# Towards Practical Whitebox Cryptography: Optimizing Efficiency and Space Hardness

Andrey Bogdanov[1(✉)], Takanori Isobe[2], and Elmar Tischhauser[1]

[1] Technical University of Denmark, Kongens Lyngby, Denmark
{anbog,ewti}@dtu.dk
[2] Sony Global Manufacturing & Operations Corporation, Tokyo , Japan
Takanori.Isobe@jp.sony.com

**Abstract.** Whitebox cryptography aims to provide security for cryptographic algorithms in an untrusted environment where the adversary has full access to their implementation. Typical security goals for whitebox cryptography include *key extraction security* and *decomposition security*: Indeed, it should be infeasible to recover the secret key from the implementation and it should be hard to decompose the implementation by finding a more compact representation without recovering the secret key, which mitigates code lifting.

Whereas all published whitebox implementations for standard cryptographic algorithms such as DES or AES are prone to practical key extraction attacks, there have been two dedicated design approaches for whitebox block ciphers: ASASA by Birykov et al. at ASIACRYPT'14 and SPACE by Bogdanov and Isobe at CCS'15. While ASASA suffers from decomposition attacks, SPACE reduces the security against key extraction and decomposition attacks in the white box to the security of a standard block cipher such as AES in the standard blackbox setting. However, due to the security-prioritized design strategy, SPACE imposes a sometimes prohibitive performance overhead in the real world as it needs many AES calls to encrypt a single block.

In this paper, we address the issue by designing a family of dedicated whitebox block ciphers SPNbox and a family of underlying small block ciphers with software efficiency and constant-time execution in mind. While still relying on the standard blackbox block cipher security for the resistance against key extraction and decomposition, SPNbox attains speed-ups of up to 6.5 times in the black box and up to 18 times in the white box on Intel Skylake and ARMv8 CPUs, compared to SPACE. The designs allow for constant-time implementations in the blackbox setting and meet the practical requirements to whitebox cryptography in real-world applications such as DRM or mobile payments. Moreover, we formalize resistance towards decomposition in form of *weak* and *strong space hardness* at various security levels. We obtain bounds on space hardness in all those adversarial models.

Thus, for the first time, SPNbox provides a practical whitebox block cipher that features well-understood key extraction security, rigorous


J.H. Cheon and T. Takagi (Eds.): ASIACRYPT 2016, Part I, LNCS 10031, pp. 126–158, 2016.
DOI: 10.1007/978-3-662-53887-6_5

analysis towards decomposition security, demonstrated real-world efficiency on various platforms and constant-time implementations. This paves the way to enhancing susceptible real-world applications with whitebox cryptography.

**Keywords:** White-box cryptography · Space hardness · Code lifting · Decomposition · Key extraction · Mass surveillance · Trojans · Malware

# 1 Introduction

## 1.1 Black Box vs White Box

Whitebox cryptography was introduced by Chow et al. in 2002 [14] as a technique to secure software implementations of block ciphers when the adversary has full access to the execution environment. This setup is called the *whitebox setting*, which is opposed to the standard *blackbox setting* where the attacker can neither observe nor influence the internals of the block cipher. The functionality of the cipher shall be the same when implemented in the black-box and white-boxe settings. However, the *whitebox implementation* in the untrusted environment (as e.g. in the mobile client software) and *blackbox implementation* in the secure environment (as e.g. on the backend server) can vary significantly to meet distinct security demands arising from two different threat models:

- **In the black box:** The adversary is able to access inputs and outputs of the cipher with known, chosen or adaptively chosen plaintexts/ciphertexts. Given the blackbox implementation, the attacker aims to recover the secret key (*key recovery*) or to distinguish the block cipher from a randomly drawn permutation (*distinguishing*).
- **In the white box:** The attacker has full access to the execution environment of the cipher. Given the whitebox implementation of the cipher, the adversary's goal is then to extract the secret key (*key extraction*) or to decompose the implementation to find a more compact representation that can be used as an effective key to replicate the functionality (*decomposition*, or *code lifting*).

## 1.2 Whitebox Cryptography in the Wild

The seminal papers [14,15] in whitebox cryptography had the goal to provide security in digital rights management (DRM) applications where encrypted contents (e.g. a music or movie file) are decrypted on the user's device. A malicious end user may attempt to extract the key from its software and then illegally distribute it outside the DRM system.

15 years have passed since those papers were published, and the context of whitebox cryptography has drastically changed. With the rapidly increasing demand for software-only security solutions in embedded devices, laptop PCs, mobile and server systems as well as the ever growing field of cloud-based services, the target for whitebox cryptography is no longer limited to the software

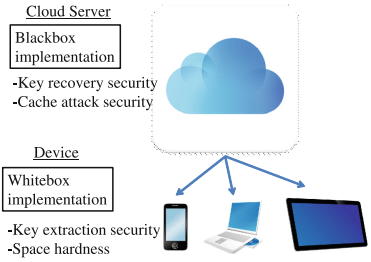


**Fig. 1.** Cloud-based content distribution: Cloud server encrypts contents in the black box and distributes them to user devices. User devices decrypt the contents in the white box.

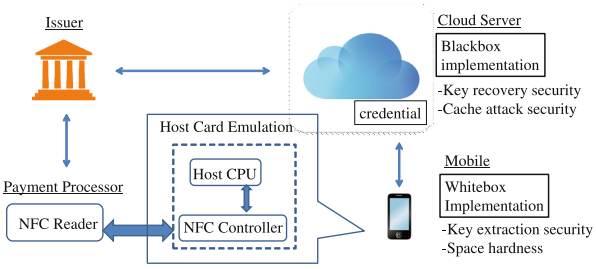


**Fig. 2.** Cloud-based mobile payments with HCE: Cloud server sends tokenized payment credentials provided by the issuer, to the mobile. Mobile phone transfers payment data with tokenized payment credentials to the payment processor via HCE. Payment processor sends it to the issuer for authorization.

implementation in the user-controlled device only. Such a device is now merely a part of a larger system, as e.g. in cloud computing or cloud-based payment. In addition, as whitebox cryptography inherently addresses resistance to malware, Trojans and zero-day vulnerabilities, it will find more and more applications in banking and other security-critical settings as well.

For illustrative purposes, we mention three application scenarios for whitebox cryptography, see also Figs. 1 and 2.

**DRM in the Cloud.** DRM-based services have moved to cloud-based contents distribution systems such as Adobe Primetime Cloud DRM [1] and Akamai's Secure Cloud-Based Workflows for Premium Content [2]. State-of-the-art contents distribution services often utilize IaaS (Infrastructure as a Service), for instance, Google cloud platform, IBM, Amazon AWS and Microsoft Azure, in order to optimize costs and to scale infrastructure. This application is illustrated in Fig. 1.

On the user device that plays the contents, whitebox implementation shall protect the contents key against key extraction and decomposition attacks [6, 28, 39] and recent side-channel attacks [11, 35]. A useful security property in this

context is space hardness, which aims to mitigate code lifting, and discourages the adversary from illegally distributing the code due to its large size [9].

On the cloud server that distributes the contents, a blackbox implementation is used to deal with a large number of user keys simultaneously, since running whitebox implementations for all users would require a huge amount of memory. Though usually much better protected than the player devices, cloud computing infrastructures do pose additional threats to the application. Namely, they are based on co-residency and multi-tenancy, i.e. the user runs multiple virtual machines (VMs) in the hardware resources of the same physical machine. Therefore, VM isolation raises a new security concern: *cache timing attacks* which exploit the fact that cache memory access times are data dependent. This may allow one to extract the secret key, given shared cache across co-located VMs. With the rapidly increasing deployment of cloud services, cache timing attacks have lately received a lot of attention [18,22,23,34]. Thus, cloud service providers have to deal with countermeasures. Indeed, having seen the novel cache timing attacks of [23,41], VMware made memory deduplication an opt-in feature, and Amazon disabled deduplication on its EC2 cloud servers. However, Irazouqui et al. show that attacks exploiting the L3 shared cache are still applicable even if such system-level countermeasures are deployed [22]. Thus, this threat has to be addressed at the cipher implementation level as well.

