
Searching the space of tower field implementations of the \mathbb{F}_{2^8} inverter – with applications to AES, Camellia and SM4

Zihao Wei, Siwei Sun*, Lei Hu and Man Wei

State Key Laboratory of Information Security,
Institute of Information Engineering,
Chinese Academy of Sciences,
Beijing 100093, China
and

School of Cyber Security,
University of Chinese Academy of Sciences,
Beijing 100049, China
Email: weizihao@iie.ac.cn
Email: sunsiwei@iie.ac.cn
Email: hulei@iie.ac.cn
Email: weiman@iie.ac.cn
*Corresponding author

René Peralta

Computer Security Division,
NIST,
100 Bureau Drive, Stop 8930,
Gaithersburg, MD 20899-8930, USA
Email: rene.peralta@nist.gov

Abstract: The tower field implementation of the \mathbb{F}_{2^8} inverter is not only the key technique for compact implementations of the S-boxes of several internationally standardised block ciphers such as AES, Camellia, and SM4, but also the underlying structure many side-channel attack resistant AES implementations rely on. In this work, we conduct an exhaustive study of the tower field representations of the \mathbb{F}_{2^8} inverter with normal bases by applying several state-of-the-art combinatorial logic minimisation techniques. As a result, we achieve improved implementations of the AES, Camellia and SM4 S-boxes in terms of area footprint. Surprisingly, we are still able to improve the currently known most compact implementation of the AES S-box from CHES 2018 by 5.5 GE, beating the record again. For Camellia and SM4, the improvements are even more significant. The Verilog codes of our implementations of the AES, Camellia and SM4 S-boxes are openly available.

Keywords: tower field; inverter; S-box; AES; Camellia; SM4.

Reference to this paper should be made as follows: Wei, Z., Sun, S., Hu, L., Wei, M. and Peralta, R. (2023) ‘Searching the space of tower field implementations of the \mathbb{F}_{2^8} inverter – with applications to AES, Camellia and SM4’, *Int. J. Information and Computer Security*, Vol. 20, Nos. 1/2, pp.1–26.

Biographical notes: Zihao Wei received his BS in Computer Science from the Xidian University, Xian, China, in 2015. Currently, he is working toward his PhD at the School of Cyber Security, University of Chinese Academy of Sciences, China. His research interests include efficient and secure hardware implementation of cryptographic primitives and cryptanalysis of lightweight cipher.

Siwei Sun is an Associate Professor at the Institute of Information Engineering, Chinese Academy of Sciences, where his main research interest includes symmetric-key cryptography, automatic cryptanalysis and implementation, and cryptanalysis with quantum algorithms. He holds a PhD in Computer Science from the University of Chinese Academy of Sciences, Beijing.

Lei Hu received his BS and MS from the Peking University, Beijing, China, in 1988 and 1991, respectively, and received his PhD from the Chinese Academy of Sciences in 1994. Since 2002, he has been a Professor at the Chinese Academy of Sciences. His research interest includes cryptograph and information security.

Man Wei received her BS in Information and Computing Sciences from the Nankai University, Tianjin, China, in 2015. She is currently working toward her PhD at the School of Cyber Security, University of Chinese Academy of Sciences, China. Her research interests include side channel attack and protection.

René Peralta is a computer scientist in the Cryptographic Technology Group at NIST, where he is mainly engaged with the interoperable randomness beacons, privacy enhancing technologies, circuit complexity, and post-quantum cryptography projects. He holds a PhD in Computer Science from the University of California, Berkeley. He has taught at various universities in the USA, Chile, and Japan.

1 Introduction

For encrypting and authenticating the largest part of the workload of today’s secure communication, symmetric-key primitives are regarded as the crypto workhorse (whereas public-key schemes are generally used for setting up the session keys). In many cases, the components of symmetric-key schemes are built on operations over finite fields. Since the symmetric-key cryptographic algorithms will eventually be implemented in software or hardware to play a role in the real world, it is of great importance to investigate how to implement their common operations efficiently (Beierle et al., 2016; Boyar et al., 2013; Jean et al., 2017b; Stoffelen, 2016; Li et al., 2019; Tan and Peyrin, 2020). For instance, due to the rapid development of lightweight IoT

devices, ongoing efforts have been made to obtain more compact ASIC implementations of symmetric-key ciphers (Banik et al., 2016a, 2016b; Jean et al., 2017a). Just recently, the most compact implementation of the MixColumns and the S-box of AES were reported at FSE 2018 (Kranz et al., 2017) and CHES 2018 (Reyhani-Masoleh et al., 2018) respectively.

In this work, we focus on area-optimised implementations of the multiplicative inverse operation (and its affine equivalences) over \mathbb{F}_{2^8} . The AES S-box, which is affine equivalent to the \mathbb{F}_{2^8} inverter, is the strongest 8×8 S-box known so far in terms of local security properties (i.e., nonlinearity, differential uniformity, algebraic degree, etc.). Several internationally standardised block ciphers, such as Camellia and SM4, apply variants of the AES S-box in their designs, which are all affine equivalent to the \mathbb{F}_{2^8} inverter. Despite its desirable local cryptographic properties, to implement the AES S-box in ASIC with small footprint is not a trivial task. The naive approach that encodes the AES S-box as a look-up table in a hardware description language and produces the actual circuit relying on open-source or commercial CAD tools will certainly lead to unsatisfactory results for many resource constrained applications. Today's most compact ASIC implementations of the AES S-box are based on the tower field architecture, where the operations over $\mathbb{F}_{2^{2k}}$ are represented with operations over \mathbb{F}_{2^k} recursively.

Moreover, several cost-effective threshold implementations of the AES S-box with resistance against side-channel attacks are built on top of the tower field architecture (Bilgin et al., 2014, 2015; Cnudde et al., 2016; Moradi et al., 2011). In threshold implementations, the most important design consideration includes the security level, number of fresh random bits required, and area consumption. Therefore, providing different implementations of the tower field structure without increasing the circuit footprint potentially offers more flexible area-randomness-security trade-off in threshold implementations.

Apart from these, breaking the AES S-box into several layers with the tower field architecture allows registers to be inserted into the middle of the computation such that the critical path can be reduced, and therefore the frequency of the system clock can be increased to boost the performance.

1.1 Related work

The tower field architecture was first proposed by Itoh and Tsujii for computing multiplicative inverse in finite fields of characteristic two (Itoh and Tsujii, 1988). At the beginning, it was applied in developing efficient implementations of public-key cryptographic algorithms involving inverse operations over \mathbb{F}_{2^n} (Gujardo and Paar, 1997; Paar and Soria-Rodriguez, 1997). Later, after the development of the advanced encryption standard (AES) – a block cipher using an S-box affine equivalent to the \mathbb{F}_{2^8} multiplicative inverter (Daemen and Rijmen, 2002), the tower field architecture found applications in compact hardware implementations of AES (Mentens et al., 2005; Rudra et al., 2001; Satoh et al., 2001; Wolkerstorfer et al., 2002). After a series of improvements, Canright (2005a, 2005b), Boyar et al. (2013) and Circuit Minimization Team (CMT) (<http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html>) achieved the most compact implementations at the time, which has become the *de facto* standard for compact implementations of the AES S-box. Such tower field implementations of the AES S-box were also intensively applied in side-channel resistant implementations of AES to reduce resource consumption. Recently Reyhani-Masoleh et al. (2018) broke

the record set by Canright and Boyar et al., presenting so far the most compact ASIC implementation of the AES S-box, which costs 182.25 GE under the STM 65 nm CMOS technology.

Due to the strong local cryptographic properties of the AES S-box, several well known block ciphers employ affine equivalences of the AES S-box as their S-boxes, including Camellia and SM4. Therefore, the technique of tower field implementation naturally applies to these ciphers (Abbasi and Afzal, 2011; Bai et al., 2009; Martínez-Herrera et al., 2012, 2013; Satoh and Morioka, 2003).

Table 1 Previous tower field implementations of the AES, Camellia and SM4 S-boxes, where \mathcal{P} means a polynomial basis is used, \mathcal{N} means a normal basis is used, and #Cases denotes the number of cases considered

<i>Cipher</i>	<i>Source</i>	<i>Tower field architecture and basis</i>	<i>#Cases</i>
AES	Rudra et al. (2001) and Wolkerstorfer et al. (2002)	$\mathbb{F}_2 \xrightarrow{w^4+w+1} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+C_1}{\mathcal{P}}} \mathbb{F}_{2^8}$	1
	Satoh et al. (2001)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{P}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+C_2}{\mathcal{P}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+C_3}{\mathcal{P}}} \mathbb{F}_{2^8}$	1
	Mentens et al. (2005)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{P}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+C_2}{\mathcal{P}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+\nu}{\mathcal{P}}} \mathbb{F}_{2^8}$	64
	Canright (2005b)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+N}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+\nu}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^8}$	432
	Boyar et al. (2013)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{N}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+C_4}{\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+C_5}{\mathcal{N}}} \mathbb{F}_{2^8}$	1
	Reyhani-Masoleh et al. (2018)	$\mathbb{F}_2 \xrightarrow{\frac{w^4+w^3+w^2+w+1}{\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+\nu}{\mathcal{N}}} \mathbb{F}_{2^8}$	32
Camellia	Satoh and Morioka (2003)	$\mathbb{F}_2 \xrightarrow{w^4+w+1} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+C_6}{\mathcal{P}}} \mathbb{F}_{2^8}$	1
	Martínez-Herrera et al. (2012)	$\mathbb{F}_2 \xrightarrow{\frac{w^4+w+1}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+\tau y+\nu}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^8}$	13
		$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+Tz+N}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+\tau y+\nu}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^8}$	
SM4	Martínez-Herrera et al. (2013)	$\mathbb{F}_2 \xrightarrow{\frac{w^4+w+1}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+C_7y+C_8}{\mathcal{P}/\mathcal{N}}} \mathbb{F}_{2^8}$	4
	Bai et al. (2009)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{P}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+C_9}{\mathcal{P}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+C_{10}}{\mathcal{P}}} \mathbb{F}_{2^8}$	1
	Abbasi and Afzal (2011)	$\mathbb{F}_2 \xrightarrow{\frac{w^2+w+1}{\mathcal{N}}} \mathbb{F}_{2^2} \xrightarrow{\frac{z^2+z+N}{\mathcal{N}}} \mathbb{F}_{2^4} \xrightarrow{\frac{y^2+y+\nu}{\mathcal{N}}} \mathbb{F}_{2^8}$	16

