Fast Primitives for Internal Data Scrambling in Tamper Resistant Hardware

Eric Brier[1], Helena Handschuh[2], and Christophe Tymen[3]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
eric.brier@gemplus.com
[2] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
helena.handschuh@gemplus.com
[3] École Normale Supérieure
45 rue d'Ulm, 75230 Paris, France
christophe.tymen@gemplus.com

Abstract. Although tamper-resistant devices are specifically designed to thwart invasive attacks, they remain vulnerable to micro-probing. Among several possibilities to provide data obfuscations, keyed hardware permutations can provide compact design and easy diversification. We discuss the efficiency of such primitives, and we give several examples of implementations, along with proofs of effectively large key-space.

Keywords. Tamper-resistance, Probing attacks, Data scrambling, Keyed permutations, Smart-cards.

1 Introduction

Microprobing techniques are invasive attacks consisting in introducing a conductor point into certain parts of a tamper-resistant chip to monitor the electrical signal at this spot[3,1], in order to extract some secret information. A natural means to thwart these attacks consists in encrypting the data stored or exchanged inside the chip. Using classical block-ciphers like DES provides a natural solution, but this method becomes quickly illusory when the concerned data transit through highly time critical processes, like for example the communication between the microprocessor and the RAM. In this case, more hasty techniques must be used to provide very fast processing at the expense of a lower, but acceptable security level. This category of techniques is usually and informally called scrambling, or obfuscation, as opposed to encryption[4].

A popular primitive for scrambling in highly constrained environments consists simply in bit permutations, these permutations being parameterized by a key. As it appears in what follows, such functions result in very compact designs, where only one cycle is needed to process the data. Furthermore, a large number of permutations can be generated, with a one-to-one correspondence with the key space. Ultimately, keyed permutations can be easily used in more complex functions which require some keyed linear components.

Ç.K. Koç, D. Naccache, and C. Paar (Eds.): CHES 2001, LNCS 2162, pp. 16–27, 2001.

More precisely, this paper addresses the problem of designing keyed permutations of compact shape, that generate a large set of permutations when the key runs over the key space, and that offer good properties against chosen plaintext attacks in the context of physical probing. This combinatorial issue is tractable for a small number of bits, but becomes more intricate for realistic values like 16 or 32, which brings intrinsic interest to the results of Section 3. The rest of this paper is organized as follows. Section 2 defines a security model for scrambling functions, and proposes a criterion for the design of keyed permutations. Section 3 is the main part of our paper. Three different constructions for keyed permutations are proposed, along with proofs of some of their properties. Hardware engineers interested in quickly evaluating the practical contribution of this paper can directly jump to Section 4, which contains some numerical data about our new keyed permutations. Some possible applications are also listed. An example of a very fast on-chip data scrambler which integrates keyed permutations is proposed.

2 Scrambling Functions and Probing Attacks

2.1 Security Model

We consider the context of a smart-card microprocessor, which communicates with the RAM. The memory, and the channel which links it to the microprocessor, are subject to probing attempts. Consequently, to prevent information disclosure, a data word $(b_0, \ldots, b_{n-1})$ is encrypted with a key K using a scrambling function C_K before being sent to the memory. The key K may be refreshed each time the card is reset, but might also be regenerated more often, using multiple keys encryption techniques. We assume that the attacker is allowed to play with the microprocessor, which implies that he can send any data he wants to the memory. His goal is to decipher a secret data present in the card, read from the RAM at some time. The difference with a classical chosen plaintext attack on a block-cipher is that the attacker has only a partial knowledge of the ciphertext. Indeed, probing attacks are usually not easy to mount, and in particular, the attacker might rarely probe wherever he wants[3]. Consequently, we restrict the capabilities of the attacker to recovering only *some* of the bits $(b'_0, \ldots, b'_{n-1}) = C_K(b_0, \ldots, b_{n-1})$.