All in all, for DRM applications in the cloud, the blackbox cipher implementation should be secure against cache timing attacks on the cloud server, whereas the whitebox implementation should provide key extraction security and space hardness on the consumer device.

**Host Card Emulation in Cloud-based Mobile Payments.** NFC (Near Field Communication) is extensively used in applications such as payment systems. A standard NFC payment implementation employs a mobile phone with credentials stored inside a hardware-based secure element. HCE (Host Card Emulation) is a technology that enables NFC transactions in a pure software environment without secure elements — here anyone can create a mobile application without depending on the secure element. This allows one to launch new payment services in a more flexible way with a much less complex ecosystem. Thus, HCE is expected to become a game changer for mobile payments. Google provides the HCE architecture from Android 4.4 Kitkat on, by which anyone can emulate an NFC smart card for a payment service. Moreover, Visa and MasterCard also support the cloud-based HCE payments. In the HCE, instead of expensive secure hardware, credentials are stored in alternative media such as cloud. Figure 2 provides an overview of cloud-based payment systems with HCE.

In cloud-based payments, resilient whitebox cryptography on the mobile phone is central to the overall security. More precisely [29,37], a whitebox implementation shall replace the secure element in two ways. First, it should protect sensitive data such as tokens, payment information and card data from malware and spyware possibly running on the same CPU. Second, it should ensure that legitimate devices and users are accessing their payment credentials in the cloud by means of secure authentication between the cloud and the device.

From the implementation viewpoint, a mobile phone may not have rich resources, and available memory can be restrictive. Thus, the deployed whitebox cipher shall support variable sizes of its whitebox tables to meet a variety of implementation demands. In the cloud, which manages credentials, the corresponding blackbox implementation should prevent cross-VM cache timing attacks [18,22,23,34] similar to the previous application.

**Memory-Leakage Resilient Software.** Leakage of memory by vulnerabilities such as buffer overflows, cold boot attacks [20], bus monitoring attacks, Trojans and malware, or heartbleed-type vulnerabilities is a major problem in today's software. The notion of space hardness has been used to restrict the effect of memory leakage in applications where the leakage channel from the implementation environment to the adversary's backend is of limited capacity [9]. In particular, the use of space-hard whitebox cryptography can mitigate the damage of a memory-leakage vulnerability in security-critical systems. Indeed, those are typically insulated from the Internet, making it infeasible for Trojans to use low-capacity covert and side channels for the transmission of necessary key material if space-hard ciphers are employed.

Thus, for a memory-leakage resilient software implementation, the space hardness is necessary. It can be considered as a class of leakage resilient cryptography in bounded retrieval model where malware has complete control over the computer but can only send out a bounded amount of information.

### 1.3 Existing Whitebox Constructions

In order to meet some of the demands arising from applications, several whitebox constructions have been proposed.

**Whitebox Implementations of DES and AES.** Whitebox implementations of DES and AES were first proposed by Chow et al. in [14,15]. Their approach is to find a representation of the algorithm as a network of look-ups in randomized and key-dependent tables. In the wake of these seminal papers, several further variants of whitebox implementations for DES and AES were proposed [12,24,27,40]. However, all published whitebox solutions for DES and AES to date have been *practically* broken by key extraction and table-decomposition attacks [6,26,30,31,39].

**ASASA.** Dedicated whitebox block ciphers were proposed by Biryukov et al. in [7] at ASIACRYPT'14. They are based on the ASASA structure that consists of two secret nonlinear layers (S) and three secret affine layers (A), with affine and nonlinear layers interleaved. The security of ASASA against the key extraction in the whitebox setting relies on the hardness of the decomposition problem for ASASA. Unfortunately, efficient decomposition attacks on ASASA have been proposed [28]. The security of constructions based on multiple secret nonlinear and linear layers is still to be explored and seems hard to evaluate, despite several cryptanalytic efforts [8,10,38]. Moreover, generic ASASA-type constructions are difficult to implement in the constant-time fashion in the black box, which makes them potentially susceptible to side channel leakage.

**SPACE.** At CCS '15, Bogdanov and Isobe proposed a family of whitebox-secure block ciphers SPACE [9]. The design of SPACE is such that the security against key extraction and decomposition attacks in the whitebox setting reduces to the well-studied problem of key recovery for block ciphers in the standard blackbox setting. Their approach is to construct the whitebox table from a well-understood standard block cipher (AES in their example) by constraining the plaintext and truncating the ciphertext. Furthermore, to mitigate code lifting, they proposed the new security notion of *space hardness* which is a generalization of the weak whitebox security notion of [7]. Space hardness quantifies security against code lifting by the amount of code that needs to be extracted from an implementation by a whitebox adversary to maintain its functionality with a certain probability.

However, in order to strongly guarantee security against key extraction and space hardness in the whitebox setting, SPACE employs a very conservative design strategy. Namely, a target-heavy Feistel construction is deployed that does not allow for parallel or even pipelined implementations. Moreover, the internal F-function of SPACE requires one full 10-round AES-128 call. As estimated in [9], at least 128 full-round AES-128 calls are necessary to perform a single block encryption. That appears rather unacceptable in real-world applications. However, it's possible to derive a constant-time implementation of SPACE in the black box.

Thus, all existing designs have important practical limitations. This paper aims to bridge this gap by a novel design that addresses the key extraction security, the decomposition security (space hardness), constant-time blackbox implementation requirement as well as efficiency issues simultaneously.

### 1.4 Our Contributions

The contributions of this paper are as follows.

**Design of SPNbox: New Efficient Whitebox Block Cipher.** We propose SPNbox, a new family of space-hard block ciphers, which significantly improves upon the SPACE ciphers proposed at CCS 2015 [9]. While SPACE is based on a target-heavy Feistel construction, SPNbox is an SPN-type design with small block ciphers as the key-dependent S-boxes. In order to efficiently utilize the parallelism offered by both standard SIMD and the AES-NI instructions on contemporary microprocessors, the small block ciphers are based on the AES round transformation. The resulting parallelization opportunities allow for significantly faster implementations both in the black box and in the white box. At the same time, similarly to SPACE, SPNbox still offers all important whitebox security properties of quantifiable space hardness as well as reduction of key extraction security to the blackbox key-recovery security of the underlying block cipher. See Sect. 2.

**Security Analysis of SPNbox in the Black Box.** Our constructions come with security analysis as block ciphers. As the overall design as well as the design of underlying small block cipher follows the principles of substitution-permutation networks, we use the well-established tools of symmetric-key

cryptanalysis. See Sect. 3. In addition, we stress that our ciphers are secure against new types of attacks such as differential computational and differential fault attacks [11,35] in the white box as well as cross-VM cache timing attacks for cloud in the black box [18,22,23,34].

**Refined Compression Attack Settings.** Resistance to decomposition attacks is formalized by the notions of weak whitebox security and incompressibility [7], $(M, Z)$-space hardness and strong $(M, Z)$-space hardness [9] as well as by a related notion of $(\lambda, \delta)$ compressibility [16]. As opposed to previous studies of space hardness [9] that did not go beyond a weak whitebox adversary, this paper considers various levels of space hardness for table-based whitebox implementations, which are classified with respect to the adversary's abilities such as types of table accesses, knowledge about the execution environment or reverse engineering capabilities. This covers a very wide class of real-world adversaries that are thinkable in applications. In particular, we introduce known-space, chosen-space and adaptively-chosen-space attacks on space hardness. See Sect. 4.

**Provable Bounds on Space Hardness.** Moreover, we obtain bounds on space hardness in all those adversarial models under the assumption that the underlying tables are secure against decomposition, which is in turn guaranteed by the security of the underlying small block ciphers in the standard blackbox setting. This enables us to obtain rigorous upper bounds on the success probability, given a space of size $M$, in each adversarial model. These are the *first* security bounds on space hardness for table-based whitebox implementations, while previous results only roughly evaluate the security by an attack-based approach [9]. Furthermore, we apply our bounds to SPNbox and SPACE ciphers. As a result, we update the evaluations of space hardnesses of SPACE ciphers, and show that SPNbox offers a conservative level of space hardness in each adversary model.

