Cryptanalysis, Differential Cryptanalysis.

1 The Cipher Design Problem

We discuss an application of the Constraint Satisfaction Problem (CSP) framework to the design of Substitution Boxes (S-Boxes) used extensively in cryptographic algorithms to secure data for confidentiality purposes.

J. Lee (Ed.): CP 2011, LNCS 6876, pp. 54–68, 2011.

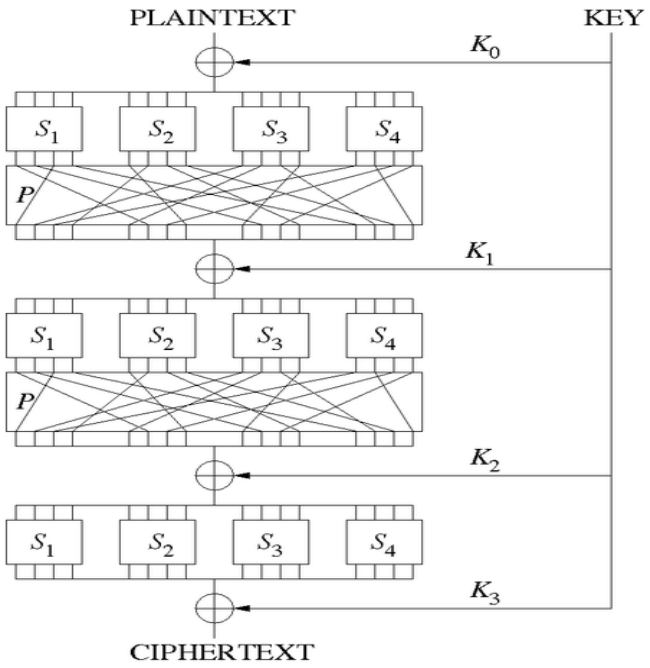

Fig. 1. Shannon's Substitution Permutation (SP) Network

Claude Shannon proposed the Substitution Permutation (SP) network, considered to be the heart of modern cryptography [22]. To transform (encrypt or decrypt) bits of data to ensure confidentiality, an SP network such as the one shown in Fig. 1, performs three steps. First, a function of the transformation key, called a *subkey*, is exclusively-ORed into the input data bits.

The second step is the one we are interested in. A substitution function $S_i : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^n}$ ($i = 1, \ldots, 4$ in Fig. 1) replaces n bits of data by another set of n bits to introduce *confusion* into the data. By $\mathbb{Z}_k$ we denote the set of residues $\{0, 1, ..., k-1\}$ modulo-k. The replacement is performed using lookup tables called Substitution Boxes, or S-Boxes.

In the third step, a permutation function P shuffles the bits to cause *diffusion* within the data. Shannon's SP network requires that each of the S-functions be invertible. The three steps constitute a *round* of the SP network and are repeated several times. Each round other than the first acts on the output of a previous round. Fig. 1 constitutes a three-round SP network, with subkeys $K_0, \ldots, K_3$ derived from the transforming key.

One of the most productive contributions to modern cryptography is Feistel's architecture [8], also referred to as the *Feistel network* in the literature. It offers a simple

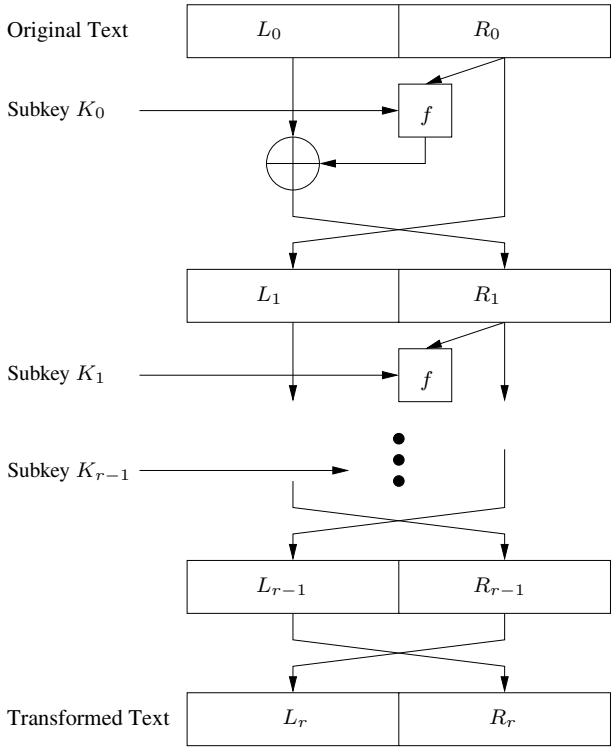

Fig. 2. Rounds of encryption / decryption in a typical Feistel Cipher

mechanism for generating countless sound[1] encryption schemes. The Feistel network, first designed by Horst Feistel and depicted in Fig. 2, is a product cipher. Each block of data being transformed is divided into two halves, a left and right half. Input bits being transformed are permuted to introduce *diffusion* into the bits.

Next, a function f applies the S-Boxes on these permuted bits to further introduce *confusion* into the data being transformed. These substitution-permutation steps form a *round* of transformation, and are repeated several times. In addition, f mixes a function of the transformation key called *subkey*, precomputed from the transformation key using a *key schedule*. Each round uses a subkey different from the others.

