A simple proof of ARX completeness

Adriano Koleci[1]

[1]Dipartimento di Scienze Matematiche, Politecnico di Torino
Corso Duca Degli Abruzzi 24, 10129, Torino, Italy
adriano.koleci.97@gmail.com

Abstract

In the recent years there has been a growing interest in ARX ciphers thanks to their performance in low cost architectures. This work is a short and simple proof that Add, Rotate and Exclusive-OR (ARX) operations generate the permutation group S_{2^n} and it is made up by elementary arguments with minimal use of group theory.

1 Introduction

ARX represents a class of cryptographic algorithms based only on three operations: Addition (ADD), Rotation(ROT) and Exclusive-OR(XOR) (hence the name ARX). Those operations are basic CPU instructions which combined can create fast and efficient algorithms. There is a wide variety of designs based on those operations, starting from block ciphers like SPECK [Bea+15] and stream ciphers like Salsa20 [Ber08] to hash functions such as MD4 [Riv92a] and MD5[Riv92b]. The advantage of such designs is their efficiency and they have gained popularity especially among lightweight ciphers whose application is crucial in low cost architectures. Although ARX based ciphers gained popularity in the recents years, it is important to note that those designs are not new since there are ARX ciphers that date back to the 80s and the 90s: some examples are block ciphers like FEAL[SM88] and RC5[Riv95].

ARX designs are often compared to S-Box based ciphers whose representatives are the very famous AES[DR02] and the much older DES [BD05]. The advantages of ARX ciphers over S-Box based ones come from the absence of table look-ups. From a security standpoint, the absence of table look-ups improves the resilience against cache-timing and side-channel attacks. From a performance standpoint, the advantage over S-Box designs comes from both their use of basic CPU instructions and the absence of tables since the latter reduces the memory usage which, again, is important in low cost architectures, making ARX designs good alternatives to S-Box based ciphers.

Cryptoanalysis of such systems, however, is not as mature as S-Box based ones and proving the completeness of ARX shows that those systems do not

have any generic property that holds for ARX ciphers. The aim of this work is to prove then that we can realize any permutation with ADD, ROT and XOR. Although this is a well known result, its only proof of this dates back to 1997 [Zie97]. This work provides a shorter and simpler proof since there is little use of group theory and it is made up only by elementary arguments.

2 ARX operations

In this section we define the ARX operations ADD_c, ROT and XOR_c. To make things easier it is useful to define the following bijection:

$$\begin{aligned} &\phi : (\mathbb{Z}/2\mathbb{Z})^n \to \mathbb{Z}/2^n\mathbb{Z} && (1)\\ &(x_0, .., x_{n-1}) \mapsto x_0 + 2x_1 + 2^2x_2 + ... + 2^{n-1}x_{n-1}. \end{aligned}$$

This bijection enables us to define ARX operations on either $(\mathbb{Z}/2\mathbb{Z})^n$ or $\mathbb{Z}/2^n\mathbb{Z}$, whichever is more convenient.

ADD_c

The operation ADD_c is defined as a bijection on $\mathbb{Z}/2^n\mathbb{Z}$:

$$\begin{aligned} \mathsf{ADD}_c : \mathbb{Z}/2^n\mathbb{Z} &\to \mathbb{Z}/2^n\mathbb{Z} && (2)\\ x &\mapsto x + c \mod 2^n && (3) \end{aligned}$$

ROT_d

To better understand the bijection ROT_d it is useful to see its effect on both $(\mathbb{Z}/2\mathbb{Z})^n$ and $\mathbb{Z}/2^n\mathbb{Z}$.
Bitwise it moves the last $n-1$ bits to the left putting the first bit in the last place:

$$\begin{aligned} \mathsf{ROT}_1 : (\mathbb{Z}/2\mathbb{Z})^n &\to (\mathbb{Z}/2\mathbb{Z})^n && (4)\\ (x_0, \cdots, x_{n-1}) &\mapsto (x_1, \cdots, x_{n-1}, x_0) && (5) \end{aligned}$$

hence the name *rotation* or *circular shift*.
On the other hand we can see ROT as a piecewise function:

$$\mathsf{ROT} : \mathbb{Z}/2^n\mathbb{Z} \to \mathbb{Z}/2^n\mathbb{Z} \quad (6)$$

$$\mathsf{ROT}(x) = \begin{cases} 2x \mod 2^n & \text{if } 0 \le x < 2^{n-1} \quad (7)\\ 2x+1 \mod 2^n & \text{if } 2^{n-1} \le x \le 2^n - 1 \quad (8) \end{cases}$$