**Efficient Optimized Software Implementations of SPNbox and SPACE.** We implement both SPNbox and SPACE families of whitebox block ciphers on Intel Skylake and ARMv8. Our implementations use SIMD/AVX, AES-NI and NEON extensions whenever possible to optimize performance. As a result, we report that instances of SPNbox achieve speed-ups of up to 6.5 times in the black box and up to 18 times compared to SPACE in the whitebox setting. See Sect. 5.

## 2 SPNbox: Efficient Space-Hard Block Ciphers

### 2.1 Design Choices

*From Feistel to nested SPN.* The SPACE family of space-hard block ciphers employs a very conservative design strategy which involves using the full 10-round AES-128 transformation, even for 8-bit inputs. Furthermore, its Feistel structure prevents the exploitation of any parallel execution or pipelining possibilities. At the same time, it seems likely that the security margin offered by the proposed SPACE instances can be reduced without ill effects.

The requirement of parallelism immediately points to an SPN-type design. For the desired level of space hardness, key-dependent S-boxes of varying size can be employed. This can then be combined with a public linear MDS diffusion layer operating on the entire state, allowing rigorous security arguments for standard blackbox security.

Within this design framework, it remains to construct key-dependent S-boxes of different sizes (for instance 8, 16, 24 and 32 bits as in SPACE). This is accomplished by using smaller internal block ciphers, which are themselves SPNs, yielding a *nested SPN structure* [3]. For the reasons of efficiency, security and side-channel protection, it is desirable to base these internal SPNs on the AES round transformation, especially given the availability of the AES-NI instructions [19]. The efficiency requirements also dictate that little or no truncation should take place, and ideally, the AES round transformation should be used to compute some of the larger S-boxes in parallel.

In order to also have an efficient implementation for the inverse cipher, the design should employ involutory MDS matrices wherever possible. Since we mainly target high-performance software implementations, our selection criteria for efficient MDS matrices differs somewhat from the widely studied area of lightweight hardware implementations as in [36]: In software, arbitrary bit permutations are costly, which means that a matrix with smaller coefficients but higher theoretical XOR count can result in a more efficient SIMD implementation.

*Efficient Constant-Time Small Block Ciphers.* We note that these small SPN-type block ciphers used to construct the key-dependent S-boxes are of potential independent interest: Block ciphers of sizes smaller than 32 bit are virtually unstudied, and an AES-NI based implementation further allows an efficient constant-time implementation, which avoids the pitfalls of key-dependent table lookups (which is the usual way of implementing small nonlinear functions due to efficiency reasons though bit-sliced implementations may be possible as well). In addition, in order to prevent the differential computational attacks [11], this small SPN-type block cipher depends on 128 bits of key information.

### 2.2 Specification

We now define the SPNbox family of block ciphers and their concrete instantiations SPNbox-8, SPNbox-16, SPNbox-24, and SPNbox-32. SPNbox-$n_{in}$ is a substitution-permutation network (SPN) with a block length of $n$ bits, a $k$-bit secret key, and based on $n_{in}$-bit substitution boxes. For SPNbox-8, SPNbox-16 and SPNbox-32, the block length is $n = 128$ bits, whereas SPNbox-24 has $n = 120$. While SPNbox can support a wide range of key sizes, we use $k = 128$ for concreteness in the following.

*Representation of Finite Fields.* We will in the sequel sometimes view the set $\{0,1\}^m$ of bit strings as the finite field $\mathrm{GF}(2^m)$. For this, we identify $\mathrm{GF}(2^m)$ with the quotient ring $\mathrm{GF}(2)[\mathrm{x}]/(p)$ for a suitable irreducible polynomial $p \in$

$\mathrm{GF}(2)[\mathtt{x}]$. An $m$-bit string $a_{m-1}a_{m-2}\cdots a_1a_0 \in \{0,1\}^m$ then corresponds to the polynomial $a_{m-1}\mathtt{x}^{m-1} + a_{m-2}\mathtt{x}^{m-2} + \cdots + a_1\mathtt{x} + a_0 \in \mathrm{GF}(2^m)$. We write such an element in a hexadecimal representation of its bit string, e.g. $4_x$ for 100.

For $\mathrm{GF}(2^8)$, we use the same irreducible polynomial as the AES, namely $p(\mathtt{x}) = \mathtt{x}^8 + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x} + 1$. Similarly, we use $p(\mathtt{x}) = \mathtt{x}^{16} + \mathtt{x}^5 + \mathtt{x}^3 + \mathtt{x} + 1$ for $\mathrm{GF}(2^{16})$, $p(\mathtt{x}) = \mathtt{x}^{24} + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x} + 1$ for $\mathrm{GF}(2^{24})$ and $p(\mathtt{x}) = \mathtt{x}^{32} + \mathtt{x}^7 + \mathtt{x}^3 + \mathtt{x}^2 + 1$ for $\mathrm{GF}(2^{32})$, respectively.

**State.** The state of SPNbox-$n_{in}$ is organised as a vector of $t \stackrel{\text{def}}{=} n/n_{in}$ elements of $n_{in}$ bits each:

$$X = \{X_0, \ldots, X_{t-1}\}.$$

Each of the $n_{in}$-bit elements $X_i$ can in turn be represented by a vector of $\ell \stackrel{\text{def}}{=} n_{in}/8$ bytes: $X_i = \{X_{i,\ell-1}, \ldots, X_{i,0}\}$.

**Key Schedule.** The $k$-bit master key is expanded to $(R_{n_{in}} + 1)$ round keys $k_0, \ldots, k_{R_{n_{in}}}$ of $n_{in}$ bits using any generic key derivation function (KDF) [32]:

$$(k_0, \ldots, k_{R_{n_{in}}}) = \mathrm{KDF}(k, n_{in} \cdot (R_{n_{in}} + 1)).$$

For example, one can use the SHAKE extendable output function which is based on the SHA-3 hash [33].

**Round Transformation.** The encryption of a plaintext $X^0$ to a ciphertext $X^R$ is accomplished by applying $R$ rounds of the following round transformation to the plaintext:

$$X^R = \left(\bigcirc_{r=1}^{R} (\sigma^r \circ \theta \circ \gamma)\right)(X^0).$$

For all concrete proposals SPNbox-8, SPNbox-16, SPNbox-24 and SPNbox-32, we set the number of rounds to $R = 10$. We now define in turn each of the components $\gamma, \theta$ and $\sigma^r$. An overview of the round transformation is given in Fig. 3

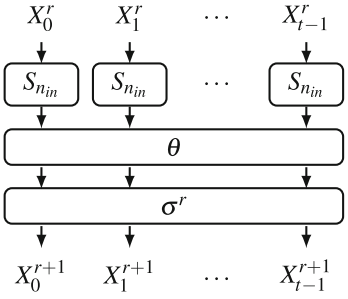


**Fig. 3.** Round transformation of SPNbox.

*The Nonlinear Layer* $\gamma$. $\gamma$ is a nonlinear substitution layer, in which $t$ key-dependent identical bijective $n_{in}$-bit S-boxes are applied to the state:

$$\begin{aligned}\gamma : \mathrm{GF}(2^{n_{in}})^t &\to \mathrm{GF}(2^{n_{in}})^t \\ (X_0, \ldots, X_{t-1}) &\mapsto (S_{n_{in}}(X_0), \ldots, S_{n_{in}}(X_{t-1}))\,.\end{aligned}$$

In SPNbox-$n_{in}$, the substitution $S_{n_{in}}$ is realised by an internal small block cipher of block length $n_{in}$, which will be defined in the next subsection.

*The Linear Layer* $\theta$. $\theta$ is a linear diffusion layer that applies a $t \times t$ MDS matrix to the state:

$$\begin{aligned}\theta : \mathrm{GF}(2^{n_{in}})^t &\to \mathrm{GF}(2^{n_{in}})^t \\ (X_0, \ldots, X_{t-1}) &\mapsto (X_0, \ldots, X_{t-1}) \cdot M_{n_{in}}.\end{aligned}$$

We denote by $\mathrm{cir}\,(a_0, \ldots, a_{t-1})$ the $t \times t$ circulant matrix $A$ with the coefficients $a_0, \ldots, a_{t-1}$ in the first row; and by $\mathrm{had}\,(a_0, \ldots, a_{t-1})$ the $t \times t$ Hadamard matrix $A$ with coefficients $A_{i,j} = a_{i \oplus j}$, with $t$ a power of two.