New sound encryption schemes on $2mk$ bits are obtained[2] for any choice of k $n \times m$ S-Boxes for any desired m and n. Feistel's cipher architecture is a variant of Shannon's substitution-permutation (SP) network [22]. When compared with Shannon's network, the soundness requirement imposes no constraint on the Feistel's substitution boxes (i.e., on the S function)[3].

[1] An encryption scheme is sound if decryption always returns the original plaintext.
[2] One for each key schedule.
[3] Shannon required that S-Boxes be invertible.

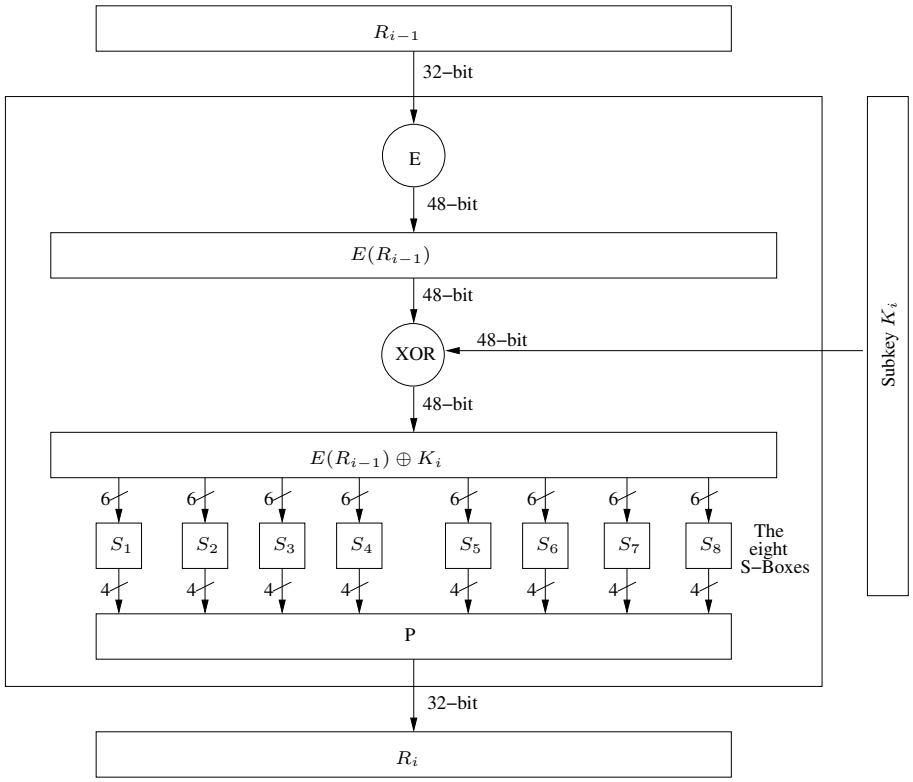

Fig. 3. One Round of encryption / decryption in 3DES using the S-Boxes

1.1 Examples of SP Networks

Blowfish, Twofish, Camellia, RC5, IBM's Data Encryption Standard (DES) [15], and the widely used *Triple*-DES (3DES), are examples of Feistel ciphers. Note that 3DES is one of the main ciphers used in protocols such as Secure Sockets Layer (SSL) and the newer Transport Layer Security (TLS). It is also employed in the Secure Shell (SSH) protocol used in applications such as `sftp` and `ssh`. Non-Feistel architectures abound in the literature such as, for example, the International Data Encryption Algorithm (IDEA) [11] and Rijndael, the current Advanced Encryption Standard (AES) [7], that employ the SP architecture.

A round of 3DES employs eight 6×4 S-Boxes numbered $S_1, S_2, \ldots, S_8$ as depicted in Fig. 3 with S_1 shown in Fig. 4. An S-box substitution of 4 bits for a 6-bit input i is obtained by indexing into the row number formed by the first and last bits of i, and the column number formed by the middle bits of i. For example, input of $45 (= 101101_2)$ to S-Box S_1 yields 1, obtained by reading the entry in row 3 $(= 11_2)$, column 6 $(= 0110_2)$ of Fig. 4.

S_1	$y_1y_2y_3y_4$															
y_0y_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Fig. 4. S-box S_1 used in 3DES

Table 1. The S-box criteria used by IBM for designing 3DES [6]

S-1	Each S-box has six bits of input and four bits of output.
S-2	No output bit of an S-box should be too close to a linear function of the input bits. (That is, if we select any output bit position and any subset of the six input bit positions, the fraction of inputs for which this output bit equals the exclusive-OR of these input bits should not be close to 0 or 1, but rather should be near $\frac{1}{2}$).
S-3	If we fix the leftmost and rightmost input bits of the S-box and vary the four middle bits, each possible 4-bit output is attained exactly once as the middle four input bits range over their 16 possibilities.
S-4	If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.
S-5	If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.
S-6	If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.
S-7	For any nonzero 6-bit difference between inputs $\Delta I_{i,j}$, no more than eight of the 32 pairs of inputs exhibiting $\Delta I_{i,j}$ may result in the same output difference $\Delta O_{i,j}$.