2.2 A Security Criterion for Linear Functions

For efficiency reasons, it is practical to choose for C_K a linear function. This choice does not provide any security against full chosen plaintext attacks, but might be sufficient if we assume that the attacker knows very few bits b'_i. One of the possible strategies of the attacker to decrypt a secret data might be to recover completely K during a preliminary phase when several plaintext messages are sent to the scrambling function. In this context, we can quantify the security provided by C_K by determining the number of wires that the attacker has to

be able to probe simultaneously to recover the key. In particular, when C_K is a permutation σ_K of the group S_n of the permutations of $\{0,\ldots,n-1\}$, this question boils down to: what is the minimal number of pairs $(i,\sigma_K(i))$ the attacker needs to know to recover K entirely ? To formalize this condition, we introduce some definitions and notations. If μ and σ are two elements of S_n, we denote by $\mu\sigma$ the permutation defined by $i \mapsto \mu(\sigma(i))$. We also denote by ι the permutation such that $\iota(i) = i$ for all $i \in \{0,\ldots,n-1\}$.

An $(n,k)-$*keyed permutation* is a map from the set $\{0,1\}^k$ to S_n :

$$\begin{array}{rcl} \sigma : \{0,1\}^k & \longrightarrow & S_n \\ K & \longmapsto & \sigma_K \quad . \end{array}$$

The *degree of freedom* of an $(n,k)-$keyed permutation is the smallest integer $m \geq 1$ such that there exists an $(m+1)-$tuple $(i_1,\ldots,i_{m+1})$ of pairwise distinct elements of $\{0,\ldots,n-1\}$, such that the map

$$\begin{array}{ll} \left\{\sigma_K / K \in \{0,1\}^k\right\} & \longrightarrow \{0,\ldots,n-1\}^{m+1} \\ \sigma_K & \longmapsto (\sigma_K(i_1),\ldots,\sigma_K(i_{m+1})) \end{array}$$

is injective. Informally, the degree of freedom is equal to the minimum number of pairs $(i,\sigma_K(i))$ we have to fix to determine uniquely σ_K. Note that this does not mean that this suffices to determine K, as the map from $\{0,1\}^k$ to S_n might not be injective, but in our context, the secret key is completely recovered as soon as σ_K is known. From a practical standpoint, this definition implies also that we should look for keyed permutations with a degree of freedom as high as possible.

For example, in the strongest case, if σ is surjective in S_n, then σ has degree of freedom $n-1$: we need exactly $n-1$ distinct pairs $(i,\sigma_K(i))$ to determine completely σ_K (the missing value is infered from the $n-1$ others, since σ_K is a bijection). For the weakest case, let $\mu \neq \iota$ be in S_n, and consider the keyed permutation σ_b from $\{0,1\}$ to S_n such that $\sigma_0 = \iota$ and $\sigma_1 = \mu$. Then σ as degree of freedom one: as $\mu \neq \iota$, there exist i_1 such that $\sigma_b(i_1) \neq i_1$ iff $b = 1$.

3 A Recursive Construction

3.1 Outline of the Result

This section explains the construction of three different $(n,k)-$keyed permutations when n is a power of two. These three constructions can be realized using combinatorial logic, and the corresponding circuits are of depth $\log_2 n$. Consequently, they achieve a very compact shape, and very short propagation delay features.

The construction of Section 3.3 generates 2^{n-1} permutations, which are in one-to-one correspondance with the key space. This construction is improved in Section 3.4, where we generate $n^{n/2}$ permutations. Section 3.5 still improves this result by generating at least $n^{\alpha n}2^{-\beta n}$ permutations, with $\alpha = (\log_2 6)/4 \approx 0.65$ and $\beta = (\log_2 6)/4 - 1/2 \approx 0.15$. Furthermore, we prove that the last two constructions have degree of freedom at least $n/2 - 1$.

