Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R)

Eli Biham

Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
Email: biham@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/

Abstract. In a series of papers Patarin proposes new efficient public key systems. A very interesting proposal, called *2-Round Public Key System with S Boxes*, or *2R*, is based on the difficulty of decomposing the structure of several rounds of unknown linear transformations and S boxes. This difficulty is due to the difficulty of decomposing compositions of multivariate binary functions. In this paper we present a novel attack which breaks the 64-bit block variant with complexity about 2^{30} steps, and the more secure 128-bit blocks variant with complexity about 2^{60} steps. It is interesting to note that this cryptanalysis uses only the ciphertexts of selected plaintexts, and does not analyze the details of the supplied encryption code.

1 Introduction

The search for efficient public key cryptosystems is as old as the idea of public key cryptosystems itself [1]. Many of the most efficient proposed schemes were based on multivariate polynomials [3, 9, 2], but they were usually broken later [8, 10, 4]. In a series of papers Patarin proposes new secure and efficient public key systems [5, 6] based on hiding the structure of polynomials in a difficult-to-analyze encryption code, and analyzes other similar schemes [4]. One of his more promising scheme is the very efficient *2-Round Public Key System with S Boxes* (shortly called *2R*) [7]. The design of this scheme is unique as it uses techniques from symmetric ciphers in designing a public key cryptosystem, while still claiming security based on relation to the difficulty of decomposing compositions of multivariate binary functions.

Patarin's public key cryptosystem with S boxes encrypts by performing the following secret operations on 64-bit or 128-bit plaintexts:

1. Invertible linear transformation L_0
2. First layer of 8 (or 16) 8x8-bit quadratic S boxes $S_{1,0}, \ldots S_{1,7}$, collectively denoted by S_1
3. Another linear transformation L_1
4. Second layer of S boxes $S_{2,0}, \ldots S_{2,7}$, collectively denoted by S_2

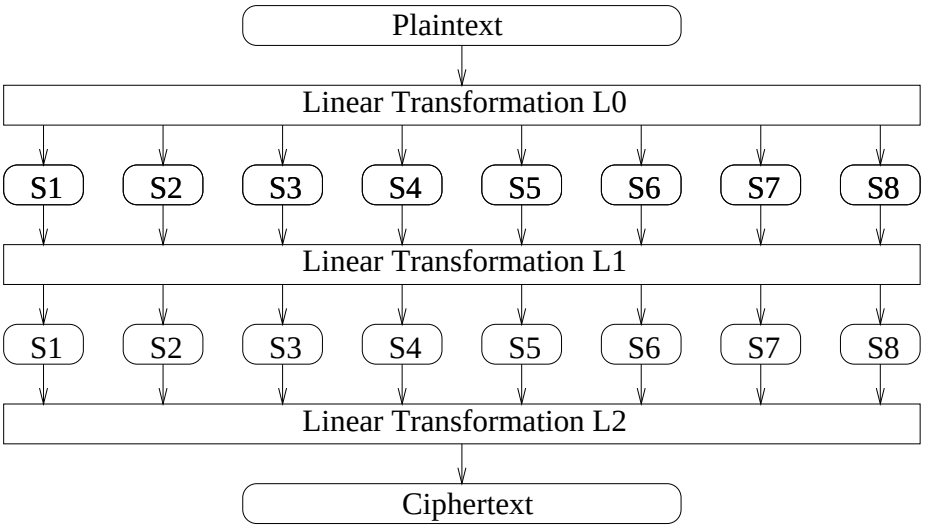


Fig. 1. Outline of Patarin's 2R scheme

5. Final linear transformation L_2
6. The ciphertext is $C = E(P) = L_2(S_2(L_1(S_1(L_0(P)))))$

Figure 1 outlines this scheme.

Only the owner of the system knows these transformations, and uses this knowledge for decryption. The publication of the system hides the structure by giving an equivalent description from which it is claimed to be very difficult to identify the original description, due to the difficulty of decomposing compositions of binary functions.

In the rest of this paper we assume that the encryption function is given as an oracle (black box). Our analysis does not study the supplied code of the system, and does not try to decompose the binary function. On the contrary, it observes the full details of the function given only the ciphertexts of (many) selected plaintexts, which are encrypted using the supplied encryption function. Moreover, it does not rely on the quadraticness of the S boxes.