1.2 The Security Requirements of S-Boxes

The only part of the Feistel network that is highly nonlinear and therefore difficult to cryptanalyze, consists of the S-Boxes in the function f of Fig. 2. Thus, the security of the S-Boxes is highly important. The numbers in Fig. 4 are obtained due to S-box design. The design requirements have evolved through years of research by the cryptographic community.

For example, the S-Boxes of 3DES are so designed to satisfy the security criteria numbered **S-1**, **S-2**, and so on [6], which are listed in Table 1. These criteria were classified and eventually, revealed [6] only after reporting of results of differential cryptanalysis by Biham, et. al [4] and linear cryptanalysis by Matsui [13].

Subsequently, security requirements such as maximum nonlinearity, minimum autocorrelation, the strict avalanche criterion (SAC), the bit independence criterion (BIC), highest dynamic distance, and several others, have found their way into the design principles to enhance S-box security [18,14].

2 Rationale of the CSP Approach to S-Box Design

The first S-Boxes for Feistel ciphers were designed by hand. Early security attacks have propelled the research for guidelines (i.e., requirements) that avoid known vulnerabilities. These requirements prove to be so difficult to achieve, to the point where it is said [15] that the 3DES designing team dropped guards when hand-picking their last S-box (given the fact that their last S-box is susceptible to attacks from differential cryptanalysis [4]). Some subsequent proposals build keyed one-time usage S-Boxes dynamically. This avoids the need of hand-building them, but results in expensive start-up times at encryption and decryption (e.g., Blowfish, Twofish [21]).

Some of the criteria used for design of static S-Boxes (such as maximum nonlinearity and minimum autocorrelation) are defined using numerical satisfaction functions where the absolute satisfaction appears unreachable[4]. Therefore the design process becomes an *optimization* procedure where the satisfaction of the criteria is maximized.

The approach on which we build here, is to automatically generate the needed S-Boxes based on the relevant security criteria. S-box generation approaches can be classified as: random generation of S-Boxes, random generation-and-testing of S-Boxes, human-made S-Boxes, and math-made S-Boxes [23].

An exhaustive generation of all possible S-Boxes followed by validating them using security criteria known at that time has been tested for 4×4 S-Boxes as reported in [1]. Among the existing math-made S-box generation schemes, a number of approaches pre-load bent functions, often constructing them bit by bit, into an S-box entry, and testing the entry against design criteria. For example Mister, et.al [14] loads a bent function bit-by-bit into an S-box entry and tests for its nonlinearity and highest dynamic distance. Adams, et.al [2] test combinations of the bits of a 4×4 S-box entry against design criteria such as nonlinearity, strict avalanche and output bit independence. O'Connor [16] combinatorially analyzes the bit-by-bit approach and shows that there are practical limits up to which this scheme can generate S-boxes efficiently. Pieprzyk, et.al construct $n \times n$ bijective S-Boxes by focussing only on nonlinearity requirements [18]. Gupta, et.al [9] construct $n \times m$ S-Boxes in two ways – one, by modifying a technique by Zhang and Zheng and the other, by using a sharpened version of Maiorana-McFarland technique to construct nonlinear resilient functions.

Recently, evolutionary approaches using local search have been applied to obtain S-Boxes satisfying security requirements [12]. The approaches employ hill climbing, simulated annealing and spectral inversion. A typical approach generates a fully-filled S-box that is "approximate" in that not all criteria are satisfied, and entries in the S-box are adjusted to guide the search towards criteria satisfaction. None of these approaches has the elegance of a CSP model. They did not manage to produce S-Boxes of a higher quality than the hand-made ones. Neither can they be so simply extended with new constraints, nor do their efficiency benefit immediately from advances in general computing techniques. In contrast, in our work, we employ *automatic generation* of S-Boxes using CSPs.

[4] This observation of unreachability is supported by our experiments, as well as by the existing S-box selections of various ciphers.

3 The CSP Approach

Our model of the problem is based on the following definition of a CSP:

Definition 1. *A Constraint Satisfaction Problem (CSP) is a tuple* $(X, D, \mathcal{C})$ *where* X *is a set of* variables*,* D*, a set of* domains *of each variable in* X*, and* $\mathcal{C}$ *is a set of* constraints *between variables in* X*, all of which are required to be satisfied.*

3.1 Notations

For a number x, we use $|x|$ to denote its absolute value. If S is a set, then $|S|$ represents its cardinality (number of elements in S). Whenever a set is written with braces, its cardinality is written with a $\#$ preceding the set itself, such as $\#\{a_0, a_1, a_2, \ldots a_{k-1}\}$. The symbols $\cdot$ and $\oplus$ represent, respectively, the bit-wise AND and exclusive-OR (XOR) operations on two identical-sized bit patterns. Bit pattern $\bar{x}$ denotes the one's-complement of x. A linear combination of Boolean variables $x_0, x_1, x_2, \ldots, x_{k-1}$, is given by the expression

$$\bigoplus_{i=0}^{k-1} a_i \cdot x_i = a_0 \cdot x_0 \oplus \ldots \oplus a_{k-1} \cdot x_{k-1} \tag{1}$$

where the a_i's are Boolean coefficients. A linear Boolean function $L_\omega(x)$ on an n-bit input $x = x_0 \ldots x_{n-1}$ defined by an n-bit selector $\omega = \omega_0 \ldots \omega_{n-1}$ is computed [5] as:

$$L_\omega(x) = \omega_0 \cdot x_0 \oplus \ldots \oplus \omega_{n-1} \cdot x_{n-1} = \bigoplus_{i=0}^{n-1} \omega_i \cdot x_i \tag{2}$$