In tower field implementations, a sequence of field extensions starting from \mathbb{F}_2 and ending at \mathbb{F}_{2^8} of the type $\mathbb{F}_{2^k} \subseteq \mathbb{F}_{(2^k)2^l}$ is considered. At each level of the field extension, an irreducible polynomial over \mathbb{F}_{2^k} and a corresponding basis of $\mathbb{F}_{(2^k)2^l}$ over \mathbb{F}_{2^k} have to be specified. The irreducible polynomials and bases induce a basis of \mathbb{F}_{2^8} over \mathbb{F}_2 . The tower field architecture is implemented over this new basis with

proper basis transformations to maintain the original field representation. Therefore, the choices of the field extensions, the corresponding irreducible polynomials and the bases determine the overall cost of the implementation. A summary of the choices of existing work is given in Table 1 for AES, Camellia and SM4 respectively.

1.2 Contributions

As shown in Table 1, only a part of the design space of tower field implementation was explored by choosing irreducible polynomials of special forms in previous work. In particular, previous work preferred a class of parameter choices where the irreducible polynomials selected for the field extension $\mathbb{F}_{2^2} \subseteq \mathbb{F}_{2^4} \subseteq \mathbb{F}_{2^8}$ are of the form $z^2 + z + N$ and $y^2 + y + \nu$, and indeed the most well known implementations of Canright's and Boyar et al.'s schemes are in this class (Boyar et al., 2013; Canright, 2005b). The preference for this special class is reasonable, since with these choices of parameters, the implementations of some subcomponents of the circuit are free. Despite this heuristic, there is no concrete evidence that this configuration will result in optimal implementations. Therefore, we exhaustively examine all possible tower field representations under normal bases induced by irreducible polynomials (720 cases in total), and find several cases which were never considered previously enjoy the most compact implementations. Along the way, we do not only apply well-known logic minimisation techniques from Canright and Boyar et al., but also resort to several state-of-the-art combinatorial logic minimisation techniques (Fuhs and Schneider-Kamp, 2010; Jean et al., 2017b; Stoffelen, 2016) developed in recent years. As a result, we beat the new record set by the work of Reyhani-Masoleh et al. (2018) for compact implementations of the AES S-box. Moreover, the implementations of the Camellia and SM4 S-boxes are improved significantly, and we refer reader to Table 2 for a summary of the results. Naturally, these results serve to achieve more compact implementations of AES, Camellia and SM4, and potentially provide more flexible security-randomness-area trade-offs for threshold implementations of these block ciphers. The Verilog codes of our implementation of the AES, Camellia and SM4 S-boxes are provided in Appendix.

1.3 Organisation

In Section 2, we give a brief introduction of the mathematical background of the tower field representations under different bases, as well as the frequently-used logic gates for constructing digital circuits. Subsequently, we describe the details of the tower field implementation of the \mathbb{F}_{2^8} inverter in Section 3. In Section 4, we apply state-of-the-art logic minimisation techniques to a list of tower field representations of the AES, Camellia and SM4 S-box under all possible normal bases. As a result, we obtain so far the most compact implementations of these S-boxes. We conclude the paper in Section 5 and propose possible future work. The source codes of the optimised implementations for the S-boxes of AES, Camellia and SM4 are provided in Appendix.

Table 2 Synthesised results, where the functionalities of some uncommon gates (e.g., XOR3, OAI21, AOI21, etc.) are described in Section 2

Cipher	Source	Gates used										Synthesis results			
		XOR/XNOR	XOR3	MAJ3	AND	MAJ3	NOR	NOT	OAI21	AOI21	OAI32	SMIC 130 nm	SMIC 65 nm	STM 65 nm	Hangate 45 nm
AES	Rudra et al. (2001)	111			58							336.33	336.75	294.50	299.33
	Sato et al. (2001)	100			36						281.33	279.00	245.00	248.00	
	Mentens et al. (2005)	96			36						272.00	270.00	237.00	240.00	
	Canright (2005b)	80		34		6					226.67	220.00	200.00	200.00	
	Boyar et al. (2013)	83			32						236.33	234.75	206.00	208.67	
	Circuit Minimization Team (CMT)	81			32						231.67	230.25	202.00	204.67	
		81		32							221.00	214.25	194.00	194.00	
	Reyhani-Masoleh et al. (2018)	69		39		4	3	4			211.00	205.25	188.00	188.00	
		69		31			3	5	7	1	N/A	N/A	N/A	186.00	
		63		27			7	4		4	N/A	N/A	182.25	N/A	
Ours		69		33		8					202.00	196.25	179.00	179.00	
		51		33		8					N/A	N/A	176.75	N/A	
		113		9			9				316.33	313.50	276.50	278.67	
Camellia	Martínez-Herrera et al. (2012)	68		33		8	1				200.33	194.75	177.75	177.67	
SM4	Ours	99			58		8				315.67	318.00	278.75	282.67	
	Martínez-Herrera et al. (2013)	157			63		11				450.33	447.75	392.75	398.00	
	Bai et al. (2009)	134			36						360.67	355.50	313.00	316.00	
	Abbasi and Afzal (2011)	66		32		9	1				195.67	190.25	173.75	173.67	

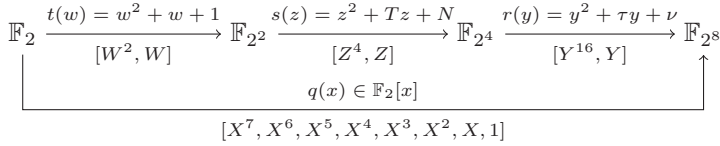
2 Preliminaries

We first give a brief introduction of the tower field representation. Then we list a set of gates together with their functionalities and areas. These gates will be used to implement the circuits constructed in this paper, and the overall area of each circuit will be computed accordingly.

2.1 Tower field representation

Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field of two elements. It is well known that the field \mathbb{F}_{2^k} with 2^k elements can be induced by an irreducible polynomial $q(x) \in \mathbb{F}_2[x]$ with degree k , i.e., $\mathbb{F}_{2^k} \cong \mathbb{F}_2[x]/(q(x))$. Assuming that X is a root of $q(x)$ over \mathbb{F}_{2^k} , then every element in \mathbb{F}_{2^k} can be represented as an \mathbb{F}_2 -linear combination $b_{k-1}X^{k-1} + \dots + b_1X + b_0$ of $[X^{k-1}, \dots, X, X^0]$, which is a *polynomial basis* of \mathbb{F}_{2^k} over \mathbb{F}_2 . To be concrete, we take $k = 8$, and we call (b_7, \dots, b_0) the bit-vector representation of $b_{k-1}X^{k-1} + \dots + b_1X + b_0$ under the basis $[X^7, \dots, X, X^0]$.

Figure 1 The tower field structure



Considering a sequence of field extensions $\mathbb{F}_2 \subseteq \mathbb{F}_{2^2} \subseteq \mathbb{F}_{2^4} \subseteq \mathbb{F}_{2^8}$ shown in Figure 1. Let $r(y) \in \mathbb{F}_{2^4}[y]$, $s(z) \in \mathbb{F}_{2^2}[z]$ and $t(w) \in \mathbb{F}_2[w]$ be irreducible polynomials over their respective fields, and let $Y \in \mathbb{F}_{2^8}$, $Z \in \mathbb{F}_{2^4}$ and $W \in \mathbb{F}_{2^2}$ be roots of $r(y)$, $s(z)$ and $t(w)$ over the corresponding fields respectively. Then we obtain a set of normal basis: $[Y^{16}, Y]$ is a basis of \mathbb{F}_{2^8} over \mathbb{F}_{2^4} , $[Z^4, Z]$ is a basis of \mathbb{F}_{2^4} over \mathbb{F}_{2^2} , and $[W^2, W]$ is a basis of \mathbb{F}_{2^2} over \mathbb{F}_2 . Therefore, for an element $b = b_7X^7 + \dots + b_1X + b_0 \in \mathbb{F}_{2^8}$ we have

$$\begin{aligned}
 b &= \gamma_1 Y^{16} + \gamma_0 Y, \gamma_1, \gamma_0 \in \mathbb{F}_{2^4}, \\
 \gamma_1 &= \Gamma_3 Z^4 + \Gamma_2 Z, \\
 \gamma_0 &= \Gamma_1 Z^4 + \Gamma_0 Z, \Gamma_3, \Gamma_2, \Gamma_1, \Gamma_0 \in \mathbb{F}_{2^2}, \\
 \Gamma_3 &= g_7 W^2 + g_6 W, \\
 \Gamma_2 &= g_5 W^2 + g_4 W, \\
 \Gamma_1 &= g_3 W^2 + g_2 W, \\
 \Gamma_0 &= g_1 W^2 + g_0 W, g_i \in \mathbb{F}_2, 0 \leq i \leq 7,
 \end{aligned}$$

which implies

$$\begin{aligned}
 b &= b_7 X^7 + \dots + b_1 X + b_0 \\
 &= g_7 W^2 Z^4 Y^{16} + g_6 W Z^4 Y^{16} + g_5 W^2 Z Y^{16} + g_4 W Z Y^{16} \\
 &\quad + g_3 W^2 Z^4 Y + g_2 W Z^4 Y + g_1 W^2 Z Y + g_0 W Z Y.
 \end{aligned}$$

That is, b can be represented as (g_7, \dots, g_0) under the *tower basis*

$$\mathcal{TB} = [W^2Z^4Y^{16}, WZ^4Y^{16}, W^2ZY^{16}, WZY^{16}, W^2Z^4Y, \\ WZ^4Y, W^2ZY, WZY]$$

induced by W , Z and Y . We call (g_7, \dots, g_0) the bit-vector representation of b under the tower basis. Assuming that the tower basis \mathcal{TB} can be represented by the original polynomial basis with a matrix $M_t \in \mathbb{F}_2^{8 \times 8}$ as

$$\mathcal{TB} = (X^7, \dots, X^0)M_t,$$

we have

$$(b_7, \dots, b_0)^\top = M_t \cdot (g_7, \dots, g_0)^\top \text{ or } (g_7, \dots, g_0)^\top = M_t^{-1} \cdot (b_7, \dots, b_0)^\top.$$

Therefore, we can change the representations by multiplying M_t or M_t^{-1} , and we call M_t the *basis transformation matrix*.