A major observation is that the S boxes are not bijective. The designer claims that if the S boxes were bijective, the security of the system might had been compromised. Therefore, decryption is not unique, and some redundancy should be added to the plaintext to allow unique decryption[1].

In this paper we present the first cryptanalysis of this scheme, for which we received the prize promised by the designer for the first person to break this scheme. Later, Ye, Lam and Dai devised a different attack based on the algebraic structure of the scheme [11]. Their attack is less efficient than ours, although it

[1] Although an attacker may decide not to add this redundancy when he generates the ciphertexts used for the attack.

imposes more restrictions than we do (e.g., their attack would not work if the S boxes were not quadratic, while our attack is not very sensitive to the method used to construct the S boxes).

Our attack can break the 2R scheme with complexity about 2^{30} when the block size is 64 bits. Moreover, it can also break the more secure variant with 128-bit blocks with complexity about 2^{60}.

This paper is organized as follows: In Section 2 we describe our main observations and tools used later in the analysis, and in Section 3 we describe the attack. Section 4 discusses possible improvements of the scheme. The paper is summarized in Section 5.

2 The Main Observations and Tools

Our main observation is that the S boxes are not bijective, and thus outputs of different inputs to the same S box may collide. Given a pair of plaintexts, such S boxes which have different inputs in both encryptions but have the same outputs will be called *active* S boxes. Therefore, there exist many pairs P, P^* of plaintexts for which the outputs of all the S boxes in the first round collide, which cause the ciphertexts to collide as well. Such collisions can be constructed as follows: For any S box S_i there exist pairs of 8-bit values Y and Y^* such that $S_{1,i}(Y) = S_{1,i}(Y^*)$. Let X be a 64-bit value whose bits $8i, \ldots, 8i+7$ equal Y, and let X^* be equal to X, except for these bits, whose value is replaced by Y^*. Let $P = L_0^{-1}(X)$ and $P^* = L_0^{-1}(X^*)$. Then, the ciphertexts $E(P)$ and $E(P^*)$ are equal. Figure 2 outlines the differences in pairs of encryptions.

The attacker does not know the details of the linear transformation and the S boxes, and cannot construct collisions using this algorithm. However, if the attacker finds a collisions, he might be interested to know if the collision already occurs after the first level of S boxes, and if there is only one active S box. He can also be interested to know if there is a common active S box in two different colliding pairs. In the next subsections we present two algorithms which will be useful for this purpose.

2.1 Algorithm A

We observe that given a pair P, P^* such that $E(P) = E(P^*)$, we can identify whether there is only one active S box in the first round (and none in the second round), or there are two or more active S boxes. This identification can be performed by the following algorithm:

1. Given P and P^*
2. Repeat about 1000 times:
   (a) Select Q at random
   (b) Compute $Q^* = Q \oplus P \oplus P^*$
   (c) Check whether $E(Q) = E(Q^*)$
3. Let q be the fraction of times with equality $E(Q) = E(Q^*)$ in the previous step