1 The *parity* $P(x)$ of an n-bit pattern $x = x_0 x_1 \ldots x_{n-1}$ is equal to the exclusive-OR of the bits in x, that is, $P(x) = x_0 \oplus x_1 \oplus \ldots \oplus x_{n-1}$. Using these facts, we derive the following property of $L_\omega(x)$:

Property 1. $L_\omega(\bar{x}) = L_\omega(x) \oplus P(\omega)$

Some existing criteria are based on the concepts of *Hamming weight* and *Hamming Distance*. The Hamming weight of a given bit-pattern u, denoted by $wt(u)$, is defined as the number of 1's in u. Two n-bit numbers x and y *differ* by an amount equal to $x \oplus y$. The Hamming Distance between x and y is the minimum number of changes to be made to x to obtain y, and is equal to $wt(x \oplus y)$.

	$y_1y_2y_3y_4$							
y_0y_5	0	1	2	3	...	13	14	15
0	x_0	x_2	x_4	x_6	...	x_{26}	x_{28}	x_{30}
1	x_1	x_3	x_5	x_7	...	x_{27}	x_{29}	x_{31}
2	x_{32}	x_{34}	x_{36}	x_{38}	...	x_{58}	x_{60}	x_{62}
3	x_{33}	x_{35}	x_{37}	x_{39}	...	x_{59}	x_{61}	x_{63}

Fig. 5. The relation between the selected variables and a 6×4 S-box

Each security criteria is now implemented by a set of corresponding constraints, taking IBM's criteria (Table 1) applied to $n \times m$ S-box design as an example. IBM's Criterion **S-1** is implicit in the choice of variables. Criteria **S-4** through **S-6** are formulated as binary constraints. Criteria **S-2** and **S-7** have to be implemented using n-ary constraints and **S-3** generates `Alldiff` constraints. The constraints for **S-2** are the most involved and are first presented (Sec. 3.3).

3.2 Variables and Domains for an $n \times m$ S-Box

To model, using constraints, an $n \times m$ S-box (i.e., on n-bit inputs), we propose to use 2^n variables. The i^{th} variable will be denoted x_i, $0 \leq i < 2^n$. Each x_i specifies the output of the S-box for input i. The set of variables for the CSP is the set $X = \{x_0, x_1, \ldots, x_{2^n-1}\}$. Since the output of an S-box is m bits long, the domain of each variable x_i is defined as $x_i \in \{0, 1, \ldots, 2^m - 1\}, 0 \leq i < 2^n$.

In the example for 3DES, where $n = 6$ and $m = 4$, there are 64 different S-box input values. Let the corresponding 64 output values be represented by variables $x_0, x_1, \ldots x_{63}$. Since each output is 4 bits, the domains are defined by $x_i \in \{x : 0 \leq x \leq 15\},\ 0 \leq i \leq 63$. Using these variables, a 6×4 S-box such as the one for 3DES is organized as shown in Fig. 5, as addressed by incrementing the input. In Fig. 5, 6-bit inputs $i, 0 \leq i \leq 63$ are represented as bit-patterns $y_0y_1y_2y_3y_4y_5$ for clarity.

3.3 The Nonlinearity Constraint S-2

The rationale behind IBM's criterion **S-2** (see Table 1) is to ensure that an S-box is highly nonlinear. Matsui's work on linear cryptanalysis [13] uses a table called the Linear Approximation Table that records the counts of linear combinations of all subsets of input and output bits, for a particular S-box. Consider an $n \times m$ S-box, i.e., that for any n-bit input $i = i_0 \ldots i_{n-1}$ yields the m-bit output $x_i = x_{i_0} \ldots x_{i_{m-1}}$. The linear combinations to be checked for equality are obtained by selecting bits in i and x_i using selectors a and b respectively, where $0 \leq a < 2^n$ and $0 \leq b < 2^m$. For a given S-box Φ with all variables in X, let us define $N_X^\Phi(a, b)$ as follows:

$$N_X^\Phi(a, b) = \#\{i : L_a(i) = L_b(x_i); a, i \in \mathbb{Z}_{2^n}; b, x_i \in \mathbb{Z}_{2^m}\} \tag{3}$$

where $L_\omega(x)$ is defined in (2). The minimum value of $N_X^\Phi(a, b)$ is zero and the maximum value is 2^n. Matsui [13] considered the general case when b is not a power of 2, corresponding to a criterion **S-2′** that is stricter than **S-2**.