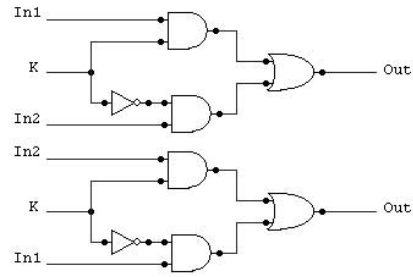

Fig. 1. Hardware realization of a switch

3.2 Hardware Representation of Keyed Permutations

The most natural approach to design hardware permutations is to use the set of the transpositions of S_n. We recall that a transposition is an element (i, j) of S_n which exchanges the symbol i with the symbol j. A well-known fact is that every permutation on can be expressed as a product of transpositions. If t is a transposition, a keyed permutation $b \mapsto t^b$ with one bit of key can be realized using two parallel multiplexers. We call such a block a *switch*. A hardware realization of a switch is given in figure 1. Oriented graphs provide a compact representation of switch based circuits. For example, figure 2 represents the keyed permutation $(b_0, b_1) \mapsto (1, 3)^{b_0}(0, 1)^{b_1} \in S_4$. The grey nodes correspond to the switches, and are commanded by additional key wires, which do not appear on the figure. In the following, the *depth* of a circuit will refer to the number of stages composing the circuit, this number being related to a switch-based design. Note that the switch-depth is less than or equal to the multiplexer-depth.

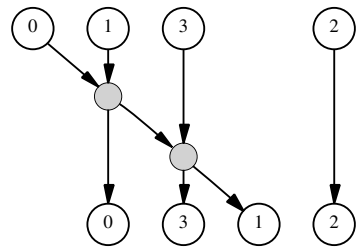

Fig. 2. Graphical representation of the keyed permutation $(1, 3)^{b_0}(0, 1)^{b_1}$

3.3 A Group Theoretic Construction

We denote by $\mathcal{H}_2^n$ a greatest subgroup of S_n which order is a power of two. $\mathcal{H}_2^n$ is called a *Sylow* $2-$*subgroup* of S_n. In all the following, we will suppose that n is a power of two. In this case, $\mathcal{H}_2^n$ has order 2^{n-1}[5]. A set of generators of $\mathcal{H}_2^n$ can be constructed recursively as follows. Consider the set $\{0, \ldots, n-1\}$, and

the permutation $g \in \mathcal{H}_2^n$ which exchanges $\{0, \ldots, n/2-1\}$ with $\{n/2, \ldots, n-1\}$ by $i \leftrightarrow i + n/2$. Now, we can repeat inductively this procedure by considering the sets $\{0, \ldots, n/2-1\}$ and $\{n/2, \ldots, n-1\}$. We get finally $n-1$ elements of S_n which generate $\mathcal{H}_2^n$.

These generators are very easy to implement in hardware, since permuting two sets of k bits can be done using k switches parameterized by the same bit of key. This yields to a $(n, \log_2 n - 1)-$keyed permutation, that can be realized using $(n \log_2 n)/2$ switches. Figure 3 summarizes schematically this recursive construction.

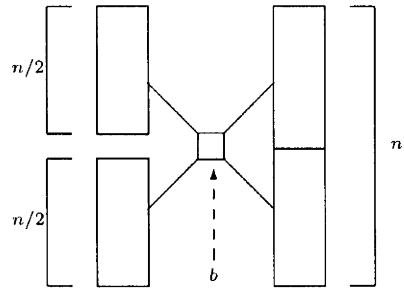

Fig. 3. Recursive contruction based on Sylow 2−subgroups

An interesting property of this design is that the set of generated permutations forms a group. We can take advantage of this fact to increase the number of generated permutations: In our previous construction, we built a hardware design which realized the keyed permutation g_K, where g_K takes all the values of $\mathcal{H}_2^n$ when K runs over the key space. Denote by ρ a well-chosen permutation, which we implement in hardware, that is, by permuting physically the wires. Then, by reusing the previous construction, we can realize the keyed permutation

$$s_{(K_1,K_2)} = g_{K_1} \circ \rho \circ g_{K_2} \ ,$$

which should generate more permutations. The question is to determine how many permutations are effectively generated by this method. It is easy to see that no collisions appear (i.e. the number of generated permutations is equal to $|\mathcal{H}_2^n|^2$) iff the following algebraic condition is verified:

$$\rho \mathcal{H}_2^n \rho^{-1} \cap \mathcal{H}_2^n = \{\iota\} \ . \tag{1}$$

The naive complexity of checking if a given permutation ρ verifies (1) is equal to $|\mathcal{H}_2^n| = 2^{n-1}$. Consequently, our approach fails as soon as say $n \geq 32$, since this last verification has to be made $32!/2$ times on average before finding a solution. Nevertheless, for $n = 32$, we may still get a result using the following trick: we define H, the subgroup of $\mathcal{H}_2^{32}$ which preserves $\{0, \ldots, 15\}$ and $\{16, \ldots, 31\}$. H is isomorphic to $\mathcal{H}_2^{16} \times \mathcal{H}_2^{16}$, and has cardinality 2^{30}. Consider the keyed

permutation $(K_1, K_2) \mapsto h_{K_1} \beta g_{K_2}$, where h_{K_1} runs over H, g_{K_2} runs over $\mathcal{H}_2^{32}$, and where β is a fixed permutation. This map is injective iff

$$\beta H \beta^{-1} \cap \mathcal{H}_2^{32} = \{\iota\} \ . \tag{2}$$

A simple method to find such a β is first to solve (1) for $n = 16$, and then to set $\beta = (\rho(\cdot), \rho(\cdot - 16) + 16)$. This search terminates on average after $(|S_{16}/\mathcal{H}_2^{16}| \cdot |\mathcal{H}_2^{16}|)^{1/2} \approx 2^{22}$ trials. For instance, the following permutation is a solution of (1) for $n = 16$:

$$\rho = (0, 15, 9, 10, 11, 12, 13, 14)(1, 2, 3)(4, 5, 6, 7, 8) \ .$$

The resulting number of generated permutations is equal to $2^{30} \cdot 2^{31} = 2^{61}$.

3.4 Generalization of the Group-Based Design

Unfortunately, this improvement works only for $n \leq 32$. Furthermore, we generate only 2^{61} permutations, among the $32! \approx 2^{118}$ elements of S_{32}. Nevertheless, as we will see, a slight modification of the set of generators leads to generating a much larger subset of S_n. The price to pay for this improvement is to lose the group property, but this has no impact for our application. As before, the solution is built recursively by induction on $\log_2 n$, so that at each step of the induction, we add a new stage to the corresponding circuit.

Theorem 1. *If n is a power of two, then there exists a circuit of depth $\log_2 n$ involving $(n \log_2 n)/2$ switches, which realizes a $(n, (n \log_2 n)/2)-$keyed permutation δ. Furthermore, the number of distinct generated permutations is equal to the number of keys, that is $n^{n/2}$.*

Proof. We proceed by induction. Let σ be a $(n/2, ((n/2)\log_2(n/2))/2)-$keyed permutation with the properties stated in the theorem. For convenience, we set $k = ((n/2)\log_2(n/2))/2$. First, we defined the $(n, 2k)-$keyed permutation μ as

$$\mu_{(K_1,K_2)}(i) = \begin{cases} \sigma_{K_1}(i) & \text{if } \ 0 \leq i < n/2 \\ \sigma_{K_2}(i - n/2) + n/2 & \text{if } \ n/2 \leq i < n \ , \end{cases} \tag{3}$$

where $K_1, K_2 \in \{0,1\}^k$. Let set $k' = 2k + n/2$, and define the $(n, k')-$keyed permutation δ by

$$\delta_{(K_1,K_2,E)} = \nu_E \circ \mu_{(K_1,K_2)} \ ,$$

where

$$\nu_E = \prod_{j=0}^{n/2-1} (j, j + n/2)^{e_j} \ ,$$

and $E = (e_0, \ldots, e_{n/2-1}) \in \{0,1\}^{n/2}$. First, $k' = 2k + n/2 = n(\log_2 n - 1)/2 + n/2 = (n \log_2 n)/2$, which corresponds to what we expect. Furthermore, the number of switches used for realizing δ is equal to $2k + n/2 = k'$. It remains to prove that δ is injective. This comes from the fact that for all $0 \leq i < n/2$,