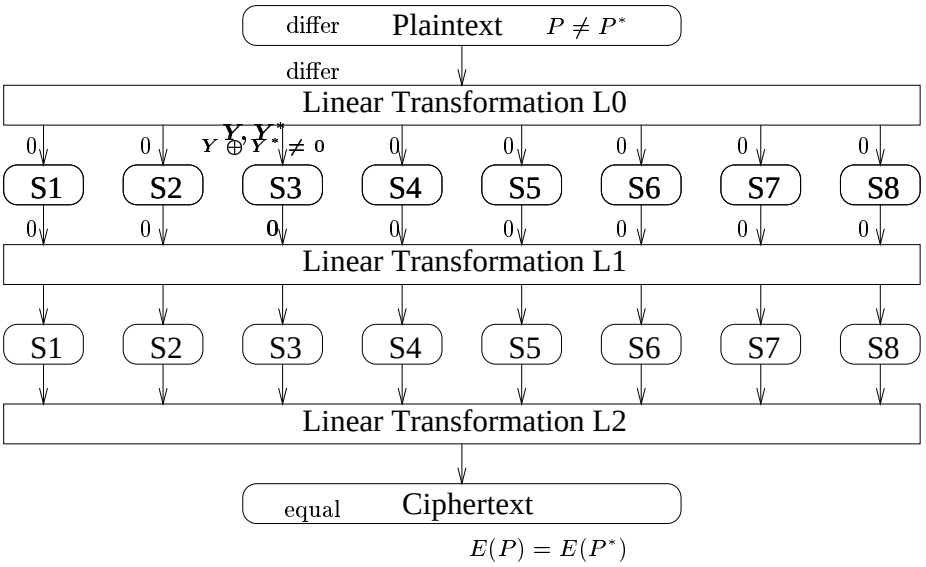


Fig. 2. Outline of the main observation

4. If $q > 1/256$, output *one-active-S-box*, and conclude that there is only one active S box in the first round.
5. Otherwise output *several-active-S-boxes*.

This algorithm works since the differences in the inputs of the S boxes in the first round are the same for P and P^* and for Q and Q^* due to linearity. Thus,

1. If there is a difference only in one S box for P and P^*, so is for Q and Q^*. Since Q is chosen randomly, and since there are 128 pairs of inputs to an S box with the particular difference, there is a probability of $1/128$ that the pair of inputs of the active S box are the same in both pairs, and thus a fraction of at least $1/128$ of the pairs of outputs collide in Q and Q^*. The repetition 1000 times ensures that the probability of a wrong output is very small.
2. If there are differences in the inputs of two or more S boxes in the first round, the expected q reduces to about 2^{-7m} or a small factor of it, where m is the number of S boxes with differing inputs.
3. If the outputs of the first layer of S boxes of P and P^* do not collide, but the ciphertexts do collide, the expected q is negligible (typically about 2^{-60} or less for 64-bit blocks).

2.2 Algorithm B

We can also identify if two pairs P_1, P_1^* and P_2, P_2^* which satisfy $E(P_1) = E(P_1^*)$ and $E(P_2) = E(P_2^*)$ (and collide already after the first round) have distinct active S boxes in the first round, or whether there is some common active S box:

1. Given P_1, P_1^*, P_2, P_2^*
2. Compute $Q_1 = P_1 \oplus P_2 \oplus P_2^*$ and $Q_1^* = P_1^* \oplus P_2 \oplus P_2^*$
3. Compute $Q_2 = P_2 \oplus P_1 \oplus P_1^*$ and $Q_2^* = P_2^* \oplus P_1 \oplus P_1^*$
4. If $E(Q_1) \neq E(Q_1^*)$ or $E(Q_2) \neq E(Q_2^*)$ output *common*, and conclude that there exists a common active S box.
5. If $E(Q_1) = E(Q_1^*)$ and $E(Q_2) = E(Q_2^*)$ output *distinct* and conclude that the pairs have distinct active S boxes with a high probability.

This algorithm should detect almost all cases of common active S boxes. If it does not detect as such, there is a high probability that there are no common active S boxes.

This algorithm works, as the pairs Q_1, Q_1^* (and Q_2, Q_2^*) differ exactly in the same S boxes as of P_1, P_1^* (P_2, P_2^*, respectively). If there are no common active S boxes, the active S boxes in Q_1, Q_1^* (Q_2, Q_2^*) have exactly the same inputs as in P_1, P_1^* (P_2, P_2^*), and thus the same collision occurs. If there is some common active S box, the inputs in Q_1, Q_1^* (Q_2, Q_2^*) are different than in P_1, P_1^* (P_2, P_2^*), and thus the probability of collision is small.

3 The Attack

3.1 Analyzing the First Linear Transformation

The first step of the attack computes the ciphertexts of many random plaintexts, and collects pairs P, P^* whose ciphertexts $E(P)$, $E(P^*)$ are equal, and for which Algorithm A outputs one-active-S-box.