Given an $n \times m$ S-box Φ' and $X' \subseteq X$, let us define $N_{X'}^{\Phi'}(a, b)$ as follows:

$$N_{X'}^{\Phi'}(a, b) = \#\{i : L_a(i) = L_b(x_i); x_i \in X'; a \in \mathbb{Z}_{2^n}; b, x_i \in \mathbb{Z}_{2^m}\}$$

A Measure of Nonlinearity. For selectors a and b defined as above, let $p(a, b)$ denote the fraction of cases when $L_a(i) = L_b(x_i)$, computed as:

$$p(a, b) = \frac{N_X^\Phi(a, b)}{2^n} \tag{4}$$

If $p(a,b)$ is equal to 1, this indicates that the linear combination of the output bits selected by b equals a linear combination of the input bits selected by a, i.e., $\forall i, L_a(i) = L_b(x_i)$. If $p(a,b)$ is equal to zero, then the linear combination of the output bits selected by b is never equal to the linear combination of input bits selected by a. **S-2** stipulates that $p(a,b)$ should be near $\frac{1}{2}$, i.e. $|N_X^{\Phi}(a,b) - \frac{|X|}{2}|$ should be as close to zero as possible.

The Score of an *S*-box. The ideal case where $N_X^{\Phi}(a,b) - \frac{|X|}{2}$ is zero for all selector-pairs (a,b), has so far not been attained in the literature for common cryptosystems. The most effective linear approximation of a 3DES S-box is obtained if, for some a and b, $|N_X^{\Phi}(a,b) - \frac{|X|}{2}|$ is maximal. To reduce the weakest point of the S-box, we use the so called *effectiveness* of linearization [17] of an S-box Φ as its score:

$$\sigma_X(\Phi) = max\{|N_X^{\Phi}(a,b) - \frac{|X|}{2}| : 1 \leq a < |X|; 1 \leq b < |D|\} \tag{5}$$

An S-box with a smaller score is considered better. For our search heuristics we proved and use the following property:

Property 2. The score $\sigma_X(\Phi)$ of a complete assignment Φ does not change if all of its assigned values are replaced by their one's-complements, into an assignment $\bar{\Phi}$.

The score $\sigma_{X'}, X' \subseteq X$, of a partially-filled $n \times m$ S-box Φ' is defined as follows:

$$\sigma_{X'}(\Phi') = max\{|N_{X'}^{\Phi'}(a,b) - \frac{|X|}{2}| : 1 \leq a < \frac{|X|}{2}; 1 \leq b < |D|\} \tag{6}$$

The Constraint for S-2. The criteria **S-2** leads to a soft constraint that minimizes $\sigma_X(\Phi)$. When implemented as a hard constraint for a threshold τ, it has the form:

$$\sigma_X(\Phi) \leq \tau \tag{7}$$

The maximum value of $\sigma_X(\Phi)$ is equal to $\frac{|X|}{2}$, which is attained when the S-box output bits are given by a linear combination of its input bits.

This constraint is not implemented using an extensional representation. Rather, a specialized function is added to the solver that works with a 2^{n+m} size storage, replicated at each level in the search tree. This results in a total space requirement of 2^{2n+m} bytes. For 3DES-size boxes the constraint requires $64kB$. This heuristic will be referred to as $H_S^{\Phi,\tau}$.

An Incomplete, Incremental Heuristic for S-2 using Partial Assignments. The incomplete constructive search heuristic $H_I^{\phi,\tau}$ is based on abandoning partial assignments larger than ϕ variables, with score exceeding a threshold τ. For example, for 6×4 S-box generation with $H_I^{48,16}$, partial S-Boxes having 48 instantiated variables will be rejected if they do not have entries with $N_{X'}^{\Phi'}(a,b)$ of at least 16. The $H_I^{48,16}$ heuristic with MAC yielded a large number of S-Boxes having score 8, (better than those in 3DES) whose retrieval using other search techniques required a lot of computation time.

Projections of n-ary constraints to partial assignments. The following property of a partial assignment allows for projection of (Eq. 7) into lower-arity constraints.

Property 3 (Projections). A partial assignment Φ' with values for variables in X', $X' \subseteq X$, cannot be extended to a solution with score better than a threshold τ if the following inequality is not satisfied:

$$|X'| - \tau - \frac{|X|}{2} \leq \max_{a,b} N_{X'}^{\Phi'}(a,b) \leq \frac{|X|}{2} + \tau \tag{8}$$

A *complete, incremental heuristic* that uses this property will be referred to as $H_C^{\phi,\tau}$.

S-3 (see Table 1). Fixing the leftmost and rightmost input bits $y_0 y_5$ to any of the possible four combinations, selects one of four subsets of the variables. To formulate constraints for **S-3**, all we require is that no two output variables, in each subset, should be equal. The inequations are directly expressible as `Alldiff` constraints [19], [10]:

$$\texttt{Alldiff}(x_0, x_2, x_4, ..., x_{30})$$
$$\texttt{Alldiff}(x_1, x_3, x_5, ..., x_{31})$$
$$\texttt{Alldiff}(x_{32}, x_{34}, x_{36}, ..., x_{62})$$
$$\texttt{Alldiff}(x_{33}, x_{35}, x_{37}, ..., x_{63})$$

3.4 Constraints for Criteria S-4 to S-6

For criteria **S-4**, **S-5** and **S-6**, consider any two n-bit inputs i and j and their corresponding m-bit outputs $x_i, x_j \in D$, of a 3DES S-box S.