Considering the example from AES shown in Figure 1, where $q(x)$ is the Rijndael polynomial

$$\begin{aligned} q(x) &= x^8 + x^4 + x^3 + x + 1, \\ \tau &= X^7 + X^5 + X^4 + X^3 + X^2 + 1, \\ \nu &= X^7 + X^6 + X^5, \\ T &= X^7 + X^5 + X^4 + X^3 + X^2 + 1, \\ N &= 1, \\ W &= X^7 + X^5 + X^4 + X^3 + X^2, \\ Z &= X^6 + X^4, \\ Y &= X^6 + X^3. \end{aligned}$$

Then we have $\mathcal{TB} = (X^7, \dots, X^0) \cdot M_t$, where

$$M_t = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

2.2 Frequently-used gates

The circuits of this paper are eventually synthesised with the gates provided in common cell libraries. We list a set of frequently-used gates in Table 3, where the area is measured in gate equivalence (GE), corresponding to the area of a two-input drive-strength-one NAND gate.

Note that apart from those common gates (XOR, XNOR, AND, NAND, OR, NOR, NOT) which are available in almost all CMOS technology libraries, we also list some compound gates (XOR3, NAND3, OAI21, AOI21, OAI32).

The data of STM 65 nm library is collected from Reyhani-Masoleh et al.'s (2018) paper, while the others comes from library files and databooks. The cell in blank of STM 65 nm means the corresponding gate does not appear at Reyhani-Masoleh et al. (2018), and the cell labelled as N/A means the library does not support this kind of gate.

Table 3 Frequently-used gates in common CMOS technology libraries

Gate	Area (GE)			
	SMIC 130 nm	SMIC 65 nm	STM 65 nm	Nangate 45 nm
XOR: $(a, b) \mapsto a \oplus b$	2.33	2.25	2	2
XNOR: $(a, b) \mapsto a \odot b$	2.33	2.25	2	2
XOR3: $(a, b, c) \mapsto a \oplus b \oplus c$	5.67	4.75	3.75	N/A
AND: $(a, b) \mapsto a \cdot b$	1.33	1.5	1.25	1.33
NAND: $(a, b) \mapsto \overline{a \cdot b}$	1	1	1	1
NAND3: $(a, b, c) \mapsto \overline{a \cdot b \cdot c}$	1.33	1.25	1.25	1.33
OR: $(a, b) \mapsto a b$	1.33	1.5	1.25	1.33
NOR: $(a, b) \mapsto \overline{a b}$	1	1	1	1
NOT: $a \mapsto \bar{a}$	0.66	0.75	0.75	0.66
OAI21: $(a, b, c) \mapsto \overline{(a b) \cdot c}$	1.67	1.5		1.33
AOI21: $(a, b, c) \mapsto \overline{(a \cdot b) c}$	1.67	1.5		1.33
OAI32: $(a, b, c, d, e) \mapsto \overline{(a b c) \cdot (d e)}$	2.33	N/A	2	N/A

3 Tower field implementation of the \mathbb{F}_{2^8} inverter

In this section, we give an introduction to the tower field implementation of the \mathbb{F}_{2^8} inverter. Please note that the derivation of these results can all be found in Canright's paper (Canright, 2005a, 2005b).

Consider the field extension $\mathbb{F}_{2^4} \subseteq \mathbb{F}_{2^8}$ with an irreducible polynomial $r(y) = y^2 + \tau y + \nu \in \mathbb{F}_{2^4}[y]$. Let $Y \in \mathbb{F}_{2^8}$ be a root of $r(y)$. Then Y^{16} and Y form a normal basis, and every element $g \in \mathbb{F}_{2^8}$ can be represented as $g = \gamma_1 Y^{16} + \gamma_0 Y$, where $\gamma_1, \gamma_0 \in \mathbb{F}_{2^4}$. Let $g^{-1} = \delta_1 Y^{16} + \delta_0 Y$ with $\delta_1, \delta_0 \in \mathbb{F}_{2^4}$ be the inverse of g . By solving the equation

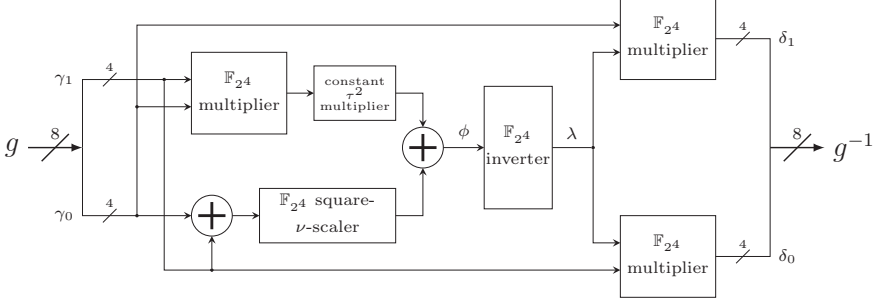
$$g \cdot g^{-1} = (\gamma_1 Y^{16} + \gamma_0 Y)(\delta_1 Y^{16} + \delta_0 Y) = 1$$

for δ_1 and δ_0 , we obtain

$$\begin{aligned} \delta_1 &= [\gamma_1 \gamma_0 \tau^2 + (\gamma_1 + \gamma_0)^2 \nu]^{-1} \gamma_0 \\ \delta_0 &= [\gamma_1 \gamma_0 \tau^2 + (\gamma_1 + \gamma_0)^2 \nu]^{-1} \gamma_1. \end{aligned}$$

Therefore, given $r(y)$ and the basis $[Y^{16}, Y]$, we can compute $g^{-1} = (\delta_1, \delta_0)$ from $g = (\gamma_1, \gamma_0)$ using operations over \mathbb{F}_{2^4} , which is illustrated in Figure 2, where $\phi = \gamma_1\gamma_0\tau^2 + (\gamma_1 + \gamma_0)^2\nu$ and $\lambda = \phi^{-1}$.

Figure 2 The \mathbb{F}_{2^8} inverter



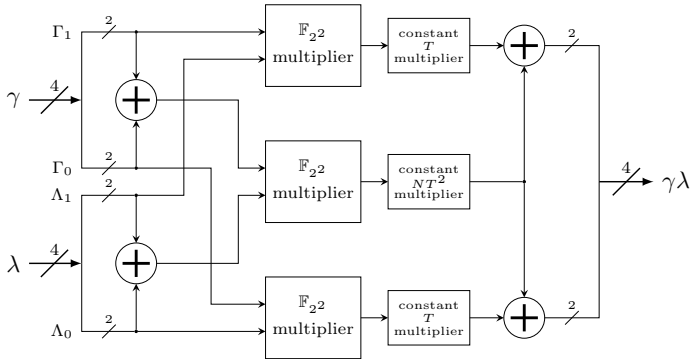
3.1 Multiplication and inverse over \mathbb{F}_{2^4}

Extend \mathbb{F}_{2^2} to \mathbb{F}_{2^4} with an irreducible polynomial $s(z) = z^2 + Tz + N \in \mathbb{F}_{2^2}[z]$, and let $Z \in \mathbb{F}_{2^4}$ be a root of $s(z)$. Then every element in \mathbb{F}_{2^4} can be represented as an \mathbb{F}_{2^2} -linear combination of the normal basis $[Z^4, Z]$. Let $\gamma = \Gamma_1 Z^4 + \Gamma_0 Z$, and $\lambda = \Lambda_1 Z^4 + \Lambda_0 Z$, where $\Gamma_i, \Lambda_j \in \mathbb{F}_{2^2}$. Then the multiplication of γ and λ can be calculated as

$$\begin{aligned} \gamma\lambda &= (\Gamma_1 Z^4 + \Gamma_0 Z)(\Lambda_1 Z^4 + \Lambda_0 Z) \\ &= [\Gamma_1 \Lambda_1 T + (\Gamma_1 + \Gamma_0)(\Lambda_1 + \Lambda_0)NT^2]Z^4 \\ &\quad + [\Gamma_0 \Lambda_0 T + (\Gamma_1 + \Gamma_0)(\Lambda_1 + \Lambda_0)NT^2]Z, \end{aligned} \quad (1)$$

which is illustrated in Figure 3.