$\delta^{-1}_{(K_1,K_2,E)}(i) < n/2$ iff $e_i = 0$. Consequently, we can uniquely recover E from $\delta_{(K_1,K_2,E)}$. As μ is injective, we can also recover uniquely K_1 and K_2 from $\mu_{(K_1,K_2)} = \nu_E^{-1} \circ \delta_{(K_1,K_2,E)}$, which concludes the proof. □

Figure 4 represents an example of this construction for the case $n = 8$.

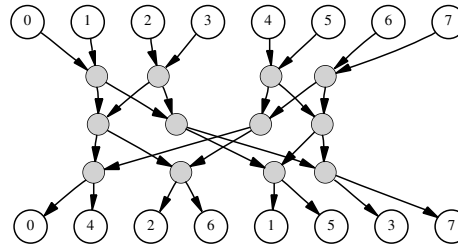

Fig. 4. Representation of δ for $n = 8$

As motivated in section 2.2, we want to check that δ has high enough degree of freedom. This is guaranteed by the following result :

Theorem 2. *The degree of freedom d_δ of the keyed permutation δ of theorem 1 verifies*

$$d_\delta \geq n/2 - 1 \ .$$

Proof. We proceed by induction on n. When $n = 2$, the theorem is true, as in order to guess the state of the switch, we have to know at least one pair $(i, \delta_K(j))$. Suppose that the theorem is true at step $n/2$. Recall that with the notations of theorem 1, δ is given by the recursion formula

$$\delta_{K=(K_1,K_2,E)} = \nu_E \circ \mu_{(K_1,K_2)} \ ,$$

where μ is defined from the $(n/2, k)-$keyed permutation σ following (3). The induction formula implies that σ has degree of freedom $n/4 - 1$. We set $r = n/2 - 1$, and we choose an $r-$tuple $(i_1, \ldots, i_r)$ of pairwise distinct elements of $\{0, \ldots, n-1\}$, a key $K = (K_1, K_2, E)$, and we set $j_l = \delta_K(i_l)$. Consider the set $I_1 = \left\{l / \nu_E^{-1}(j_l) < n/2\right\}$, and $I_2 = \left\{l / \nu_E^{-1}(j_l) \geq n/2\right\}$. As $|I_1|+|I_2| = n/2-1$, one of the two sets (for example I_1) has strictly less than $n/4$ elements: $|I_1| \leq n/4-1$. Furthermore, as μ preserves $\{0, \ldots, n/2-1\}$ and $\{n/2, \ldots, n-1\}$, for all $l \in I_1$, $i_l < n/2$. Consequently, there exists $K_1' \neq K_1$ such that

$$\forall l \in I_1, \ \ \sigma_{K_1}(i_l) = \sigma_{K_1'}(i_l) \ .$$

This implies that

$$\delta_{(K_1,K_2,E)}(i_1, \ldots, i_r) = \delta_{(K_1',K_2,E)}(i_1, \ldots, i_r) \ .$$

This proves that δ has degree of freedom greater than $n/2 - 1$. □

3.5 Further Improvements

We may still try to improve the previous construction by modifying it so that we could generate a larger set of permutations. We always consider the keyed permutation δ as defined above, with the recursion formula

$$\delta_{K=(K_1,K_2,E)} = \nu_E \circ \mu_{(K_1,K_2)} \ .$$

Define the vector $\epsilon = (\epsilon_0, \ldots, \epsilon_{n-1})$ as $\epsilon_i = \mathbf{1}\{\delta_K^{-1}(i) < n/2\}$, where $\mathbf{1}\{P\}$ is equal to one when the predicate P is true, and to zero otherwise. As underlined in the proof of theorem 1, there is a one-to-one correspondence between the set of all the keys E and the set of all the $n/2-$tuples $(\epsilon_0, \ldots, \epsilon_{n/2-1})$. This is because μ preserves the segments $i < n/2$ and $i \geq n/2$. This means that ϵ contains twice too much information. Consequently, our idea is to group the transpositions $(j, j + n/2)$ of μ two by two, and to compose them with a cycle of the four concerned elements, so that we could still invert our map thanks to the associated 4-tuple of bits ϵ_i.