S-4 *"If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits."* This requirement is expressible in First-Order Logic as:

$$(\forall i)(\forall j)(0 \leq i < j < 2^n) \wedge wt(i \oplus j) = 1 \Rightarrow wt(x_i \oplus x_j) \geq 2 \tag{9}$$

For 3DES, each variable will participate in exactly 6 such binary constraints (one for each bit), generating 192 binary constraints. For an $n \times m$ S-box, the number of binary constraints for criterion **S-4** is equal to $n \times 2^{n-1}$.

S-5 *"If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits."* The fact that n-bit inputs i and j differ in the two middle bits implies that the 6-bit difference is exactly equal to $3 \cdot 2^{\frac{n}{2}-1}$ when n is even. **S-5** is expressible in First-Order Logic as:

$$(\forall i)(\forall j)(0 \leq i, j < 2^n) \wedge (i \neq j) \wedge (i \oplus j = 3 \cdot 2^{\frac{n}{2}-1} \Rightarrow wt(x_i \oplus x_j) \geq 2 \tag{10}$$

For 3DES, this results in 32 binary constraints, each input (S-box entry) participating in exactly one such binary constraint. For an $n \times m$ S-box, the number of binary constraints for criterion **S-5** is equal to 2^{n-1} when n is even.

S-6 *"If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same."* The fact that n-bit inputs i and j differ in their first two bits and are identical in their last two bits, implies that the input-difference $(i \oplus j) \wedge 3(2^{n-2}+1)$ is exactly equal to $3 \cdot 2^{n-1}$ when $n \geq 4$. **S-6** is expressible in First-Order Logic as:

$$(\forall i)(\forall j)(0 \leq i < j < 2^n), [(i \oplus j) \wedge 3(2^{n-2}+1)] = 3 \times 2^{n-2} \Rightarrow x_i \neq x_j \quad (11)$$

For 3DES, each variable is involved in 4 such binary constraints (one for each possible combination of the two middle input bits), resulting in a total of 128 new binary constraints. For an $n \times m$ S-box, the number of binary constraints for criterion **S-6** is equal to 2^{2n-5} when $n \geq 4$.

Total Number of Binary Constraints. For the $n \times m$ S-box design problem using this framework, the total number of binary constraints, obtained by adding the three results for **S-4**, **S-5** and **S-6**, is equal to $2^{n-1}(2n-1)$ when $n \geq 4$. One can observe the independence of the total number of constraints on m. For 3DES, this works out to 352 constraints. Also, no binary constraints are observed to contain the same two variables.

3.5 The Global Constraint S-7

S-7: *"For any nonzero 6-bit difference between inputs $\Delta I_{i,j}$, no more than eight of the 32 pairs of inputs exhibiting $\Delta I_{i,j}$ may result in the same output difference $\Delta O_{i,j}$."*

Let $O_7 = \{(x_i, x_{2^n-1-i}) : 0 \leq i < 2^{n-1}\}$ be the set of pairs of variables corresponding to pairs of subscripts $(i, 2^n-1-i)$ of those n-bit inputs to an $n \times m$ S-box that differ by all n bits with $|O_7| = 2^{n-1}$. **S-7** applies to m-bit differences $d = x_i \oplus x_{2^n-1-i}$.

Let $f : \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_{2^{n-1}}$ denote a *count* function, where $f(d)$ is the *frequency of occurrence* of an m-bit number $d = x_i \oplus x_{2^n-1-i}$ where $(x_i, x_{2^n-1-i}) \in O_7$, $0 \leq i < 2^{n-1}$, and $0 \leq d < 2^m$. Note that $\Sigma_{i=0}^{2^{n-1}-1} f(x_i \oplus x_{2^n-1-i}) = 2^{n-1}$.

According to **S-7**, at most eight elements in O_7 should evaluate to the same m-bit difference d. **S-7** is modeled as a global, n-ary, Boolean constraint in the following way:

$$\bigwedge_{i=0}^{2^{n-1}-1} (f(x_i \oplus x_{2^n-i-1}) \leq 8) \quad (12)$$

This n-ary global constraint is not straightforwardly decomposable into smaller arity constraints. After assigning x_i, if the count of a given AND-term in Eq. 12 equals 8, values from domains of not yet assigned variables that would further increase this count are removed (as they violate Eq. 12).

4 The Advantages of the CSP Approach

The CSP solver helped us find S-Boxes of quality better (with respect to the standard 3DES security metrics) than that of any other published S-box.