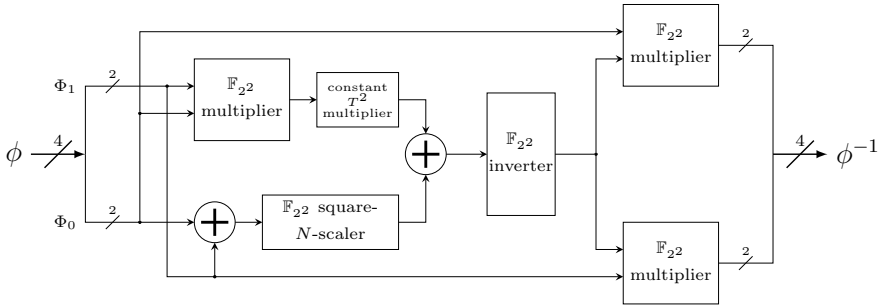
Figure 3 The \mathbb{F}_{2^4} multiplier



Let $\phi = \Phi_1 Z^4 + \Phi_0 Z$ with $\Phi_i \in \mathbb{F}_{2^2}$. It can be shown that the inverse ϕ^{-1} of ϕ is

$$[\Phi_1 \Phi_0 T^2 + (\Phi_1 + \Phi_0)^2 N]^{-1} \Phi_0 Z^4 + [\Phi_1 \Phi_0 T^2 + (\Phi_1 + \Phi_0)^2 N]^{-1} \Phi_1 Z, \quad (2)$$

whose circuit is depicted in Figure 4.

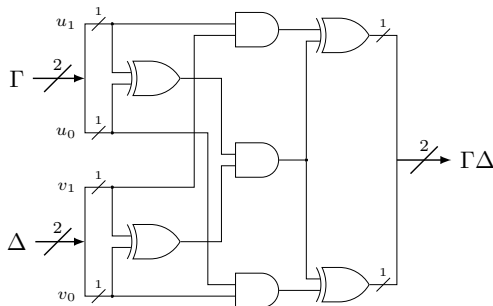
Figure 4 The \mathbb{F}_{2^4} inverter


3.2 Multiplication and inverse over \mathbb{F}_{2^2} .

Consider the field extension $\mathbb{F}_2 \subseteq \mathbb{F}_{2^2}$ with irreducible polynomial $t(w) = w^2 + w + 1 \in \mathbb{F}_2[w]$ (the only irreducible polynomial in $\mathbb{F}_2[w]$). Let W be a root of $t(w)$ over \mathbb{F}_2 . Then every element $\Gamma \in \mathbb{F}_{2^2}$ can be represented as an \mathbb{F}_2 -linear combination $\Gamma = u_1W^2 + u_0W$ of the normal basis $[W^2, W]$, with $u_i \in \mathbb{F}_2$. Let $\Delta = v_1W^2 + v_0W$ with $v_j \in \mathbb{F}_2$ be another element in \mathbb{F}_{2^2} . The multiplication is given by

$$\begin{aligned} \Gamma\Delta &= (u_1W^2 + u_0W)(v_1W^2 + v_0W) \\ &= [u_1 \cdot v_1 \oplus (u_1 \oplus u_0) \cdot (v_1 \oplus v_0)]W^2 \\ &\quad + [u_0 \cdot v_0 \oplus (u_1 \oplus u_0) \cdot (v_1 \oplus v_0)]W, \end{aligned} \quad (3)$$

whose implementation is shown in Figure 5. In addition, if $\Gamma\Delta = 1$, it can be shown that $v_1 = u_0$ and $v_0 = u_1$. That is, the \mathbb{F}_{2^2} inverter can be implemented by swapping the two 1-bit input signals, which is free.

Figure 5 The \mathbb{F}_{2^2} multiplier


Remark: Finally, we would like to mention two other formulas which are useful later:

$$\begin{aligned} \Gamma\Delta \cdot W &= (u_1 \cdot v_1 \oplus u_0 \cdot v_0)W^2 + [u_1 \cdot v_1 \oplus (u_1 \oplus u_0) \cdot (v_1 \oplus v_0)]W \\ \Gamma\Delta \cdot W^2 &= [u_0 \cdot v_0 \oplus (u_1 \oplus u_0) \cdot (v_1 \oplus v_0)]W^2 + (u_1 \cdot v_1 \oplus u_0 \cdot v_0)W. \end{aligned} \quad (4)$$

According to equation (4), the implementation cost of a multiplication followed with a W (or W^2) scaler is the same as that of the multiplication $\Gamma\Delta$, which requires four XOR gates and three AND gates.

4 Applications to the S-boxes of AES, Camellia, and SM4

The S-boxes of AES, Camellia, and SM4 are all affine equivalent to the \mathbb{F}_{2^8} inverter, which can be unified into the following form

$$S(b) = M_2 \cdot I_{\mathcal{P}\mathcal{B}}^q(M_1 \cdot b \oplus C_1) \oplus C_2, \quad b \in \mathbb{F}_{2^8},$$

where M_1, M_2 are 8×8 matrices over \mathbb{F}_2 , C_1, C_2 are constant column vectors in \mathbb{F}_2^8 , and $I_{\mathcal{P}\mathcal{B}}^q: \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ is a function that maps the bit-vector representation of an element in \mathbb{F}_{2^8} to the representation of its inverse in \mathbb{F}_{2^8} under a polynomial basis of \mathbb{F}_{2^8} over \mathbb{F}_2 induced by an irreducible polynomial $q(x) \in \mathbb{F}_2[x]$. We refer reader to Table 4 for the concrete values of these parameters for AES, Camellia and SM4.

However, it is difficult to implement the function $I_{\mathcal{P}\mathcal{B}}^q$ directly with small circuit footprint. Therefore, we first implement the function $I_{\mathcal{T}\mathcal{B}}: \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ which maps the representation of an element in \mathbb{F}_{2^8} to the representation of its inverse element under the tower basis $\mathcal{T}\mathcal{B}$. According to the discussion of Section 2, we have

$$I_{\mathcal{P}\mathcal{B}}^q(b) = M_t \cdot I_{\mathcal{T}\mathcal{B}}(M_t^{-1} \cdot b).$$

Therefore, $S(b)$ can be implemented in practice as

$$S(b) = M_2 M_t \cdot I_{\mathcal{T}\mathcal{B}}(M_t^{-1} M_1 \cdot b \oplus M_t^{-1} C_1) \oplus C_2, \quad b \in \mathbb{F}_2^8. \quad (5)$$

Our goal is to identify a proper tower basis such that the overall circuit footprint of the implementation of $S(b)$ is minimised. Recalling the tower field architecture shown in Figure 1, the tower basis is completely determined by the three irreducible polynomials $r(y) = y^2 + \tau y + \nu \in \mathbb{F}_{2^4}[y]$, $s(z) = z^2 + Tz + N \in \mathbb{F}_{2^2}[z]$, $t(w) = w^2 + w + 1 \in \mathbb{F}_2[w]$, and their roots Y, Z and W . Therefore, the 2^{15} possible choices of τ, ν, T, N, Y, Z , and W form the overall design space¹, in which there are only 720 valid cases (we discard equivalent classes and non-irreducible polynomials).

Concretely, there are six possibilities for (T, N) making $s(z)$ irreducible, and they are $\{(1, W), (1, W^2), (W, 1), (W, W), (W^2, 1), (W^2, W^2)\}$. For each possible choice of (T, N) , 120 cases of (τ, ν) can be identified such that $r(y)$ is irreducible. We can choose either one of the two roots for W, Z and Y because the other roots are exactly W^2, Z^4 and Y^{16} . So altogether there are $6 \times 120 \times 1 \times 1 \times 1 = 720$ valid cases. We exhaust all these cases for AES, Camellia and SM4 and list the optimal parameter choices in terms of compactness in Table 5.

According to Table 5, for the parameters of AES, we have the following relationship:

$$T = W, N = 1, \tau = T \cdot Z^4 + T \cdot Z, \nu = 0 \cdot Z^4 + T^2 \cdot Z. \quad (6)$$

Similarly, for Camellia, we have $T = W, N = 1, \tau = T^2 \cdot Z^4 + T^2 \cdot Z, \nu = T \cdot Z^4 + 0 \cdot Z$, and for SM4, we have $T = W, N = 1, \tau = T^2 \cdot Z^4 + 1 \cdot Z, \nu = T \cdot Z^4 + T^2 \cdot Z$. In what follows, we focus on the optimisation of the AES S-box with the optimal parameter we identified as an example. For Camellia, SM4 and other parameters, the same procedure is performed.

Table 4 The parameters of the S-boxes of AES, Camellia, and SM4, where a hexadecimal number represents an irreducible polynomial in $\mathbb{F}_2[x]$ (e.g., $x^8 + x^4 + x^3 + x + 1$ is represented by 0x11B)

<i>Cipher</i>	M_1	C_1	M_2	C_2	$q(x)$
AES	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$	0x11B
Camellia*	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	0x169
SM4	$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$	0x1F5

Notes: *the description of the Camellia S-box in the original specification (Aoki et al., 2000) is different from ours. Reader could check the substitution table to confirm the equivalence.

Table 5 Optimal choices of parameters for AES, Camellia and SM4 in terms of compactness

<i>Cipher</i>	W	T	N	Z	τ	ν	Y
AES	0xBC	0xBC	0x01	0xB0	0xBD	0x5C	0xF4
Camellia	0x7E	0x7E	0x01	0x15	0x01	0x06	0x02
SM4	0x5C	0x5C	0x01	0x7A	0x77	0x27	0x66

Note: The parameters are given with their polynomial basis representations (e.g., $X^7 + X^5 + X^4 + X^3 + X^2$ is represented by 0xBC).