Consider first the set $\{0, 1, 2, 3\}$, and the map

$$g : (b_0, b_1, b_2) \in \{0,1\}^3 \longmapsto (0,2,1,3)^{b_0}(0,2)^{b_1}(1,3)^{b_2} \ .$$

We consider also the map defined by

$$h(b_0, b_1, b_2) = \left(\mathbf{1}\{g(b_0, b_1, b_2)^{-1}(i) < 2\}\right)_{0 \leq i \leq 3} \ .$$

The truth table of h is given below:

(b_0, b_1, b_2)	$h(b_0, b_1, b_2)$
(0, 0, 0)	(1,1,0,0)
(0, 0, 1)	(1,0,0,1)
(0, 1, 0)	(0,1,1,0)
(0, 1, 1)	(0,0,1,1)
(1, 0, 0)	(0,0,1,1)
(1, 0, 1)	(1,0,1,0)
(1, 1, 0)	(0,1,0,1)
(1, 1, 1)	(1,1,0,0)

As it can be seen, $h(\{0,1\}^3)$ has cardinality six.

Now, group the points of $\{0, \ldots, n-1\}$ four by four (when $n \geq 4$):

$$\begin{aligned} &A_0 = (0, 1, n/2, n/2+1), \\ &A_1 = (2, 3, n/2+2, n/2+3), \\ &\vdots \\ &A_{n/4-1} = (n/2-2, n/2-1, n-2, n-1) \ . \end{aligned}$$

Finally, form the cycles $c_0, \ldots, c_{n/4-1}$, with support respectively equal to the A_i, obtained from the cycle $(0, 2, 1, 3)$ by applying for each $0 \leq i < n/4$ the substitutions

$$0 \mapsto i, 1 \mapsto i+1, 2 \mapsto n/2+i, 3 \mapsto n/2+i+1 \ .$$

We are now ready to build recursively a keyed permutation χ, with the same method as in the proof of theorem 1. Following analogous notations, we define $\chi_{(K_1,K_2,E,F)}$ inductively as

$$\chi_{(K_1,K_2,E,F)} = \xi_F \circ \nu_E \circ \mu_{(K_1,K_2)} \ , \tag{4}$$

where

$$\xi_F = \prod_{j=0}^{n/4-1} c_j^{f_j} \ ,$$

with $F = (f_0, \ldots, f_{n/4-1}) \in \{0,1\}^{n/4}$.

Theorem 3. *χ is a $(n,k)-$keyed permutation, where $k = \frac{3}{4} n \log_2 n$. χ can be realized using k switches, and has degree of freedom at least $n/2-1$. Furthermore, χ generates at least a_n distinct permutations, where a_n verifies the recursion formula*

$$\begin{cases} a_n = 6^{n/4} a_{n/2}^2 & \text{if } n \geq 4 \\ a_2 = 2 \ . \end{cases}$$

Proof. We prove the recursion formula, the verification of the other points being straightforward. Suppose that we have constructed an $(n/2,k)-$keyed permutation σ that verifies our statements. We denoted by $\mathcal{E}$ the largest set of the keys such that σ restricted to $\mathcal{E}$ is injective. Referring to the recursive construction of χ of equation (4), it is clear that μ restricted to $\mathcal{E} \times \mathcal{E}$ is injective: this is a direct consequence of definition (3) of μ. Consider for each $0 \leq i < n/4$ the $3-$tuples $U_i = (e_{2i}, e_{2i+1}, f_i)$, and the set

$$\mathcal{A} = \big\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,1),(1,1,0)\big\} \ .$$