For the concrete proposals SPNbox-$n_{in}$with $n_{in} = 32, 24, 16, 8$, the matrix $M_{n_{in}}$ is then respectively defined as follows:

$$\begin{aligned}M_{32} &= \mathrm{cir}\,(\mathtt{1}_x, \mathtt{2}_x, \mathtt{4}_x, \mathtt{6}_x) && \text{for } n_{in} = 32, \\ M_{24} &= \mathrm{cir}\,(\mathtt{1}_x, \mathtt{2}_x, \mathtt{5}_x, \mathtt{3}_x, \mathtt{4}_x) && \text{for } n_{in} = 24, \\ M_{16} &= \mathrm{had}\,(\mathtt{1}_x, \mathtt{3}_x, \mathtt{4}_x, \mathtt{5}_x, \mathtt{6}_x, \mathtt{8}_x, \mathtt{b}_x, \mathtt{7}_x) && \text{for } n_{in} = 16,\end{aligned}$$

and

$$\begin{aligned}M_8 = \mathrm{had}\,(&\mathtt{08}_x, \mathtt{16}_x, \mathtt{8a}_x, \mathtt{01}_x, \mathtt{70}_x, \mathtt{8d}_x, \mathtt{24}_x, \mathtt{76}_x, \\ &\mathtt{a8}_x, \mathtt{91}_x, \mathtt{ad}_x, \mathtt{48}_x, \mathtt{05}_x, \mathtt{b5}_x, \mathtt{af}_x, \mathtt{f8}_x) \\ &\text{for } n_{in} = 8.\end{aligned}$$

Note that $M_{32}, M_{16}$ and $M_8$ are involutions. $M_{32}$ and $M_{16}$ are the matrices used in the block ciphers Anubis [4] and Khazad [5], respectively. $M_8$ is an optimised involutory Hadamard-Cauchy matrix proposed at FSE 2015 [36].

*The Affine Layer* $\sigma^r$. $\sigma^r$ is an affine layer that adds round-dependent constants to the state:

$$\begin{aligned}\sigma^r : \mathrm{GF}(2^{n_{in}})^t &\to \mathrm{GF}(2^{n_{in}})^t \\ (X_0, \ldots, X_{t-1}) &\mapsto \left(X_0 \oplus C_0^r, \ldots, X_{t-1} \oplus C_{t-1}^r\right),\end{aligned}$$

with $C_i^r \stackrel{\text{def}}{=} (r-1) \cdot t + i + 1$ for $0 \le i \le t-1$.

**The Underlying Small Block Ciphers.** The key-dependent $n_{in}$-bit bijective S-boxes $S_{n_{in}}$ in the nonlinear layer $\gamma$ are small SPN-type block ciphers themselves. They are based on the round transformation of the AES and consist of $R_{n_{in}}$ rounds operating on a state $x = \{x_0, \ldots, x_{\ell-1}\}$ of $\ell \stackrel{\text{def}}{=} n_{in}/8$ bytes:

$$S_{n_{in}} : \mathrm{GF}(2^8)^\ell \to \mathrm{GF}(2^8)^\ell$$
$$x \mapsto \left(\bigcirc_{i=1}^{R_{n_{in}}} \left(\mathsf{AK}^i \circ \mathsf{MC}_{n_{in}} \circ \mathsf{SB}\right)\right)(\mathsf{AK}^0(x)).$$

Here, $\mathsf{SB}$ denotes the application of the AES S-box to each byte of the state. For $0 \le i \le R_{n_{in}}$, $\mathsf{AK}^i$ is defined as the addition of the round key $k_i$ (as expanded by the key schedule) by XOR. $\mathsf{MC}_{n_{in}}$ implements an MDS diffusion layer on all $\ell$ bytes of the state. It is based on the AES $\mathsf{MixColumns}$ operation. For the concrete proposals of $n_{in} = 32, 24, 16$, it is defined as the multiplication of $x$ with the matrices

$$A_{32} = \mathrm{cir}\,(2_x, 1_x, 1_x, 3_x) \qquad \text{for } n_{in} = 32,$$
$$A_{24} = \begin{pmatrix} 2_x & 1_x & 1_x \\ 3_x & 2_x & 1_x \\ 1_x & 3_x & 2_x \end{pmatrix} \qquad \text{for } n_{in} = 24,$$
$$A_{16} = \begin{pmatrix} 2_x & 1_x \\ 3_x & 2_x \end{pmatrix} \qquad \text{for } n_{in} = 16,$$

respectively. For $n_{in} = 8$, $\mathsf{MC}_{n_{in}}$ is the identity mapping. Note that $A_{32}$ is the AES MixColumns matrix (adjusted for Intel's byte order), while $A_{24}$ and $A_{16}$ are obtained from $A_{32}$ as $(x, y, z, 0) \times A_{32}$ and $(x, y, 0, 0) \times A_{32}$, respectively. As square submatrices of $A_{32}$, all derived matrices are also $\ell \times \ell$ MDS matrices over $\mathrm{GF}(2^8)$. An overview of the round transformation is given in Fig. 4.

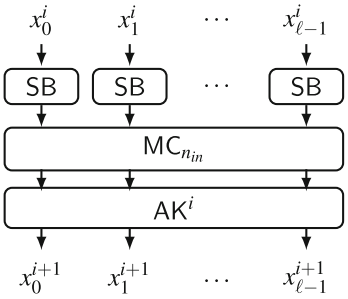


**Fig. 4.** Round transformation of the underlying block ciphers $S_{n_{in}}$.

The number of rounds for each concrete proposal is defined as $R_{32} = 16$, $R_{24} = 20$, $R_{16} = 32$ and $R_8 = 64$.

### 2.3 SPNbox vs ASASA

Both SPNbox and the ASASA construction are based on the classical substitution-permutation network structure, consisting however of secret key-dependent S-box and public linear layers. The main constructive difference is how to construct the secret key-dependent S-box. However, this is the discrepancy having far-reaching practical consequences both in terms of security arguments and implementation.

In the ASASA construction, as its name suggests, tables are based on the ASASA structure that consists of two secret nonlinear layers (S) and three secret affine layers (A), with affine and nonlinear layers interleaved. On the other hand, SPNbox is based on the SPN-type small block cipher that consists of the *public* nonlinear and linear layer, and secret key XOR layers.

Regarding the security of the whitebox implementation, the difficulty of the key extraction and the decomposition problem for ASASA relies on the hardness of the decomposition problem for ASASA, which is still to be explored and seems hard to evaluate, despite several cryptanalytic efforts [8,10,38]. Actually, efficient decomposition attacks on ASASA have been proposed [28]. On the other hand, SPNbox relies on well analyzed problem of the key recovery attack of the block cipher in the standard blackbox setting.

In the blackbox implementation, assuming the random choice of secret S-boxes, the substitution layer of ASASA is realized by the table based implementation due to the secrecy of underlying component, and is impossible to optimize the performance by AES-NI. The table-based blackbox implementation of the ASASA is not secure against cache timing attacks similar to the table-based blackbox AES implementation [18,22,23,34].

## 3 Security in the Black Box: Analysis as a Block Cipher

We evaluate the general construction of SPNbox-8, -16, -24 and -32, modeling the underlying small block cipher as pseudorandom permutation. We furthermore analyze the security of the underlying small block ciphers $S_{n_{in}}$ against cryptanalytic attacks. Finally, we evaluate the security against cross-VM cache timing attacks for cloud application.

### 3.1 General Construction

First, we evaluate the security of the general construction of SPNbox-8, -16, -24 and -32, assuming an underlying small block cipher, i.e. the key-dependent $n_{in}$-bit bijective S-boxes $S_{n_{in}}$, is a pseudo random permutation. The generic construction of all variants is a 10-round SPN-type construction.