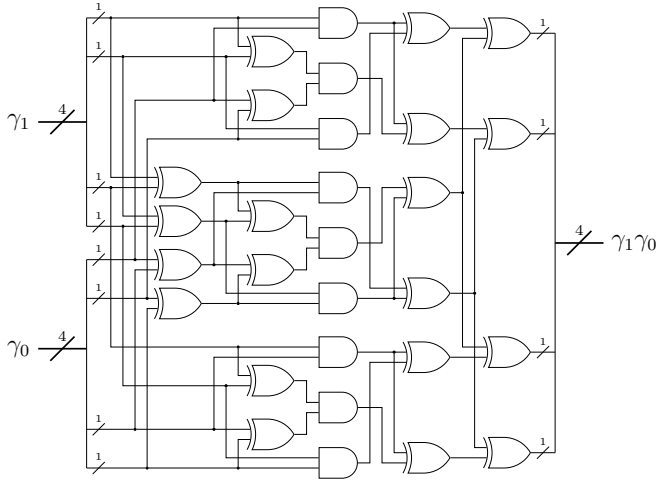
4.1 Optimised implementation of the \mathbb{F}_{2^4} multiplier, τ^2 multiplier, and the square- ν -scaler as a whole

Before giving the optimised implementation, we unfold the circuits of the \mathbb{F}_{2^4} multiplier, τ^2 multiplier, and square- ν -scaler one by one without any optimisation. Based on

these unfolded circuits, we reduce their areas by applying several logic minimisation techniques with necessary tweaks.

\mathbb{F}_{2^4} multiplier: by plugging Figure 5 into Figure 3, we obtain the gate-level circuit of the \mathbb{F}_{2^4} multiplier shown in Figure 6, which can also be derived by substituting equations (3) and (4) into equation (1).

Figure 6 \mathbb{F}_{2^4} multiplier

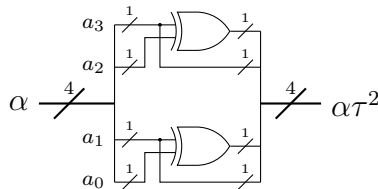


τ^2 multiplier: according to Table 5 and equation (6), we have $\tau = TZ^4 + TZ = WZ^4 + WZ$ and $\tau^2 = Z^4 + Z$. Let $\alpha = (a_3W^2 + a_2W)Z^4 + (a_1W^2 + a_0W)Z$ be an arbitrary element in \mathbb{F}_{2^4} . We have

$$\alpha\tau^2 = [(a_3 \oplus a_2)W^2 + a_3W]Z^4 + [(a_1 \oplus a_0)W^2 + a_1W]Z,$$

leading to the gate-level circuit of the τ^2 multiplier shown in Figure 7.

Figure 7 τ^2 multiplier



\mathbb{F}_{2^4} square- ν -scaler: according to Table 5 and equation (6), $\nu = W^2Z$. Let $\alpha = (a_3W^2 + a_2W)Z^4 + (a_1W^2 + a_0W)Z$ be an arbitrary element in \mathbb{F}_{2^4} . Then $\alpha^2\nu$ can be computed as $\alpha^2\nu = [(a_3 + a_1)W^2 + (a_3 + a_2 + a_1 + a_0)W]Z^4 + [(a_1 + a_0)W^2 + a_0W]Z$, whose gate-level circuit is shown in Figure 8.

Figure 8 \mathbb{F}_{2^4} square- ν -scaler

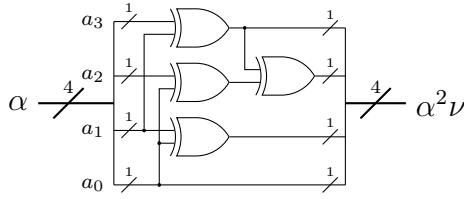
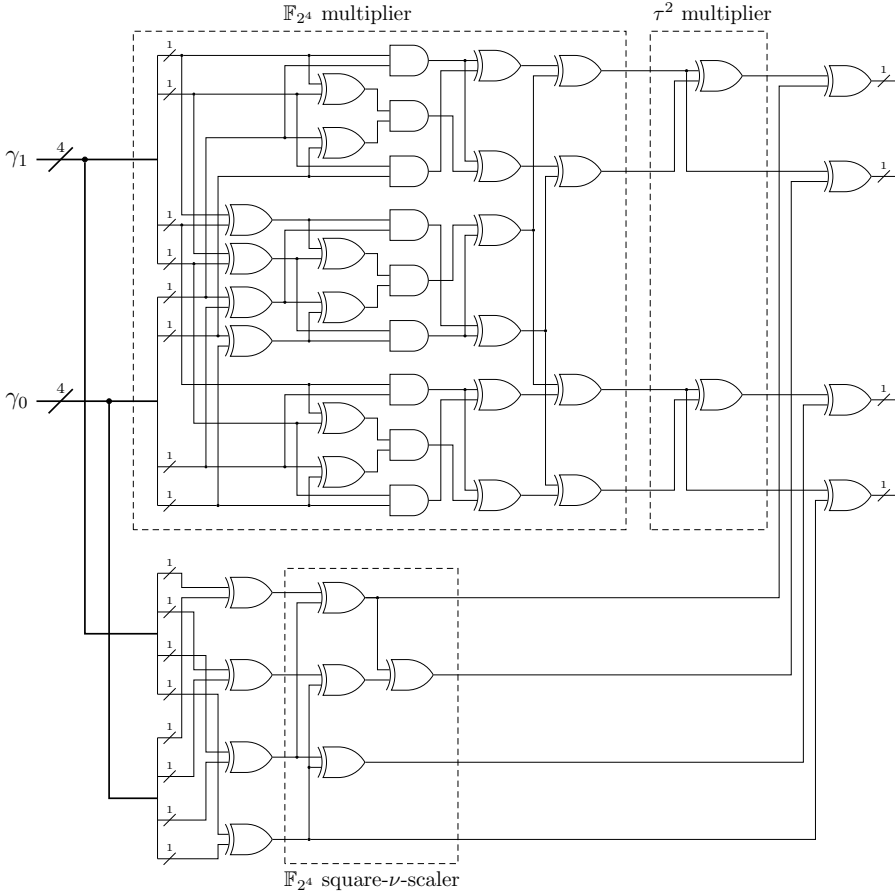


Figure 9 A part of the \mathbb{F}_{2^8} inverter (unoptimised)



Now, we can assemble the \mathbb{F}_{2^4} multiplier, τ^2 multiplier, and square- ν -scaler to obtain a part of the \mathbb{F}_{2^8} inverter according to Figure 2, which gives the circuit shown in Figure 9. According to equation (4), this circuit can be partially optimised with some tweaks on the eight XOR gates appearing at the lower part of Figure 9, leading to the circuit shown in Figure 10. Subsequently, by applying the formula

$$a \cdot b \oplus a \oplus b = a \mid b \tag{7}$$

Figure 10 A part of the \mathbb{F}_{2^8} inverter (partially optimised)

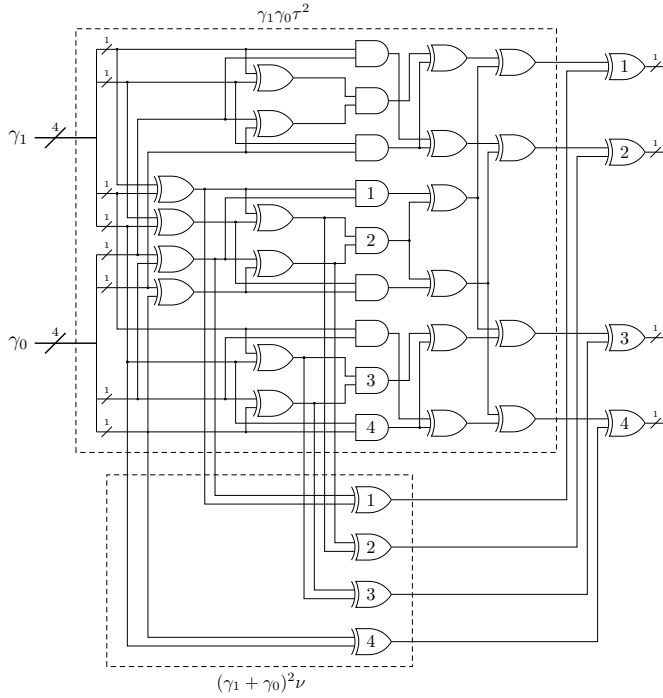
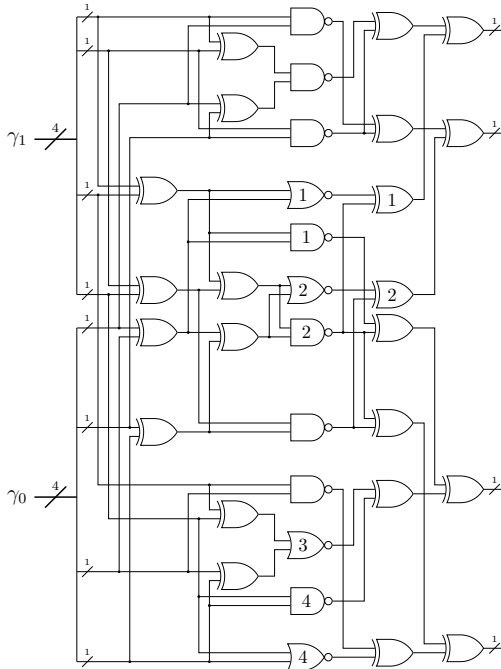


Figure 11 A part of the \mathbb{F}_{2^8} inverter (optimised)



We can transform the circuit shown in Figure 10 into the circuit presented in Figure 11. According to equation (7), at best, we can replace two XOR gates and one AND gate by a single OR gate. This happens for the gates marked by number 3. However, when some intermediate value of the computation $a \cdot b \oplus a \oplus b$ is required, we still need to keep some intermediate gates. For example, we can only replace two XOR gates with one OR gate and keep the AND gate intact. Similarly, for the gates marked with number 1 and number 2, we can only replace one XOR gates with one OR gate. Finally, by applying the formulas $a \cdot b \oplus c \cdot d = \overline{a \cdot b \oplus c \cdot d}$ and $a \cdot b \oplus c \mid d = \overline{a \cdot b \oplus c \mid d}$, the AND gates and OR gates can be substituted by NAND gates and NOR gates respectively.