Using the truth table of h, we see that h restricted to $\mathcal{A}$ is injective. Consider the set $\mathcal{F}$ of the keys defined by $\mathcal{F} = \big\{(E,F)/\forall i \ \ U_i \in \mathcal{A}\big\}$. It is clear that

$$|\mathcal{F}| = |\mathcal{A}|^{n/4} = 6^{n/4} \ . \tag{5}$$

Now, since $\mu_{(K_1,K_2)}$ preserves the sets $\{0,\ldots,n/2-1\}$ and $\{n/2,\ldots,n-1\}$, we have that

$$\chi_K^{-1}(i) < n/2 \Longleftrightarrow (\xi_F \circ \nu_E)^{-1}(i) < n/2 \ .$$

This implies that χ restricted to $\mathcal{E} \times \mathcal{E} \times \mathcal{F}$ is injective. Using equality (5) and the fact that $|\mathcal{E}| \geq a_{n/2}$, this proves the theorem. □

It is easy to check by induction that $\log_2 a_n > (n \log_2 n)/2$, which means that χ generates effectively more permutations than δ. The explicit expression of a_n announced in section 3.1 results from the fact that the sequence $(\log_2 a_n)/n$ is in arithmetic progression.

Contrary to δ, the distribution of the permutations generated by χ_K, when K is chosen uniformly, is not uniform. We leave open the question of determining exactly this distribution. Anyway, it is easy to reduce the key space so that the

restriction of χ becomes injective. For that, it suffices to restrict the keys (E, F) to the set $\mathcal{F}$ defined in the proof above, and to proceed by induction. We leave open the question of the exact distribution of the generated permutations.

The practical realization of χ implies to design in hardware a keyed permutation with a cycle of length four, like $(0, 1, 2, 3)^b, b \in \{0, 1\}$. This can be easily done in one stage using four multiplexers

Figure 5 shows a realization of χ for the case $n = 8$. The nodes with 8 edges represent the cycle $(0, 2, 1, 3)$ involved in the construction of χ.

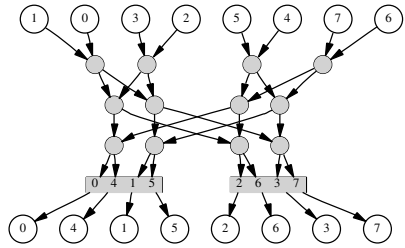

Fig. 5. Representation of χ for $n = 8$

4 Practical Examples and Applications

4.1 Numerical Examples

Table 1 shows the characteristics of the two keyed permutations δ and χ for various values of $n = 8, 16, 32, 64$. The number of multiplexers needed for their construction is denoted by N_{mux}, and the number of distinct generated permutations is denoted by N_{perm}.

Table 1. Characteristics of δ and χ for various values of n

	δ	χ	δ	χ	δ	χ	δ	χ
n	8		16		32		64	
N_{mux}	32	40	88	112	224	288	544	704
Depth	3	6	4	8	5	10	6	12
Key size	12	16	32	44	80	112	192	272
$\lceil \log_2 N_{\text{perm}} \rceil$	12	14	32	39	80	99	192	239
$\lceil \log_2 n! \rceil$	15		44		118		296	

An important fact is the very small number of stages needed to implement δ and χ. For example, for $n = 32$, the design has only five levels of gates. This property makes these functions particularly suitable for data scrambling in critical pathes. Non exhaustive applications are: scrambling of the bus between the microprocessor and the memory, scrambling of the RAM, or scrambling of the bus between the CPU and the cryptoprocessor.