Given such pairs we use algorithm B to divide them to eight sets, sorted by the active S box. We can use this result to find a basis for the differences $P \oplus P^*$ of each set. The combination of all these bases form a basis for the plaintext space, which relates to the inputs of S boxes of the first round. In total we get eight sets, each consists of eight basis vectors. Each set of basis vectors affect a different S box in the first round. We cannot identify the order of the S boxes nor any transformation on the inputs of individual S boxes. Therefore, this basis is all the information we can get on the linear transformation L_0. Without loss of generality, all the rest of the definition of L_0 can be viewed as part of the S boxes and their order.

This step of the attack is the most complex: 2^{37} random plaintexts are encrypted by the attacker in order to find about 1000 random collisions. However, a careful analysis shows that due to the structure of the cipher, the number of collisions will be about 256 times higher, and about 4000 of these collisions will have only one active S box. Then, the application of Algorithm A to identify the pairs with one active S box requires about $256 \cdot 1000 \cdot 2000 = 2^{29}$ encryptions.

The application of Algorithm B to recover the basis takes a few thousands additional encryptions. When applied efficiently, considering all the information we get during the analysis (such as a partial basis) and improvements to the attack (such as analysis of pairs with two or three active S boxes in the first round), the total complexity of this step can be reduced to below 2^{30}.

3.2 Analyzing the Last Linear Transformation

In the next step of the attack we find a basis for the ciphertexts, related to the outputs of the eight S boxes in the second round. We observe that inputs to the first layer of S boxes which differ only in the inputs to one S box cause a difference in the output of this S box only. In turn, the output of L_1 has difference in the inputs of most S boxes in the second round. In some cases however such differences do not affect about one or two S boxes in the second round. Although to overcome this weakness, designers might design linear transformations as multipermutations from the outputs of the S boxes in one round to the inputs of the S boxes in the next round, a similar property may occur when differences in pairs of S boxes in the first round lead to zero differences in the inputs of one (or a few) S boxes in the second round.

Using the basis we got for the plaintexts, we can now control the inputs of selected S boxes in the first round. In particular, we can generate structures of many plaintexts with exactly the same value in the inputs to one (or two) S boxes in the first round, but with random values in the inputs to the rest of the S boxes. We can even generate pairs of S boxes in which 1) in all pairs one member has some fixed (but unknown) input F_1 to one or two S boxes and the other member has some other fixed (but unknown) value F_2, 2) these fixed values F_1, F_2 are fixed in all the pairs, 3) in each pair random values are selected for the inputs of the rest of the S boxes, and both members of the pair have the same values in all the rest of the S boxes. As an example for such pairs, the inputs of the S boxes in a pair can be $(F_1, R_1), (F_2, R_1)$, where in another pair it is $(F_1, R_2), (F_2, R_2)$, and in a third pair it is $(F_1, R_3), (F_2, R_3)$, etc. Note that the input differences to the second layer of S boxes are fixed in all the pairs of a structure, and depend only on $F_1 \oplus F_2$.

In this step we generate many such structures, compute the ciphertext differences, and compute a basis for the space spanned by the ciphertext differences. If there are differences in the inputs of all the S boxes in the second round, it is expected that the space spanned by the ciphertext differences is of dimension 64. If there is one S box with zero difference the spanned space is expected to be with dimension 56 (and cannot be larger than 56). In general, it is expected that non-zero differences in the inputs to m S boxes in the second round lead to dimension $8m$ of the spanned space. It is also expected that in such case structures of about 100 pairs (200 encrypted plaintexts) suffice to span all the space, and additional pairs rarely enlarge the spanned space. It is also expected that about one of every $256/8 = 32$ structures lead to space of dimension less than 64. In order to divide the space to the partial spaces inherited by the outputs of the S boxes, we need about 7–10 such structures, and thus we need in total

about $10 \cdot 32 \cdot 200 = 64000$ encrypted plaintexts. At the end of this step we know all the possible information on L_2, except for the information that can be viewed as part of the S boxes.

3.3 Analyzing the Intermediate Layers

Up to this point we showed how to remove the first and last linear transformations. Using this information we can now find how many (effective) output bits of each S box in the first round affect each one of the inputs of the S boxes in the second round. This step is performed by encrypting 256 plaintexts for each S box in the first round, which contain all the possible inputs to the S box, and in which the inputs to all the other S boxes are fixed. We count the number of different outputs of each S box in the second round. If the count is 2^m (or slightly smaller) for some m, there is a high probability that the rank of the linear transformation from the output of the S box in the first round to the input of (only) the S box in the second round is m.