4.2 Optimised implementation of the \mathbb{F}_{2^4} inverter

Based on the selected parameters for AES ($T = W$ and $N = 1$) given in Table 5 and equation (6), equation (2) can be simplified as

$$\phi^{-1} = [\Phi_1 \Phi_0 W^2 + (\Phi_1 + \Phi_0)^2]^{-1} \Phi_0 Z^4 + [\Phi_1 \Phi_0 W^2 + (\Phi_1 + \Phi_0)^2]^{-1} \Phi_1 Z. \quad (8)$$

Deviating from previous implementations (Boyar et al., 2013; Canright, 2005a, 2005b), we regard the \mathbb{F}_{2^4} inverter as a 4×4 S-box whose permutation table is determined by equation (8):

$$[0x0, 0x8, 0x4, 0xC, 0x2, 0xF, 0x7, 0x6, 0x1, 0xD, 0xA, 0xE, 0x3, 0x9, 0xB, 0x5].$$

To obtain optimised implementations of this S-box, we consider two recently proposed techniques. First, we employ Stoffelen's SAT-based technique (Stoffelen, 2016) to produce a circuit of the 4×4 S-box: $(x_3, x_2, x_1, x_0) \mapsto (y_3, y_2, y_1, y_0)$ with minimised gate complexity, and the result is shown below:

$$\begin{array}{lll} t_1 = \overline{x_3 \cdot x_0} & t_2 = t_1 \mid x_2 & t_3 = \overline{x_2 \cdot x_0} \\ t_4 = x_1 \oplus t_3 & t_5 = \overline{x_2 \mid t_4} & t_6 = \overline{x_1 \cdot t_4} \\ t_7 = x_3 \mid t_4 & t_8 = t_7 \cdot t_2 & t_9 = t_5 \oplus t_7 \\ t_{10} = t_9 \odot x_3 \quad (y_0) & t_{11} = \overline{t_6 \cdot t_8} \quad (y_2) & t_{12} = \overline{t_8 \cdot x_1} \\ t_{13} = x_0 \odot t_{12} \quad (y_3) & t_{14} = \overline{t_1 \cdot x_2} & t_{15} = t_9 \cdot t_{14} \quad (y_1), \end{array}$$

which contains four XOR/XNOR gates, 1 AND gate, seven NAND gates, two OR gates and one NOR gate². This circuit (referred as SAT) can be further optimised manually. Since $a \cdot b = \overline{\overline{a} \mid \overline{b}}$, we can change the AND gate in t_8 to NOR gate, and negate its input signals without changing the overall functionality of the circuit. This new circuit (referred as SAT*) contains four XOR/XNOR gates, seven NAND gates and four NOR gates (modified signals are coloured in red):

$$\begin{array}{lll} t_1 = \overline{x_3 \cdot x_0} & \mathbf{t_2} = \overline{\overline{t_1} \mid x_2} & t_3 = \overline{x_2 \cdot x_0} \\ t_4 = x_1 \oplus t_3 & t_5 = \overline{x_2 \mid t_4} & t_6 = \overline{x_1 \cdot t_4} \\ \mathbf{t_7} = \overline{x_3 \mid t_4} & \mathbf{t_8} = \overline{t_7 \mid t_2} & \mathbf{t_9} = t_5 \odot t_7 \\ t_{10} = t_9 \odot x_3 \quad (y_0) & t_{11} = \overline{t_6 \cdot t_8} \quad (y_2) & t_{12} = \overline{t_8 \cdot x_1} \\ t_{13} = x_0 \odot t_{12} \quad (y_3) & t_{14} = \overline{t_1 \cdot x_2} & t_{15} = t_9 \cdot t_{14} \quad (y_1). \end{array}$$

We also apply the LIGHTER (Jean et al., 2017b) tool to the 4×4 S-box (the \mathbb{F}_{2^4} inverter) for four different technology libraries, which leads to the same circuit (referred as LIGHTER) containing seven XOR/XNOR gates, four NAND gates, one NAND3 gate, one NOR gate and one NOT gate shown in the following:

$$\begin{aligned}
 t_1 &= x_2 \oplus x_3 & t_2 &= \overline{t_1 \cdot x_0 \cdot x_3} & t_3 &= x_1 \odot t_2 \\
 t_4 &= \overline{t_3 \cdot t_1} & t_5 &= x_0 \oplus t_4 & t_6 &= \overline{x_3 \cdot t_5} \\
 t_7 &= t_1 \oplus t_6 & t_8 &= \overline{t_5 \mid t_7} & t_9 &= x_3 \oplus t_8 \quad (y_0) \\
 t_{10} &= \overline{t_9 \cdot t_3} & t_{11} &= t_5 \oplus t_{10} \quad (y_3) & t_{12} &= \overline{t_7} \quad (y_1) \\
 t_{13} &= \overline{t_{11} \cdot t_{12}} & t_{14} &= t_3 \odot t_{13} \quad (y_2).
 \end{aligned}$$

A comparison of the above three circuits together with their synthesising results is given in Tables 6 and 7, from which we can see that SAT* is always the best, whose circuit is depicted in Figure 12.

Figure 12 The optimised circuit for the \mathbb{F}_{2^4} inverter (SAT*)

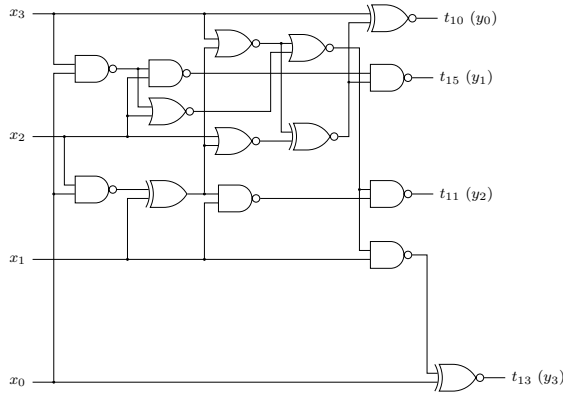
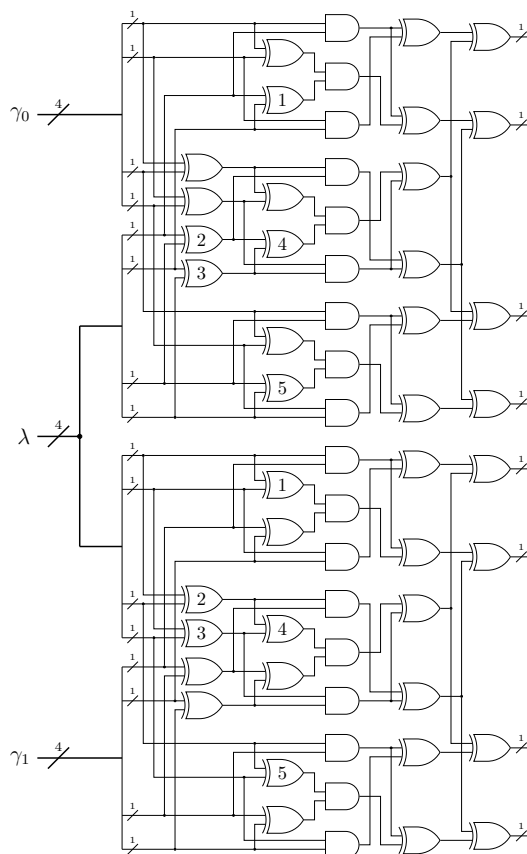


Table 6 Gate counts for different implementations of \mathbb{F}_{2^4} inverter, where the circuit named Canright is the implementation of equation (8) using the method in Canright (2005a, 2005b), and the circuit named Boyar uses the method in Boyar et al. (2013)

Circuit	Gates used						
	XOR/XNOR	AND	NAND	NAND3	OR	NOR	NOT
Canright	9		8			2	
Boyar	9	6					
Boyar*	9		6				
SAT	4	1	7		2	1	
SAT*	4		7			4	
LIGHTER	7		4	1		1	1

Table 7 Synthesised results for different implementations of the \mathbb{F}_{2^4} inverter

Circuit	Synthesis results			
	SMIC 130 nm	SMIC 65 nm	STM 65 nm	Nangate 45 nm
Canright	30.97	30.25	28.00	28.00
Boyar	28.95	29.25	25.50	25.98
Boyar*	26.97	26.25	24.00	24.00
SAT	21.31	21.50	20.25	19.99
SAT*	20.32	20.00	19.00	19.00
LIGHTER	23.30	22.75	21.00	20.99

Figure 13 The two rightmost \mathbb{F}_{2^4} multipliers with a shared 4-bit input


4.3 Optimised implementation of the two \mathbb{F}_{2^4} multipliers with 4-bit common input

Observing Figure 2, the \mathbb{F}_{2^8} inverter contains three \mathbb{F}_{2^4} multipliers, from which we can identify three pairs of \mathbb{F}_{2^4} multipliers such that each pair shares a 4-bit input signal:

the leftmost \mathbb{F}_{2^4} multiplier and the rightmost upper \mathbb{F}_{2^4} multiplier, the leftmost \mathbb{F}_{2^4} multiplier and the rightmost lower \mathbb{F}_{2^4} multiplier, and the two rightmost \mathbb{F}_{2^4} multipliers. It is shown in Canright (2005a, 2005b) that whenever two \mathbb{F}_{2^4} multipliers share a common 4-bit input signal, some XOR gates can be saved via signal reuse.