**Differential Cryptanalysis.** Here we analyze the differential properties of an $n_{in}$-bit permutation $S_{n_{in}} : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{in}}$. Given input difference $a$ and output difference $b$, the differential probability of function $f$ is defined as

$$DP(a,b) = \#\{(v,u)|u \oplus v = a \text{ and } f(v) \oplus f(u) = b\}$$

for $u, v \in \{0,1\}^{n_{in}}$. The bound of the maximum differential probability $MDP$ is proved as follows [21].

$$Pr\Big(\frac{n \ln 2}{2^{n-1} \ln n} \leq MDP < \frac{n}{2^{n-1}}\Big) \approx 1$$

Suppose that the maximum differential probability of $S_{n_{in}}$ of SPNbox-8, -16, -24 and -32 to be $2^{-4}$ $(= 8/2^7)$, $2^{-11}$ $(= 16/2^{15})$, $2^{-18.42}$ $(= 24/2^{23})$ and $2^{-26}$ $(= 32/2^{31})$, respectively. Due to properties of MDS diffusion matrices. SPNbox-8, -16, -24 and -32 have at least 34, 18, 12 and 5 active $S_{n_{in}}$ after 4, 4, 4 and 2 rounds.

**Linear Cryptanalysis.** Now we analyze the linear properties of an $n_{in}$-bit permutation $S_{n_{in}}$: $\{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{in}}$.

Given an input mask $\alpha$ and an output mask $\beta, \alpha, \beta \in \{0,1\}^{n_{in}}$, the correlation of a linear approximation $(\alpha, \beta)$ for a function $f$: $\{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{in}}$ is defined as

$$Cor = 2^{-n_{in}}[\#\{x \in \{0,1\}^{n_{in}} | \alpha \cdot x \oplus \beta \cdot f(x) = 0\} - \\ \#\{x \in \{0,1\}^{n_{in}} | \alpha \cdot x \oplus \beta \cdot f(x) = 1\}.$$

The linear probability $LP$ of $(\alpha, \beta)$ is defined as $Cor^2$. For a fixed-key block cipher, the maximum linear probability $MLP$ is normally distributed in mean $\approx (1.38 \cdot 2n - \ln(1.38 \cdot 2n) + 1) \cdot 2^{-n}$ and standard deviation $\approx 2.6 \times 2^{-n}$ [21].

Suppose that the maximum linear probability of $S_{n_{in}}$ of SPNbox-8, -16, -24 and -32 to be $2^{-3.67}$ $(= 19.99 \cdot 2^{-8})$, $2^{-10.62}$ $(= 41.37 \cdot 2^{-16})$, $2^{-18.02}$ $(= 63 \cdot 2^{-24})$ and $2^{-25.61}$ $(= 84 \cdot 2^{-32})$, respectively. SPNbox-8, -16, -24 and -32 have at least 51, 18, 12 and 5 active $F_i^{(j)}(x)$ after 6, 4, 4 and 2 rounds.

**Other Cryptanalysis.** Any input difference nonlinearly affects all states after one round due to the MDS matrix. Following the miss-in-the-middle approach, after 3 rounds, we have not found any useful impossible differentials for the respective variants. Also, a 2.5-round generic integral distinguisher against the SPN-type construction is proposed [8]. We also consider other-types of attacks including a higher order differential, a truncated differential, a slide, and an algebraic attack. Consequently, we expect that none of them work better than brute force attacks.

### 3.2 The Underlying Small Block Ciphers

We evaluate the security of underlying small block ciphers $S_{n_{in}}$. These are based on well-analyzed AES components such as the inversion base 8-bit S-box and the MDS circulant matrix on GF($2^8$).

**Differential/Linear Cryptanalysis.** The differential/linear probability of 8-bit S-box is $2^{-6}$. $S_8$, $S_{16}$, $S_{24}$, and $S_{32}$ have at least 2, 3, 4 and 10 differentially/linearly active S-boxes after 2, 2, 2 and 4 rounds, respectively. We therefore expect all $S_{n_{in}}$, for $n_{in} = 8, 16, 24, 32$, to not have any differential or linear trails with probabilities exceeding the bound $2^{-n_{in}}$ after 2, 2, 2 and 4 rounds, respectively. Since they are proposed with much higher numbers of rounds, they offer ample security margin.

**Meet-in-the-Middle and Other Cryptanalysis.** In each cipher, four times 128-bit key information is involved, and one round already achieves full diffusion. Thus, we believe that the small block ciphers are secure against MitM attacks. We developed MitM attacks on each variant using splice and cut, biclique and partial matching techniques. However, we did not find full round attacks.

Considering further attacks, the byte-oriented structure combined with full diffusion after 1 round means that for impossible (truncated) differential attacks, and integral and higher order differential attacks, we can at most construct cryptanalytic properties spanning 3 and 4 rounds, respectively. All small block ciphers are proposed with much significantly higher numbers of rounds. Finally, the use of distinct round constants in the key schedule precludes slide attacks.

### 3.3 Cache Timing Attack

There are several techniques exploiting cache information over VM isolations in the cloud: the Prime+Probe attack [22] and Flush+Reload attacks [18,23,41]. All attacks make use of timing differences between cache hits and misses. Our key-dependent small block ciphers are designed to be executed in constant time by using AES-NI, and there are no cache accesses during key-dependent operations. Thus, it is impossible to mount cache timing attacks against the blackbox implementation of SPNbox.

## 4 Security in the White Box: Analysis of Space Hardness

In this section, we first evaluate the security against key-extraction and decomposition attacks in the whitebox model. Second, we evaluate the difficulty of code lifting attacks by notions of *weak and strong space hardness* [9]. We generalize the adversarial models of space hardness to capture a wide class of adversaries: from adversaries with limited control (greybox) to stronger ones with more knowledge of the computational platform and reverse engineering abilities (whitebox). Then, we show bounds for weak and strong space hardness for table-based whitebox cryptography under the assumption that *tables are secure against key extraction and table decomposition attacks*, i.e. it is computationally infeasible to compress the tables in the whitebox models[1]. By contrast, the

[1] Whitebox AES implementations [12,14,24,40] and the ASASA construction [7] do not satisfy the assumption due to practical decomposition attacks [6,26,28,30,31].

authors of [9] evaluate the space hardness of their proposals only by attack-based approaches, called compression attack. Finally, we evaluate the security against recent advanced side-channel attacks [11,35].

### 4.1 Key Extraction and Table Decomposition Attacks

As the tables are constructed from small block ciphers, the security of key-extraction and decomposition attacks in the whitebox model reduces to the key recovery problem for these small block ciphers in the blackbox model (which is evaluated in Sect. 3). The advantage of key extraction in the whitebox model for SPNbox, $\mathrm{Adv}_{\text{KE-WB}}$, is upper-bounded by the advantage of the key recovery for the underlying block cipher in the blackbox model, $\mathrm{Adv}_{\text{KR-BB}}$: $\mathrm{Adv}_{\text{KE-WB}} \leq \mathrm{Adv}_{\text{KR-BB}}$.

### 4.2 Existing Notions of Space Hardness

The difficulty of a decomposition attack is measured by space hardness that is summarized here. The whitebox implementation of a cipher should resists decomposition: Instead of a secret key, the adversary can directly use the implementation itself as a larger effective key. In particular, he can isolate the program code where the key is embedded in order to copy the functionality of encryption/decryption routines and to utilize it in a stand-alone manner. We refer to decomposition attacks as *code lifting attack*. If a code lifting attack succeeds, the adversary gets the advantage which is almost the same as key extraction, i.e. he can encrypt/decrypt any plaintext/ciphertext.