From the same data we can even find information on the values of the S boxes, as by looking at the outputs of the S boxes in the second round we can group inputs to the S box in the first round by the values of the m bits of its output that affect the S box in the second round. By correlating this information among several S boxes in the second round we can complete most information on the S boxes in the first round, including their values, and the actual bits that are transfered by L_1 to each S box. The only values that we cannot identify are the affine constants, which can be viewed as part of the second S box layer instead.[2] It is now easy to complete the second layer of S boxes, as all the rest of the cipher is already recovered.

4 Discussion

A possible improvement is using bijective S boxes in the first layer of S boxes. This modification ensures that the first step of the attack is not applicable, although other attacks might become practical. This modification can be combined with an increased number of rounds using non-bijective S boxes to protect against other kinds of attacks that require at least two rounds of non-bijective S boxes.

If the first layer of S boxes remain non-bijective, the first step of the attack can still find the first linear transformation regardless of the number of rounds and the design of the other rounds. Therefore, such a variant of this scheme may starts with the layer of S boxes without a preceding linear transformation without affecting its security. In such a case we would propose using at least three layers of S boxes, each followed by a linear transformation to ensure that

[2] They can actually be eliminated, as the scheme is equivalent to a scheme in which the zero inputs to S boxes always have zero outputs, and the plaintext zero (or any other selected plaintext) is encrypted using only zero inputs of S boxes.

the rest of the attack is not applicable. However, adding layers come with an unacceptable penalty in the size of the public key and encryption speed.

A promising improvement seems to discard the redundant input variables from the published equations, replacing them by their equivalent formulae in terms of other variables. In such a way Algorithm A might never output one-active-S-box in the first step of the attack. If in addition some of the 64 equations are also discarded, it seems that all the known attacks will not work.

Remark: The authors of [11] propose that the S boxes should not be kept secret. However, if the S boxes are not secret, it would simplify the recovery of the linear transformations in our attack, giving more information to the attacker. We highly recommend to keep the S boxes secret, just as they should be in the original 2R scheme.

5 Summary

In this paper we proposed a practical attack, which is the first attack against Patarin's 2-Round Public Key System with S Boxes. For a blocksize of 64 bits, the complexity of the attack is about 2^{30} encryptions (that the attacker can compute on his machine as this is a public key scheme). The more secure variant with 128-bit blocks can be analyzed with complexity about 2^{60}. Efficient implementations of the attack might even have marginally smaller complexities than these.

Acknowledgments

We are grateful to Jacques Patarin for his great help during our work on this attack, and for the invitation for an excellent restaurant in Paris (the promised prize for breaking this system).

References

1. W. Diffie, M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 22, No. 6, Nov. 1976.
2. Tsutomu Matsumoto, Hideki Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'88, pp. 419-453, 1988.
3. H. Ong, C. P. Schnorr, Adi Shamir, *Efficient Signature Schemes Based on Polynomial Equations*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'84, pp. 37–46, 1984.
4. Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'95, pp. 248–261, 1995.

5. Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'96, pp. 33–48, 1996.
6. Jacques Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'96, pp. 45–60, 1996.
7. Jacques Patarin, Louis Goubin, *Asymmetric Cryptography with S Boxes*, proceedings of ICICS'97, Lecture Notes in Computer Science 1334, pp. 369–380, 1997.
8. John M. Pollard, Claus P. Schnorr, *An Efficient Solution to the Congruence* $x^2 + ky^2 = m (\mathrm{mod}\, n)$, IEEE Transactions on Information Theory, Vol. 33, No. 5, 1987.
9. Adi Shamir, *Efficient Signature Schemes Based on Birational Permutations*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 1–12, 1993.
10. Serge Vaudenay, Jacques Stern, Don Coppersmith, *Attacks on the Birational Permutation Signature Schemes*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 435–443, 1993.
11. Ding-Feng Ye, Kwok-Yan Lam, Zong-Duo Dai, *Cryptanalysis of "2R" Schemes*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'99, pp. 315–325, 1999.