4.2 Protecting the Secrecy of the Design

These functions can also easily be diversified, and thus provide a customizable design, so that the final scrambling function can remain secret. Recall that δ is built recursively from the equation

$$\delta_{(K_1,K_2,E)} = \nu_E \circ \mu_{(K_1,K_2)} \ .$$

This definition would correspond to the "normal form" of our construction. Derivated forms can be obtained as follows: at each step of the induction, we choose two permutations α_1, α_2, acting respectively on $\{0, \ldots, n/2-1\}$ and on $\{n/2, \ldots, n-1\}$, and we implement these permutations in hardware, that is, we permute physically the wires of the circuit. Here, α_1 and α_2 are supposed to be kept secret. With the same material, we can now build δ using the modified equation

$$\delta_{(K_1,K_2,E)} = \nu_E \circ \alpha_1 \circ \alpha_2 \circ \mu_{(K_1,K_2)} \ .$$

It is not difficult to see that we generate *mutatis mutandis* the same number of permutations as before, and that the resulting keyed permutation has the same degree of freedom. It suffices for this to rewrite the proofs of theorems 1 and 2. The same construction can be applied to χ, with the same consequences.

4.3 Non-linear Data Scrambling Using Keyed Permutations

The primitives that we have just described can easily be incorporated into more complex non-linear data scrambling functions. One major advantage of the proposed constructions is the large size of the key-space and of the resulting function space. Besides, the very compact shape of the resulting circuits allows to use them several times in more complex functions.

Following Shannon's basic confusion-diffusion paradigm, these keyed permutations can be used in alternating layers with small, say 4 bit to 4 bit substitution boxes (S-boxes). Clearly, such constructions cannot achieve the same security level as classical block ciphers do : following Shamir's security analysis [6], a five layer SASAS construction using alternating layers of S-boxes and affine functions (of which permutations are a special case) can be broken using approximately 2^{16} chosen plaintexts for 128 bit blocks and 8-bit to 8-bit Sboxes.

However, this kind of construction still yields a sub-exponential security bound instead of a linear security bound in terms of chosen plaintext attacks. As the attacker has quite limited resources in the probing setting anyhow, bearing in mind that she is not able to probe more than a handful of wires simultaneously using the same session scrambling key, a limited number of layers of additional key-dependent S-boxes will sufficiently increase the difficulty of unscrambling the memory and bus contents in the context of tamper-resistant objects such as smart-cards.

In terms of circuit complexity, a 4 bit to 4 bit S-box can be efficiently implemented using an average of 32 gates with a circuit depth three (in a completely optimized architecture this depth may become as low as one). Thus for the example SASAS structure for a 32 bit input size, each substitution layer adds

approximately 256 gates to the 224 gates of the proposed keyed permutation. With five layers altogether, the circuit has around 1200 gates for a depth of 19. It is then left to the designer to select whatever circuit complexity is acceptable in the concerned architecture compared to the obfuscation level fit for purpose.

5 Conclusion

We proposed three implementations of keyed permutations, which achieve very short depth, and effectively large key space. We indicated also a criterion to identify keyed permutations with good properties against chosen plaintext attacks realized by probing. These functions are particularly well suited for data obfuscation in very constrained environments like smart-cards.

Acknowledgments

We thank Guglielmo Morgari and Vittorio Bagini for careful reading, and David Naccache for fruitful discussions. We are also grateful to the anonymous referee for helpful comments.

References

1. Ross Anderson and Markus Kuhn. Tamper resistance – a Cautionary Note. In *The second USENIX Workshop on Electronic Commerce Proceeding*, pages 1–11, Oakland, California, November 1996.
2. Tamás Horvàth. Arithmetic Design for Permutation Groups. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES '99)*, number 1717 in Lecture Notes in Computer Science, pages 109–121. Springer Verlag, 1999.
3. Olivier Kömmerling and Markus Kuhn. Design principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology*, Chicago, Illinois, USA, May 1999.
4. S. Rankl and W. Effing. *Smart Card Handbook*. John Wiley & Sons, 1999.
5. Derek Robinson. *A Course in the Theory of Groups*. Number 80 in GTM. Springer Verlag, 1991.
6. Adi Shamir. Assassinating SASAS. Rump session of Crypto'2000.