To formalize the difficulty of code lifting, the notions of weak white-box security and incompressibility have been introduced in [7]. To capture the resistance towards compression attacks in a more fine-grained fashion, two further security notions were introduced in [9]: $(M, Z)$*-space hardness* and *strong* $(M, Z)$*-space hardness*. Space hardness measures the difficulty of compressing the whitebox implementation of a cipher, and quantifies security against code lifting by the amount of code that needs to be extracted from the implementation by a white-box adversary to maintain its functionality. Moreover, Delerablee et al. propose a related notion of $(\lambda, \delta)$ compressibility [16]. However, the latter aims to evaluate the difficulty of code compression, given the full code. Space hardness [9] assesses the difficulty of isolating code from execution environments, namely, code lifting, by the amount of the data. Thus, it covers a wide class of adversaries: from the one with limited control all the way to the stronger ones with full code and complete access to the environments. For the sake of clarity, the paper at hand refers to $(M, Z)$-space hardness of [9] as *weak* $(M, Z)$-space hardness:

**Definition 1 (Weak $(M, Z)$-space hardness [9]).** *An implementation of a block cipher $E_K$ is weakly $(M, Z)$-space hard if it is infeasible to encrypt (decrypt) any randomly drawn plaintext (ciphertext) with probability of more than $2^{-Z}$ given any code (table) of size less than $M$ bits.*

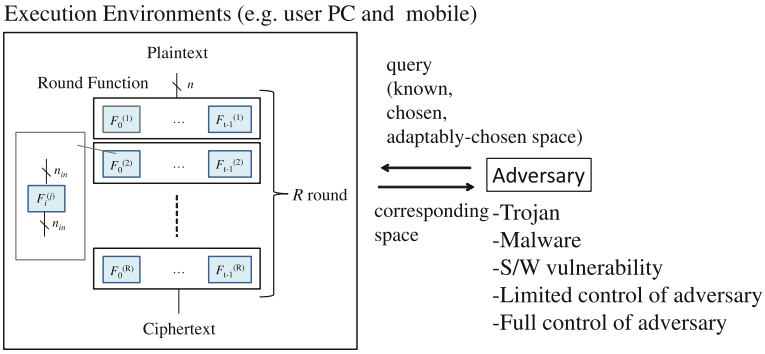


**Fig. 5.** Target block cipher construction for the white box and its adversarial models

Weak $(M, Z)$-space hardness estimates the code (table) size $M$ that needs to be isolated from the whitebox environment to be able to encrypt (decrypt) any plaintext (ciphertext) with a success probability larger than $2^{-Z}$.

**Definition 2 (Strong $(M, Z)$-space hardness [9]).** *An implementation of a block cipher $E_K$ is strongly $(M, Z)$-space hard if it is infeasible to obtain a valid plaintext and ciphertext pair with probability higher than $2^{-Z}$ given the code (table) of size less than $M$ bits.*

Strong $(M, Z)$-space hardness assumes an adversary who tries to find any valid input/output pair. It is relevant to message authentication codes in the context of forgeries.

### 4.3 Target Construction

To simplify our evaluation of space hardness in the sequel, we define a target construction: an $n$-bit block cipher that is encrypted/decrypted by *key dependent table-based* implementations in the whitebox environment as shown in Fig. 5. Let the input and output sizes of each table be $n_{in}$ and $n_{out}$, respectively, and the number of rounds be $R$, where the each round consists of $t$ tables. We denote $j$-th table in round $r$ as a function $F_j^{(r)}$: $\{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$ for $j \in \{0, 1, \ldots, t-1\}$ and $r \in \{1, 2, \ldots, R\}$. In the cases of SPNbox and SPACE [9], all tables are identical, and the total table sizes $T$ is estimated as $T = (2^{n_{in}} \times n_{out})$.

### 4.4 Adversary Models of Space Hardness

We consider three adversary models that are classified with respect to the adversary's ability, while previous works [9] do not specify the adversary model. In particular, we simulate the action of the adversary against the execution environments by access to the table (space) functions $F_j^{(r)}$ (see Fig. 5).

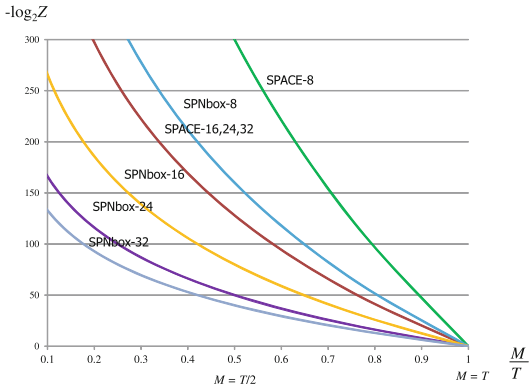


**Fig. 6.** Weak $(M, Z)$-space hardness of SPACE-8, 16, 24 and 32 and SPNbox-8, 16, 24 and 32 in known- and chosen-space attack

**Definition 3 (Known-Space (KS) Attack).** *The adversary obtains $q$ pairs of inputs and the corresponding outputs of tables $(x_i, F_j^{(r)}(x_i))$, $i \in \{0, 1, \ldots, q-1\}$, $j \in \{0, 1, \ldots, t-1\}$ and $r \in \{1, 2, \ldots, R\}$.*

**Definition 4 (Chosen-Space (CS) Attack).** *The adversary obtains $q$ pairs of inputs and the corresponding outputs of tables $(x_i, F_j^{(r)}(x_i))$ for a series of a priori chosen inputs $x_i$, $i \in \{0, 1, \ldots, q-1\}$, $j \in \{0, 1, \ldots, t-1\}$ and $r \in \{1, 2, \ldots, R\}$.*

**Definition 5 (Adaptively-Chosen-Space (ACS) Attack).** *The adversary obtains $q$ pairs of inputs and the corresponding outputs of tables $(x_i, F_j^{(r)}(x_i))$ for a series of adaptively chosen inputs $x_i$, $i \in \{0, 1, \ldots, q-1\}$, $j \in \{0, 1, \ldots, t-1\}$ and $r \in \{1, 2, \ldots, R\}$, namely he can choose $x_a$ after obtaining $(x_{a-1}, F_j^{(r)}(x_{a-1}))$.*

The known-space attack models the limited control of the adversary over the platform, where the adversary passively gets a part of space from the environments, e.g. with the aid of a trojan, malware, or a memory-leakage software vulnerability. The model is applicable to memory-leakage resilient cryptography where malware has complete control over the computer but can only send out a bounded amount of information [17].

The chosen-space attack captures the stronger adversary who has the ability to isolate any part of tables (space) with the knowledge of the memory layout, but the amount of data and the timing of access to the implementation are restricted due to the limited capacity of the communication channel and access controlled environments.

Finally, the adaptively-chosen-space attack assumes an adversary who has full access to the execution environment at *any time* by decompiler and debugger tools, e.g. IDA Pro and IL DASM, which is corresponding to the

original whitebox adversary defined in [14] and the assumption of ($\lambda$, $\delta$) compressibility [16].

Previous weak and strong $(M, Z)$-space hardness are evaluated by compression attacks [9]. The assumption of these attacks is classified as the known-table attack, i.e. weak KS-$(M, Z)$-space hardness and KS-$(M, Z)$-space hardness, respectively. Thus, previous evaluation of space hardness in [9] can capture only the weaker adversary than the standard whitebox adversary who has full access to the execution environment.

### 4.5 Weak Space Hardness

We show bounds for the weak $(M, Z)$-space hardness of the target construction in known-, chosen- and adaptively-chosen space attacks. Our evaluation assumes that the table decomposition is computationally infeasible as evaluated in Sect. 4.1, and input values of each table in the cipher are uniformly distributed, which is a reasonable assumption for block ciphers. The evaluation of the weak space hardness in the case where the adversary has a partial knowledge of a plaintext is provided in SubSect. 4.6.

**Known-Space Attack.** First, we introduce the following lemma.

**Lemma 1 (Inequality of Arithmetic and Geometric Means).** *For arbitary $n$ positive positive numbers $x_0, x_1, \ldots, x_{n-1}$, the inequality*

$$\sqrt[n]{x_0 \cdot x_1 \cdots x_{n-1}} \leq \frac{x_0 + x_1 + \ldots, x_{n-1}}{n}$$

*holds, with equality if and only if $x_0 = x_1, \ldots, = x_{n-1}$.*

There are various proofs in the literature, and for example we refer to [13].