As an example, we unfold the two rightmost \mathbb{F}_{2^4} multipliers in Figure 2 according to Figure 6, and the schematic is shown in Figure 13. By observing Figure 13 carefully, we can spot some outputs of XOR gates which are computed twice in the circuit (Canright, 2005a, 2005b) (labelled with same numbers in the figure). Therefore, for each pair of \mathbb{F}_{2^4} multipliers sharing a 4-bit input signal, we can remove five XOR gates by signal reuse. Therefore, three pairs of \mathbb{F}_{2^4} multipliers with shared input signals in total save $5 \times 3 = 15$ XOR gates.

4.4 Optimised implementation of the input and output affine parts

According to equation (5), before going into the \mathbb{F}_{2^8} inverter $I_{TB}(\cdot)$, the 8-bit input signal of the AES S-box first goes through an affine transformation

$$b \mapsto g = M_t^{-1}M_1 \cdot b \oplus M_t^{-1}C_1,$$

which then spawns 18 1-bit signals (see Figure 11) subsequently fed into some nonlinear gates (NAND, NOR). The transformation from the eight 1-bit input signals to the 18 1-bit signals is affine, and can be represented as an 18×8 matrix

$$U = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T,$$

with the constant vector expanded from $M_t^{-1}C_1$

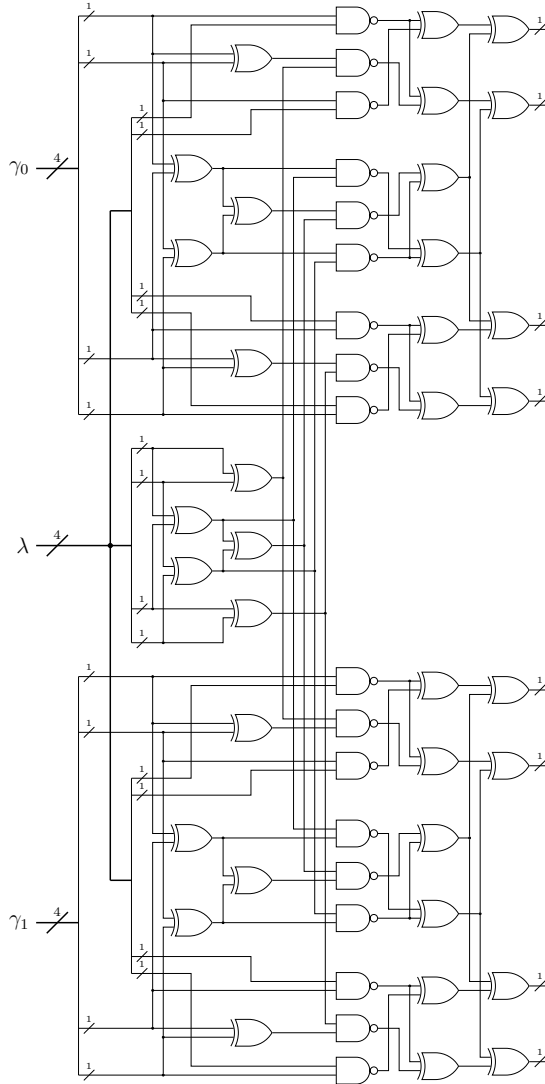
$$C = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)^T.$$

By applying the SAT-based method for solving shortest linear straight-line program (SLP) (Fuhs and Schneider-Kamp, 2010), we obtain the optimal implementation of U , which costs 19 XOR gates³:

$$\begin{array}{lll} y_{17} = x_0 & (y_{17}) & t_1 = x_7 \oplus x_4 & (y_6) & t_2 = x_3 \oplus x_1 \\ t_3 = x_2 \oplus t_2 & & t_4 = x_6 \oplus t_3 & (y_7) & t_5 = x_0 \oplus t_4 & (y_{15}) \\ t_6 = x_5 \oplus t_3 & (y_1) & t_7 = t_5 \oplus t_6 & (y_{14}) & t_8 = x_4 \oplus t_7 & (y_{13}) \\ t_9 = t_1 \oplus t_2 & (y_9) & t_{10} = t_1 \oplus t_8 & (y_{11}) & t_{11} = x_0 \oplus t_9 & (y_{16}) \\ t_{12} = x_4 \oplus x_2 & (y_2) & t_{13} = x_1 \oplus t_7 & (y_{10}) & t_{14} = x_7 \oplus x_2 & (y_4) \\ t_{15} = t_{13} \oplus t_{14} & (y_{12}) & t_{16} = x_7 \oplus x_1 & (y_0) & t_{17} = t_{12} \oplus t_{16} & (y_8) \\ t_{18} = t_6 \oplus t_9 & (y_3) & t_{19} = t_7 \oplus t_{11} & (y_5) & & \end{array}$$

where x_i 's are the input signals, t_i 's are intermediate signals, and y_i 's are output signals.

Figure 14 The circuit for bottom part



Similarly, according to equation (5), at the output end of the AES S-box, the 8-bit output of the two rightmost \mathbb{F}_{2^4} multipliers (see Figures 2 and 14) is transformed by the affine mapping $M_2M_t(\cdot) \oplus C_2$ to recover the polynomial basis representation. Observing Figure 14, the eight input bits of the affine mapping (also the output bits of the two \mathbb{F}_{2^4} multipliers) are originated from the 18 output bits of the NAND gates. Moreover, only XOR gates are involved to generate the eight input bits of the affine

mapping from these 18 bits. Therefore, the mapping from the 18 output bits of the NAND gates to the eight output bits of the S-box is affine, which can be described by the following matrix

$$B = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Observing Figure 14, there are two layers of XOR gates following the 18 NAND gates. According these two layers of XOR gates, we decompose the whole output affine transformation (from the 18 output bits of the 18 NAND gates to the eight output bits of the AES S-box) into two parts. The first part maps the 18 output bits of the 18 NAND gates to the 12 output bits of the first layer of 12 XOR gates, which can be implemented with in total 12 XOR gates as shown in Figure 14. The second part maps the 12 output bits of the 12 XOR gates to the eight output bits of the S-box, and its matrix representation B' is given in the following:

$$B' = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

Again, by applying the SAT-based method for SLP (Fuhs and Schneider-Kamp, 2010) and taking C_2 into account, the affine transformation involving B' and constant addition at the output end can be realised as follows, requiring 17 XOR/XNOR gates⁴:

$$\begin{aligned} t_1 &= x_2 \oplus x_0 & t_2 &= x_{10} \oplus t_1 & t_3 &= x_8 \oplus t_2 & (y_7) \\ t_4 &= x_5 \oplus x_1 & t_5 &= x_5 \oplus x_3 & t_6 &= x_7 \oplus t_5 \\ t_7 &= x_8 \oplus x_6 & t_8 &= x_2 \oplus t_3 & t_9 &= x_4 \oplus t_8 & (y_4) \\ t_{10} &= t_1 \oplus t_5 & t_{11} &= x_9 \odot t_6 & (y_5) & t_{12} &= t_4 \odot t_7 & (y_0) \\ t_{13} &= t_3 \oplus t_6 & t_{14} &= x_{11} \oplus t_{13} & (y_2) & t_{15} &= t_1 \odot t_9 & (y_6) \\ t_{16} &= t_{10} \oplus t_{12} & (y_1) & t_{17} &= t_4 \oplus t_9 & (y_3) \end{aligned}$$

where x_i 's are the input signals, t_i 's are intermediate signals, and y_i 's are output signals.

4.5 Overall implementation results and comparison

We synthesis the optimised implementations of the S-boxes (AES, Camellia, SM4) using Synopsys Design Compiler 2014 (DC 2014) with four technology libraries, and the

synthesised results⁵ together with their technology-independent gate counts are listed in Table 2.⁶

To make the full use of the libraries to save the circuit area, Reyhani-Masoleh et al.'s implementations (Reyhani-Masoleh et al., 2018) exploit certain compound gates in specific libraries (e.g., XOR3, NAND3, OAI21, AOI21, OAI32), which are not universally available in all technology libraries. For example, the optimal implementation offered by Reyhani-Masoleh et al. (2018) employs XOR3 and OAI32 gates, reaching 182.25 GE under the STM 65 nm CMOS technology.

According to the results shown in Table 2, our implementation requires only 179 GE, which beats the record set by Reyhani-Masoleh et al. (2018) even without using any compound gates. Moreover, when the area of one XOR3 gate is smaller than the area of two XOR gates in underlying technology library, the compound gates XOR3 can be applied in our design to take the place of some standard XOR gates. With this improvements, the area of our implementation of the AES S-box can be further reduced to 176.75 GE. For the S-boxes of Camellia and SM4, it can be seen from Table 2 that the improvements are even more obvious.

5 Conclusions

By applying state-of-the-art combinatorial logic minimisation techniques to an exhaustive list of tower field representations of the AES, Camellia, and SM4 S-boxes with normal bases, we identify so far the most compact implementations of these S-boxes. The results obtained in this work can be used in compact and threshold implementations of AES, Camellia, and SM4. As a potential further work, it is interesting to see how to apply similar techniques to obtain compact implementations of combined S-box/inverse S-box designs for AES, Camellia, and SM4. Moreover, this work only focus on minimising the circuit area, it is of equal importance to investigate how to reduce the depth of the circuit as in Li et al. (2019) and Reyhani-Masoleh et al. (2018).

Appendices/Supplementary material are available on request by emailing the corresponding author or can be obtained under <https://github.com/zihaowei/Tower-Field-Implementation>.