The solver we use is an implementation of Maintenance of Arc Consistency (MAC) [20] with AC2001 [3], as discussed in Section 3. The CSP solution was built starting

Table 2. Statistics of 4×2 S-Boxes generated by our CSP framework to satisfy combinations of 3DES criteria

Constraint Combinations	Time (seconds)	# of S-Boxes	Score-wise breakup			
			Score 8	Score 6	Score 4	Score 2
No constraints	136228.906250	4294967296	3931260	517882560	3496729600	276422720
S-3 only	35.029600	331776	11904	153600	166272	0
S-4 only	0.000089	4	4	0	0	0
S-5 only	6.410940	65536	7936	45056	12544	0
S-6 only	13214.516602	429981696	2103616	91728896	323934912	12214272
S-3, **S-5**	0.433693	4096	384	2048	1664	0
S-3, **S-6**	5.224500	46656	6240	22272	18144	0
S-5, **S-6**	2.085620	20736	4160	13312	3264	0
S-3, **S-5**, **S-6**	0.165739	1600	224	768	608	0

from an existing generic C++ implementation, to which we have added modules for dynamically checking and propagating the decompositions of the global constraints.

The generation of the constraints and the development of the related theory and involved filtering operators are the main topic of the PhD thesis of the first author, and took approximately 2 to 3 years to refine.

Experiments are being run with the final version for approximately one year. Users of the system that solely plan to design ciphers using the standard security criteria in Table 1 do not need to thoroughly understand the workings of CP solvers. Most extensions with additional constraints could also be performed with little CP knowledge, except if new filtering operators for those constraints are desired.

We have used the system to generate S-Boxes of different sizes, such as 4×2, 5×3 and 6×4 that resemble those used in 3DES. We have tuned the solver by trying various heuristics for criteria **S-2** and **S-7**. One of these heuristics instantly generated 6×4 S-Boxes that are of quality better than those published so far, with respect to Matsui's nonlinearity metric.

4×2 S-box Generation: The smallest S-Boxes we have encountered in the literature is an educational example of 4×2 [23]. Not all criteria in Table 1 apply to S-Boxes that are not of size 6×4. In the 4×2 case, our solver has proven that it is even impossible to generate S-Boxes that respect the criteria that apply. The CSP approach generated 4×2 S-Boxes when some of the conditions are relaxed. Table 2 displays the results on combinations of satisfied criteria.

5×3 S-box Generation: The complete CSP solver is able to explore the entire search space for generating 5×3 S-Boxes (32 variables, each with domain $\{0, 1, \ldots, 7\}$). Criteria **S-5** and **S-6** had to be relaxed since the original version did not apply to this size. Table 3 shows generation times and number of S-Boxes for various scores, with a total of 32,640 S-Boxes generated. The optimum score possible for 5×3 S-Boxes is 8.

6×4 S-box Generation: We have used the three heuristics $H_S^{\phi,\tau}$, $H_I^{\phi,\tau}$ and $H_C^{\phi,\tau}$ for generation of S-Boxes of this size. For $H_I^{\phi,\tau}$, we have fixed thresholds $\tau = 16, 10$ and

Table 3. The scores of obtained 5×3 S-Boxes, with criteria **S-5** and **S-6** relaxed

Total time (seconds)	Total number of S-Boxes	Score-wise breakup		
		Score 16	Score 12	Score 8
14.2659	32,640	25728	3456	3456

Table 4. Solver Performance Using Complete Heuristics, with S-box threshold $\tau = 16$

Time (hrs)	Non-incremental		Incremental	
	$r^{(6\times4)} \times 10^{49}$	S-box Count	$r^{(6\times4)} \times 10^{49}$	S-box Count
1	1.198	4	206,990	38,124
2	21.725	14	978,520	54,725
3	42.091	15	999,560	93,523
4	42.091	26	1,083,100	104,904
5	61.340	40	1,342,900	127,111

0	3	5	6	9	10	15	12	7	4	14	13	2	1	8	11
3	0	6	5	10	9	12	15	4	7	13	14	1	2	11	8
3	15	0	12	5	9	10	6	4	8	11	7	14	2	1	13
9	5	15	3	12	0	6	10	7	11	8	4	2	14	13	1

Fig. 6. A 6×4 S-box with score 8, generated by our CSP solver

for the other two heuristics, $\tau = 16$. In a 5-hour run, we observed that $H_C^{64,16}$ proceeds approximately 20–200 times faster than $H_S^{64,16}$. 3DES S-box S_7 has the maximum nonlinearity score equal to 18 while the minimum of 10 is possessed by S_4. *Heuristic $H_I^{64,10}$ is observed to yield 6×4 S-Boxes having a score of 8, which is far better than the "best" published 3DES S-Boxes.* 3,600 such 6×4 S-Boxes were generated in the first hour and this number went up to more than 13,500 in the 5-hour run. Fig. 6 shows one such S-box generated by our CSP solver employing heuristic $H_I^{64,10}$.

A New Metric: Percentage of the Search Space Explored. Although our techniques have found S-Boxes with the "best" score so far, we do not know if they are optimal. To know whether we have found optimal-quality S-Boxes we have to exhaust the whole search space. If the search space is too large to be exhausted with available techniques, we would like to at least know what fraction of this search space we have managed to explore, as a measure of the probability that the optimal solution could have been found.