For known-space attacks, we have the following theorem:

**Theorem 1.** *Given known space of size $M$, the probability that a randomly-drawn plaintext can be computed is upper bounded by $(M/T)^{tR}$.*

*Proof.* Let the number of known entries of each table $F_j^{(r)}$ be $\#F_j^{(r)}$ for $j \in \{0, 1, \ldots, t-1\}$ and $r \in \{1, 2, \ldots, r\}$. The probability that an input of a tables $F_j^{(r)}$ matches with known ones is estimated as $(\#F_j^{(r)}/2^{n_{in}})$. Hence, a randomly-drawn plaintext can be computed with the probability of

$$\prod_{j=0}^{t-1} \prod_{r=1}^{R} \frac{\#F_j^{(r)}}{2^{n_{in}}} = \left(\frac{1}{2^{n_{in}}}\right)^{tR} \prod_{j=0}^{t-1} \prod_{r=1}^{R} \#F_j^{(r)}.$$

Here the sum of the numbers of known inputs is expressed as $\sum_{j=0}^{t-1} \sum_{r=1}^{R} \#F_j^{(r)}$. According to Lemma 1, we have

$$\prod_{j=0}^{t-1} \prod_{r=1}^{R} (\#F_j^{(r)}) \leq \left(\frac{\sum_{j=0}^{t-1} \sum_{r=1}^{Rt} \#F_j^{(r)}}{tR}\right)^{tR}.$$

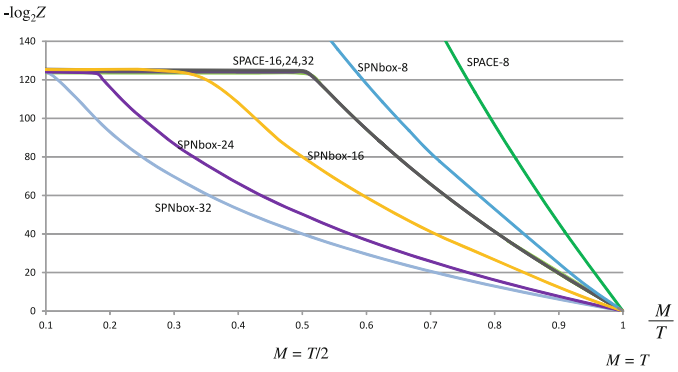


**Fig. 7.** Weak $(M, Z)$-space hardness of SPACE-8, 16, 24 and 32 and SPNbox-8, 16, 24 and 32 in adaptively-chosen attack.

Only if $\#F_0^{(1)} = \#F_1^{(1)} = \ldots = \#F_{t-1}^{(R)}$, the equation holds. Here, $M$ is estimated as $M = (\sum_{j=0}^{t-1}\sum_{r=1}^{R} \#F_j^{(r)} \cdot n_{out})/tR$ bits.

Thus, we have

$$\left(\frac{1}{2^{n_{in}}}\right)^{tR} \prod_{j=0}^{t-1}\prod_{r=1}^{R} \#F_j^{(r)} \leq \left(\frac{1}{2^{n_{in}}}\right)^{tR} \left(\frac{\sum_{j=0}^{t-1}\sum_{r=1}^{Rt} \#F_j^{(r)}}{tR}\right)^{tR}$$
$$\leq \left(\frac{1}{2^{n_{in}}}\right)^{tR} \left(\frac{M \cdot 2^{n_{in}}}{T}\right)^{tR} \square$$

From Theorem 1, we obtain weak KS-$(M, -\log_2((M/T)^{tR}))$-space hardness, i.e. given any known space of size $M$, it is infeasible to encrypt a randomly-drawn plaintext with the probability larger than $(M/T)^{tR}$. Figure 6 shows the relation between $M$ and $Z$ in terms of weak KS-$(M, Z)$ space hardness of SPACE-8, 16, 24 and 32 and SPNbox-8, 16, 24 and 32. For example, in SPNbox-16, given space of size $M = T/4$, the success probability is upper bounded by $2^{-160}$ $(= (2^{-2})^{8 \cdot 10})$ (Fig. 6).

**Chosen-Space Attack.** Due to the randomly-drawn plaintext, inputs of tables are unpredictable in advance even in the chosen-table attack. Thus, the chosen-space attack has no advantage over the known-space attack. We obtain weak CS-$(M, -\log_2((M/T)^{tR}))$-space hardness from Theorem 1.

**Adaptive-Space Attack.** The adversary is able to encrypt any plaintext by adaptively accessing the tables and computing round functions one by one. Thus he can prepare a set of pairs of plaintexts and the corresponding ciphertexts before a target plaintext is given. If the target plaintext is included in the set of prepared pairs, the corresponding ciphertext is obtained with the probability one.

Let us estimate how large space is necessary to compute $N$ plaintexts in advance. In the encryptions of $N$ plaintexts, it requires $N \cdot t \cdot R$ table accesses, and each table function $F_j^{(r)}$ has $N$ accesses. We provide the following Lemma.

**Lemma 2.** *For $q$ table accesses, the expected value of the number of used entries in the table is estimated as $(1 - ((2^{n_{in}} - 1)/2^{n_{in}})^q) \cdot 2^{n_{in}}$.*

*Proof.* An $i$-th entry of the table is used during $q$ table accesses with the probability of $(1 - ((2^{n_{in}} - 1)/2^{n_{in}})^q)$. There are $2^{n_{in}}$ entries in the table. □

Here, we define $(2^{n_{in}} - 1)/2^{n_{in}}$ as $e_{in}$. Using Lemma 2, we obtain Theorem 2.

**Theorem 2.** *Given adaptively-chosen space of size $M$, the probability that a randomly-drawn plaintext can be computed is upper bounded by $N \cdot 2^{-128} + (1 - N \cdot 2^{-128})(M/T)^{tR}$, where $N = \lceil log_{e_{in}}(1 - M/T)/tR \rceil$.*

*Proof.* According to Lemma 2, in order to compute $N$ pairs of plaintexts and the corresponding ciphertexts, it requires $(1 - (e_{in})^{N \cdot tR}) \cdot 2^{n_{in}} \cdot n_{out} = (1 - (e_{in})^{N \cdot tR}) \cdot T$-bit space. In the other words, adaptively-chosen space of size $M$ enables to compute $N (= \lceil log_{e_{in}}(1 - M/T)/tR \rceil)$ pairs of plaintexts and the corresponding ciphertexts. Then, the randomly-drawn plaintext is included in a set of the prepared pairs with probability of $2^{-128+N}$. Otherwise, given space of size $M$, the probability that the randomly-drawn plaintext can be computed is upper-bounded by $(M/T)^{tR}$ from Theorem 1. □

From Theorem 2, we obtain weak ACS-$(M, -\log_2(N \cdot 2^{-128} + (1 - N \cdot 2^{-128})(M/T)^{tR})$-space hardness. For example, in SPNbox-16, given $M = 0.46 \cdot T$ space, the success probability is upper bounded by $2^{-88.4}$ $(= 9 \cdot 2^{-128} + (1 - 9 \cdot 2^{-128}) \cdot (0.465)^{8 \cdot 10})$ (Fig. 8).

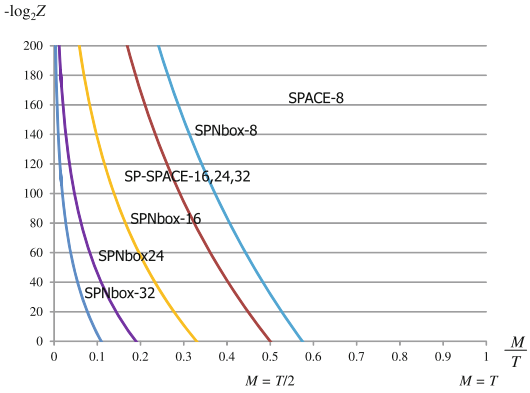


**Fig. 8.** Strong KS-$(M, Z)$-space hardness of SPACE-8, 16, 24 and 32 and SPNbox-8, 16, 24 and 32 in known/chosen space attacks

### 4.6 On (Partial) Target Plaintext for Weak Space Hardness

So far we assume that a plaintext is randomly drawn. However, the adversary might have the (partial) knowledge of a plaintext, e.g. the header of a file and the format-fixed encryption cases.