We can apply multiple times the operation ROT and it will be denoted by ROT_d. Note that if the integer $x \in \mathbb{Z}/2^n\mathbb{Z}$ is even (i.e. bitwise the least significant bit is 0) we have

$$\mathsf{ROT}_{n-1}(x) = x/2 \mod 2^n. \quad (9)$$

If the integer $x \in \mathbb{Z}/2^n\mathbb{Z}$ is odd (i.e. bitwise the least significant bit is 1) we have

$$\mathsf{ROT}_{n-1}(x) = \frac{x-1}{2} + 2^{n-1} \mod 2^n. \tag{10}$$

XOR_c

The operation XOR_c is a bijection that can be defined as follows:

$$\mathsf{XOR}_c : (\mathbb{Z}/2\mathbb{Z})^n \to (\mathbb{Z}/2\mathbb{Z})^n \tag{11}$$
$$(x_0, \cdots, x_{n-1}) \mapsto (x_0 + x_{c,0}, \cdots, x_{c,d-1}) \tag{12}$$

which is, bitwise, modulo 2 addition.
Observe that if the integer $x \in \mathbb{Z}/2^n\mathbb{Z}$ is odd,

$$\mathsf{XOR}_1 = x - 1 \mod 2^n. \tag{13}$$

To prove that $\{\mathsf{ADD}_1, \mathsf{ROT}, \mathsf{XOR}_1\}$ can create any permutation on $\mathbb{Z}/2^n\mathbb{Z}$ we will first define a reference transposition.

Lemma 1. *The bijection*

$$\mathsf{ROT}(\mathsf{ADD}_{2^n-1}(\mathsf{ROT}_{n-1}(\mathsf{ADD}_2(x)))) \tag{14}$$

is the transposition $(2^n - 2, 2^n - 1)$ *on* $\mathbb{Z}/2^n\mathbb{Z}$.

Proof. By recalling bijection (1) we have that $(2^n - 2, 2^n - 1)$ can be expressed in bits as $(11\cdots10, 11\cdots11)$. Bitwise the operation (14) transposes $2^n - 2$ with $2^n - 1$ and viceversa thanks to the carry of modulo-2^n addition. Since it is more convenient here we will use the bit representation. Evaluating the function (14) for $x = 11\cdots10$ we get:

$$\begin{aligned}
\mathsf{ADD}_2(11\cdots10) &= 00\cdots00 \\
\mathsf{ROT}_{n-1}(00\cdots00) &= 00\cdots00 \\
\mathsf{ADD}_{2^n-1}(00\cdots00) &= 11\cdots11 \\
\mathsf{ROT}(11\cdots11) &= 11\cdots11.
\end{aligned}$$

Conversely, by evaluating it for $x = 11\cdots11$:

$$\begin{aligned}
\mathsf{ADD}_2(11\cdots10) &= 00\cdots01 \\
\mathsf{ROT}_{n-1}(00\cdots00) &= 10\cdots00 \\
\mathsf{ADD}_{2^n-1}(00\cdots00) &= 01\cdots11 \\
\mathsf{ROT}(01\cdots11) &= 11\cdots10.
\end{aligned}$$

Then we have to prove that the bijection (14) acts as identity if applied to any other number. Here the representation in $\mathbb{Z}/2^n\mathbb{Z}$ is easier to understand. Let

us say that x is an even integer such that $0 \leq x < 2^n - 2$.

$$\mathsf{ADD}_2(x) = x + 2 \qquad \bmod 2^n \tag{15}$$
$$\mathsf{ROT}_{n-1}(x+2) = x/2 + 1 \qquad \bmod 2^n \tag{16}$$
$$\mathsf{ADD}_{2^n-1}(x/2+1) = x/2 \qquad \bmod 2^n \tag{17}$$
$$\mathsf{ROT}(x/2) = x \qquad \bmod 2^n. \tag{18}$$

If the integer $0 \leq x < 2^n - 2$ is odd we get

$$\mathsf{ADD}_2(x) = x + 2 \qquad \bmod 2^n \tag{19}$$
$$\mathsf{ROT}_{n-1}(x+2) = \frac{x-1}{2} + 2^{n-1} \qquad \bmod 2^n \tag{20}$$
$$\mathsf{ADD}_{2^n-1}(x/2+1) = \frac{x-1}{2} + 2^{n-1} - 1 \qquad \bmod 2^n \tag{21}$$
$$\mathsf{ROT}(\frac{x-1}{2} + 2^{n-1} - 1) = x \qquad \bmod 2^n. \tag{22}$$

By doing this we have proven that the function (14) is the transposition $(2^n - 2, 2^n - 1)$.