We therefore quantify the size of the search space, as the total number of potential $n \times m$ S-Boxes. Assuming that the solver is systematic, each node of the search tree defines a traversed distance (explored search space):

$$S_p^{(n\times m)} = \sum_{i=0}^{|X'|-1} x_i \cdot (2^m)^{|X'|-i-1} \tag{13}$$

For 6×4 S-Boxes, $S_p^{(6\times4)}$ evaluates to 78-digit base-10 numbers. Given the large size of this search space, distances typically covered by the MAC solver in reasonable time differed only in their last few assignments (78-digit numbers differed in approximately the last 15 digits). Sometimes, certain constraints rule out much larger areas of the search space. To conveniently report this, we define a *search offset* metric S-box $S_{p_1}^{(n\times m)}$:

$$r^{(n\times m)} = \frac{S_p^{(n\times m)} - S_{p_1}^{(n\times m)}}{2^{n\times 2^m}} \tag{14}$$

Here, $S_{p_1}^{(n\times m)}$ denotes the value for $S_p^{(n\times m)}$ (determined from Eq. 13) for the first S-box obtained by the solver. The solver has yielded $S_{p_1}^{(6\times4)} \approx \texttt{0x033} \times \texttt{16}^{\texttt{60}}$. The difference between $S_{p_1}^{(6\times4)}$ for the incomplete and complete heuristics is $\approx 3 \times 16^{52}$ even when they use the same value for τ (graphs not shown due to lack of space). Table 4 reports the (scaled) search offsets of the solver using complete heuristics.

Developments. One can now extend the CSP model with constraints for various special security requirements. We would like to post the obtained constraints as benchmarks for the CSP community. Once the CSP model is available, it can be easily used to generate SAT models and test SAT techniques. The obtained S-Boxes can be used to strengthen protocols such as SSL (where 3DES is now one of the main ciphers). In this direction, the first author moved to one of the main US-based providers of SSL technologies.

5 Conclusion

We conclude that CP is a powerful formalism, able to model accurately such complex criteria as the 3DES security constraints, and in particular the nonlinearity requirement. The fact that generic solvers can then address such complex problems efficiently and improve over all existing results is a testimony to the importance of this tool.

References

1. Adams, C.M., Tavares, S.E.: Good S-boxes are easy to find. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 612–615. Springer, Heidelberg (1990)
2. Adams, C., Tavares, S.: Generating and counting binary bent sequences. IEEE Transactions on Information Theory 36(5), 1170–1173 (1990)
3. Bessière, C., Régin, J.C.: Refining the basic constraint propagation algorithm. In: Nebel, B. (ed.) IJCAI, pp. 309–315. Morgan Kaufmann, San Francisco (2001)
4. Biham, E., Shamir, A.: Differential cryptanalysis of the data encryption standard. Springer, Heidelberg (1993)
5. Clark, J., Jacob, J., Maitra, S., Stanica, P.: Almost boolean functions: the design of boolean functions by spectral inversion. Evolutionary Computation 3, 2173–2180 (2003)
6. Coppersmith, D.: The data encryption standard (des) and its strength against attacks. IBM J. Res. Dev. 38(3), 243–250 (1994)
7. Daemen, J., Rijmen, V.: Aes proposal: Rijndael (September 1999), `http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf`

8. Feistel, H.: Cryptography and computer privacy. Scientific American 228, 15–23 (1973)
9. Gupta, K.C., Sarkar, P.: Construction of high degree resilient S-boxes with improved nonlinearity. Inf. Process. Lett. 95(3), 413–417 (2005)
10. Hoeve, W.V.: The alldifferent constraint: A survey. In: Proceedings of the Sixth Annual Workshop of the ERCIM Working Group on Constraints (2001)
11. Lai, X., Massey, J.L.: A proposal for a new block encryption standard. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (1991)
12. Laskari, E.C., Meletiou, G.C., Vrahatis, M.N.: Utilizing evolutionary computation methods for the design of S-boxes. In: Proceedings of the International Conference on Computational Intelligence and Security 2006 (CIS 2006), China (2006) (in press)
13. Matsui, M.: Linear cryptanalysis method for des cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
14. Mister, S., Adams, C.: Practical S-Box design (1996)
15. NIST: Data encryption standard (DES). Federal Information Processing Standard (FIPS) 46-2 (January 1988)
16. O'Connor, L.: An analysis of a class of algorithms for s-box construction. J. Cryptology 7(3), 133–151 (1994)
17. O'Connor, L.: Properties of linear approximation tables. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, Springer, Heidelberg (1995)
18. Pieprzyk, J., Finkelstein, G.: Towards effective nonlinear cryptosystem design. IEE Proceedings Computers and Digital Techniques 135(6), 325–335 (1988)
19. Puget, J.-F.: A fast algorithm for the bound consistency of alldiff constraints. In: AAAI 1998, pp. 359–366 (1998)
20. Sabin, D., Freuder, E.C.: Contradicting conventional wisdom in constraint satisfaction. In: PPCP 1994. LNCS, vol. 874, pp. 10–20. Springer, Heidelberg (1994)
21. Schneier, B.: Applied cryptography — protocols, algorithms, and source code in c. In: Textbook, ch. 12, pp. 265–301 (2002)
22. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
23. Stallings, W.: Cryptography and network security - principles and practices. In: Textbook, ch. 3, pp. 86–90 (2003), `http://www.prenhall.com/stallings`