Acknowledgements

The work is partially supported by the National Key R&D Program of China (Grant No. 2018YFA0704704), the Chinese Major Program of National Cryptography Development Foundation (Grant No. MMJJ20180102), the National Natural Science Foundation of China (61772519, 61732021, 61802400, 61802399), and the Youth Innovation Promotion Association of Chinese Academy of Sciences.

References

- Abbasi, I. and Afzal, M. (2011) ‘A compact S-Box design for SMS4 block cipher’, *IT Convergence and Services*, Vol. 107, pp.641–658.
- Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J. and Tokita, T. (2000) ‘Camellia: a 128-bit block cipher suitable for multiple platforms – design and analysis’, *Selected Areas in Cryptography*, Vol. 2012, pp.39–56.
- Bai, X., Xu, Y. and Guo, L. (2009) ‘Securing SMS4 cipher against differential power analysis and its VLSI implementation’, *IEEE Singapore International Conference on Communication Systems*, pp.167–172.
- Banik, S., Bogdanov, A. and Minematsu, K. (2016a) ‘Low-area hardware implementations of CLOC, SILC and AES-OTR’, *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016*, 3–5 May, McLean, VA, USA, pp.71–74.
- Banik, S., Bogdanov, A. and Regazzoni, F. (2016b) ‘Atomic-AES: a compact implementation of the AES encryption/decryption core’, *Progress in Cryptology – INDOCRYPT 2016, 17th International Conference on Cryptology in India, Proceedings*, 11–14 December, Kolkata, India, pp.173–190.
- Beierle, C., Kranz, T. and Leander, G. (2016) ‘Lightweight multiplication in $GF(2^n)$ with applications to MDS matrices’, *Advances in Cryptology – CRYPTO 2016, 36th Annual International Cryptology Conference, Proceedings, Part I*, 14–18 August, Santa Barbara, CA, USA, pp.625–653.
- Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V. and Rijmen, V. (2014) ‘A more efficient AES threshold implementation’, *Progress in Cryptology - AFRICACRYPT 2014, 7th International Conference on Cryptology in Africa, Proceedings*, 28–30 May, Marrakesh, Morocco, pp.267–284.
- Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V. and Rijmen, V. (2015) ‘Trade-offs for threshold implementations illustrated on AES’, *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 34, No. 7, pp.1188–1200.
- Boyar, J., Matthews, P. and Peralta, R. (2013) ‘Logic minimization techniques with applications to cryptology’, *J. Cryptology*, Vol. 26, No. 2, pp.280–312.
- Canright, D. (2005a) *A very compact Rijndael S-box*, Technical Report NPS-MA-05-001, Naval Postgraduate School [online] https://www.researchgate.net/publication/235155631_A_Very_Compact_Rijndael_S-box (accessed 12 July 2019).
- Canright, D. (2005b) ‘A very compact S-Box for AES’, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Proceedings*, 29 August–1 September, Edinburgh, UK, pp.441–455.
- Circuit Minimization Team (CMT) [online] <http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html> (accessed 12 July 2019).
- Cnudde, T. D., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V. and Rijmen, V. (2016) ‘Masking AES with $d + 1$ shares in hardware’, *Cryptographic Hardware and Embedded Systems – CHES 2016, 18th International Conference, Proceedings*, 17–19 August, Santa Barbara, CA, USA, 2016, pp.194–212.
- Daemen, J. and Rijmen, V. (2002) *The Design of Rijndael: AES – The Advanced Encryption Standard*, Information Security and Cryptography, Springer.
- Fuhs, C. and Schneider-Kamp, P. (2010) ‘Synthesizing shortest linear straight-line programs over $GF(2)$ using SAT’, *Theory and Applications of Satisfiability Testing – SAT 2010, 13th International Conference, SAT 2010, Proceedings*, 11–14 July, Edinburgh, UK, pp.71–84.
- Guajardo, J. and Paar, C. (1997) ‘Efficient algorithms for elliptic curve cryptosystems’, *Advances in Cryptology – CRYPTO ’97, 17th Annual International Cryptology Conference, Proceedings*, 17–21 August, Santa Barbara, California, USA, pp.342–356.

- Itoh, T. and Tsujii, S. (1988) ‘A fast algorithm for computing multiplicative inverses in $\text{GF}(2^m)$ using normal bases’, *Inf. Comput.* Vol. 78, No. 3, pp.171–177.
- Jean, J., Moradi, A., Peyrin, T. and Sasdrich, P. (2017a) ‘Bit-sliding: a generic technique for bit-serial implementations of SPN-based primitives – applications to AES, PRESENT and SKINNY’, *Cryptographic Hardware and Embedded Systems – CHES 2017, 19th International Conference, Proceedings*, 25–28 September, Taipei, Taiwan, pp.687–707.
- Jean, J., Peyrin, T., Sim, S.M. and Tourteaux, J. (2017b) ‘Optimizing implementations of lightweight building blocks’, *IACR Trans. Symmetric Cryptol.* Vol. 2017, No. 4, pp.130–168.
- Kranz, T., Leander, G., Stoffelen, K. and Wiemer, F. (2017) ‘Shorter linear straight-line programs for MDS matrices’, *IACR Trans. Symmetric Cryptol.*, Vol. 2017, No. 4, pp.188–211.
- Li, S., Sun, S., Li, C., Wei, Z. and Hu, L. (2019) ‘Constructing low-latency involutory MDS matrices with lightweight circuits’, *IACR Trans. Symmetric Cryptol.* Vol. 2019, No. 1, pp.84–117.
- Li, S., Sun, S., Shi, D., Li, C. and Hu, L. (2019) ‘Lightweight iterative MDS matrices: how small can we go?’, *IACR Trans. Symmetric Cryptol.*, Vol. 2019, No. 4, pp.147–170.
- Tan, Q. and Peyrin, T. (2020) ‘Improved heuristics for short linear programs’, *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, Vol. 2020, No. 1, pp.203–230.
- Martínez-Herrera, A.F., Mex-Perera, J.C. and Nolasco-Flores, J.A. (2012) ‘Some representations of the S-Box of Camellia in $\text{GF}(((2^2)^2)^2)$ ’, *Cryptography and Network Security, 11th International Conference, CANS 2012, Proceedings*, 12–14 December, Darmstadt, Germany, pp.296–309.
- Martínez-Herrera, A.F., Mex-Perera, J.C. and Nolasco-Flores, J.A. (2013) ‘Merging the Camellia, SMS4 and AES S-Boxes in a single S-Box with composite bases’, *Information Security, 16th International Conference, ISC 2013, Proceedings*, 13–15 November, Dallas, Texas, USA, pp.209–217.
- Mentens, N., Batina, L., Preneel, B. and Verbauwhede, I. (2005) ‘A systematic evaluation of compact hardware implementations for the Rijndael S-Box’, *Topics in Cryptology – CT-RSA 2005, The Cryptographers’ Track at the RSA Conference 2005, Proceedings*, 14–18 February, San Francisco, CA, USA, pp.323–333.
- Moradi, A., Poschmann, A., Ling, S., Paar, C. and Wang, H. (2011) ‘Pushing the limits: a very compact and a threshold implementation of AES’, *Advances in Cryptology – EUROCRYPT 2011, 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, 15–19 May, Tallinn, Estonia, pp.69–88.
- Paar, C. and Soria-Rodriguez, P. (1997) ‘Fast arithmetic architectures for public-key algorithms over Galois Fields $\text{GF}((2^n)^m)$ ’, *Advances in Cryptology – EUROCRYPT ‘97, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding*, 11–15 May, Konstanz, Germany, pp.363–378.
- Reyhani-Masoleh, A., Taha, M. M. I. and Ashmawy, D. (2018) ‘Smashing the implementation records of AES S-Box’, *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, Vol. 2018, No. 2, pp.298–336.
- Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R. and Rohatgi, P. (2001) ‘Efficient Rijndael encryption implementation with composite field arithmetic’, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Proceedings*, 14–16 May, Paris, France, pp.171–184.
- Satoh, A. and Morioka, S. (2003) ‘Unified hardware architecture for 128-bit block ciphers AES and Camellia’, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Proceedings*, 8–10 September, Cologne, Germany, pp.304–318.
- Satoh, A., Morioka, S., Takano, K. and Munetoh, S. (2001) ‘A compact Rijndael hardware architecture with S-Box optimization’, *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, 9–13 December, Gold Coast, Australia, pp.239–254.

- Stoffelen, K. (2016) ‘Optimizing S-Box implementations for several criteria using SAT solvers’, *Fast Software Encryption – FSE 2016, 23rd International Conference, Revised Selected Papers*, 20–23 March, Bochum, Germany, pp.140–160.
- Wolkerstorfer, J., Oswald, E. and Lamberger, M. (2002) ‘An ASIC implementation of the AES S-Boxes’, *Topics in Cryptology – CT-RSA 2002, The Cryptographer’s Track at the RSA Conference, Proceedings*, 18–22 February, San Jose, CA, USA, pp.67–78.

Notes

- 1 There are 2^2 choices for T , 2^2 choices for N , 2^4 choices for τ , 2^4 choices for ν , and 2 choices for each of W , Z and Y whenever T, N, τ, ν are fixed.
- 2 It costs about 38 minutes to obtain this 15-gate circuit on a PC. But we cannot confirm whether there is a 14-gate circuit, since we terminated the SAT solver after about two weeks’ computation.
- 3 It costs about 25 days on a PC to produce this result.
- 4 It costs about 30 days on a PC to produce this circuit.
- 5 We do not have access to the STM 65 nm technology library. However, Reyhani-Masoleh et al. (2018) provide sufficient area information for the gates involved in this particular library, based on which we can extrapolate the results without any difficulty.
- 6 Any mention of commercial products or reference to commercial organisations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.