□

Theorem 1. *The set* $\{\mathsf{ADD}_1, \mathsf{ROT}, \mathsf{XOR}_1\}$ *generates the permutation group* S_{2^n}.

Proof. An idea of this theroem has already been given in a competition called Capture the Flag for the set $(\mathbb{Z}/2\mathbb{Z})^8$ [Ano18]. Since any permutation can be seen as a product of transpositions, in order to prove that $\{\mathsf{ADD}_1, \mathsf{ROT}, \mathsf{XOR}_1\}$ generates the permutation group S_{2^n} it is sufficient to prove that it generates any transposition (a, b).
The first step is to apply the bijection

$$\mathsf{ADD}_{2^n-a}(x). \tag{23}$$

Applying such an operation on a we get

$$\mathsf{ADD}_{2^n-a}(a) = 0 \mod 2^n \tag{24}$$

whereas applying it to b leads to

$$\mathsf{ADD}_{2^n-a}(b) = b - a \mod 2^n. \tag{25}$$

The next step is to reduce the result in equation (25) to 1 and we can do this by iterating a combination of ROT and XOR:

1. apply ROT_{n-1} until the result is odd;
2. if the result is not 1 perform XOR_1 then ADD_{2^n-1}.

Iterate the above until the result is 1 and thanks to the observations in (10) and (13) we are sure that the result in (25) is reducing.
Those operations do not change the result (24) since at each iteration we have

$$\mathsf{ROT}_{n-1}(0) = 0 \tag{26}$$
$$\mathsf{ADD}_{2^n-1}(\mathsf{XOR}_1(0)) = 0. \tag{27}$$

By applying ADD_{2^n-2} we have built a bijection such that

$$\begin{aligned} a &\iff 2^n - 2 \\ b &\iff 2^n - 1. \end{aligned} \tag{28}$$

which can be easily transposed by recalling Lemma 1.
The final step is to apply the bijection (28) in reverse in order to get (b, a).
Since $(2^n - 2, 2^n - 1)$ is a transposition, if we apply the bijection (28) to any other number different from a or b we get two numbers different from $2^n - 2$ and $2^n - 1$ hence (a, b) is a transposition.

□

References

[SM88] Akihiro Shimizu and Shoji Miyaguchi. "Fast Data Encipherment Algorithm FEAL". In: *Advances in Cryptology — EUROCRYPT' 87*. Ed. by David Chaum and Wyn L. Price. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 267–278. ISBN: 978-3-540-39118-0.

[Riv92a] Ronald L. Rivest. *The MD4 Message-Digest Algorithm*. RFC 1320. Apr. 1992. DOI: 10.17487/RFC1320. URL: https://www.rfc-editor.org/info/rfc1320.

[Riv92b] Ronald L. Rivest. *The MD5 Message-Digest Algorithm*. RFC 1321. Apr. 1992. DOI: 10.17487/RFC1321. URL: https://www.rfc-editor.org/info/rfc1321.

[Riv95] Ronald L. Rivest. "The RC5 encryption algorithm". In: *Fast Software Encryption*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 86–96. ISBN: 978-3-540-47809-6.

[Zie97] Thilo Zieschang. "Combinatorial Properties of Basic Encryption Operations". In: *Advances in Cryptology — EUROCRYPT '97*. Ed. by Walter Fumy. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 14–26. ISBN: 978-3-540-69053-5.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Berlin, Heidelberg: Springer-Verlag, 2002. ISBN: 3540425802.

[BD05] Alex Biryukov and Christophe De Cannière. "Data encryption standard (DES)". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg. Boston, MA: Springer US, 2005, pp. 129–135. ISBN: 978-0-387-23483-0. DOI: 10.1007/0-387-23483-7_94. URL: https://doi.org/10.1007/0-387-23483-7_94.

[Ber08] Daniel J. Bernstein. "The Salsa20 Family of Stream Ciphers". In: *New Stream Cipher Designs: The eSTREAM Finalists*. Ed. by Matthew Robshaw and Olivier Billet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 84–97. ISBN: 978-3-540-68351-3. DOI: 10.1007/978-3-540-68351-3_8. URL: https://doi.org/10.1007/978-3-540-68351-3_8.

[Bea+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. "The SIMON and SPECK Lightweight Block Ciphers". In: *Proceedings of the 52nd Annual Design Automation Conference*. DAC '15. San Francisco, California: Association for Computing Machinery, 2015. ISBN: 9781450335201. DOI: 10.1145/2744769.2747946. URL: https://doi.org/10.1145/2744769.2747946.

[Ano18] Anonymous. *Capture The Flag*. 2018. URL: https://ctf-wiki.org/crypto/blockcipher/arx-operations/.