Let us estimate the security when the adversary has $z$-bit ($z \leq n$) information about the given plaintext in advance. In the known-space attack, since the adversary is not able to choose the entries of tables, the advantage in this setting is same as that in the randomly-drawn plaintext setting. In the chosen-space setting, the adversary is able to know inputs of some tables in advance. If the inputs of tables in first $y$ rounds is known, it is weak CS-$(X, -\log_2((M/T)^{t(R-y)}))$-space hardness, where $y$ depends on the $z$ and constructions. In the adaptively-chosen space setting, since the plaintext space is reduced to $2^{128-z}$, we have ACS-$(M, -\log_2(N \cdot 2^{-128+z} + (1 - N \cdot 2^{-128+z})(M/T)^{tR})$-space hardness.

### 4.7 Strong Space Hardness

Next, we show bounds for the strong $(M, Z)$-space hardness in known-, chosen- and adaptively-chosen space attacks.

**Known- and Chosen-Space Attack.** To begin with, we give the following lemma.

**Lemma 3.** *Given any space of size $M$, the expected number of the computable pairs is $2^n \cdot (M/T)^{tR}$.*

*Proof.* According to Theorem 1, given space of size $M$, a randomly-drawn plaintext can be computed with the probability $(M/T)^{tR}$ or less. It holds in any set of known/chosen-space of size $M$. Here, the entire space of the plaintext is $2^n$. □

From Lemma 3, the probability to find a valid pairs with known/chosen space of size $M$ is information-theoretically upper bounded by $2^n \cdot (M/T)^{tR}$. We prove strong KP- and CP- $(M, -\log_2(2^n \cdot (M/T)^{tR}))$-space hardness. For example, in SPNbox-16, given $M = T/4$ space, the success probability is upper bounded by $2^{-32}$ $(= 2^{128} \cdot (1/4)^{8 \cdot 10})$.

**Adaptively-Chosen Space Attack.** In this setting, the adversary has full access to execution environment at any time. Thus, he easily obtain a valid pair of plaintext and ciphertext by adaptively accessing inputs and outputs of each table $tR$ times. Therefore, we can not ensure strong space hardness in this setting.

### 4.8 Tradeoffs Between Strong Space Hardness and Time Complexity

In the previous subsection, we have obtained the upper bound of the probability to find a valid pair of plaintexts and ciphertexts given known- and chosen-space

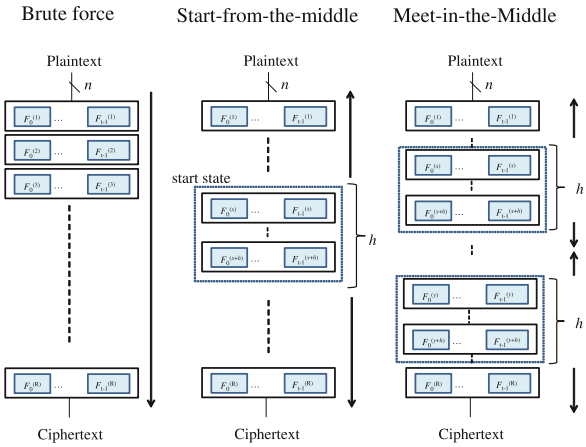


**Fig. 9.** Three types of attacks for strong space hardness

of size $M$. Here, we try to figure out how much time complexity is necessary to find the pair and reveal the tradeoff between the success probability and time complexity. In the multi-table setting, we assume $\#F_0^{(1)} = \#F_1^{(1)} = \ldots = \#F_{t-1}^{(R)}$ which is the optimal case with respect to the success probability. We consider following three types of attacks as shown in Fig. 9.

**Brute Force Attack.** The adversary simply tries to encrypt $2^b$ plaintexts with the given space of size $M$. The time complexity is estimated as $2^b$ for $b \leq n$ and the success probability is $2^b \cdot (M/T)^{tR}$. If $b = n$, the probability becomes the upper bounded value of Lemma 3.

**Start-from-the-Middle Attack.** Assume that if input values of all tables in consecutive $h$ rounds are chosen, then the $n$-bit internal state are determined. We call such states in $r$ rounds start states. We prepare a start state, and then check whether a pair of the plaintext and the ciphertext is computed from the start state through the remaining $(R-h)$ rounds with the given space of size $M$. The number of possible state states is estimated $(\#F)^{th} = 2^n \cdot (\#F/2^{n_{in}})^{th} = 2^n \cdot (M/T)^{th}$. The time complexity is estimated as $2^b$ $(\leq 2^n \cdot (M/T)^{th})$ and the success probability is $2^b \cdot (M/T)^{t(R-h)}$.

**Meet-in-the-Middle Attack.** We start with two start states in the different locations, and mount the meet-in-the-middle approach. In particular, we check whether two states match in the middle rounds, and a pair of the plaintext and the ciphertext is computed from the start states through the $(R-2h)$ rounds. The number of possible start states is estimated $2 \cdot (\#F)^{th} = 2^{(n+1)} \cdot (\#F/2^{n_{in}})^{th} = 2^{n+1} \cdot (M/T)^{th}$. The time complexity is estimated as $2^b$ $(\leq 2^n \cdot (M/T)^{th})$ and the success probability is $2^{2b-n} \cdot (M/T)^{t(R-2h)}$.

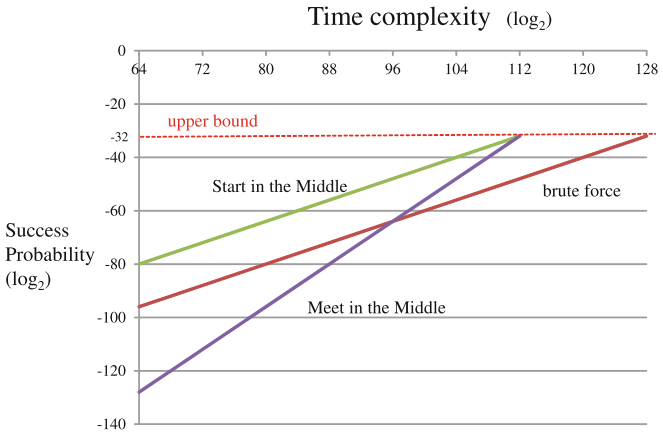


**Fig. 10.** Tradeoffs between time complexity and strong $(T/4, Z)$-space hardness of SPNbox-16 in known/chosen space attacks

**Table 1.** Summary of bounds for weak/strong space hardness against known-, chosen- and adaptively-chosen space attacks

| Known/Chosen-space attack | |
|---|---|
| Weak space hardness | $(M, -\log_2((M/T)^{tR}))$ |
| Strong space hardness | $(M, -\log_2(2^n \cdot (M/T)^{tR}))$ |
| Adaptively-chosen space attack | |
| Weak space hardness | $(M, -\log_2(2^{-128+N} + (M/T)^{tR})$ |
| $N = \lceil log_{e_{in}}(1 - M/T)/tR \rceil$, where $e_{in} = (2^{n_{in}} - 1)/2^{n_{in}}$ | |

**Evaluation.** Fig. 10 shows the trade off between time complexity and strong KS-$(T/4, Z)$-space hardness of SPNbox-16. As mentioned in Sect. 4.7, given $T/4$ space, the success probability is upper bounded by $2^{-32}$ $(= 2^{128} \cdot (1/4)^{8 \cdot 10})$. In our evaluations, in order to achieve it, it requires at most time complexity of $2^{112} (= 2^{128} \cdot (1/4)^{8 \cdot 1})$ by the meet-in-the-middle approach and the start-from-the-middle approach. If adversary's time complexity is restricted, the success probability decreases depending on the time complexity. If time complexity is $2^{80}$ or $2^{64}$, the probability is estimated as $2^{-64}$ $(=2^{80} \cdot 2^{-144})$ or $2^{-80}$ $(=2^{64} \cdot 2^{-144})$ by the start-from-the-middle attack.

### 4.9 Summary of Space Hardness

Table 1 provides a summary of weak and strong space hardness of the target construction in known-, chosen- and adaptively-chosen space attacks. It shows the upper bounds of the success probability against each attack, given space of size $M$; or, in other words, lower bounds for the required space with respect to the success probability of $2^{-Z}$.

Table 2 shows the lower bounds of the required space with respect to success probabilities of $2^{-64}$ and $2^{-128}$ of SPACE-8,-16,-24 and -32 and SPNbox-8,-16,-24 and -32. These results update the evaluations of SPACE-8,-16,-24 and -32 as weak KP-$(T/2^{0.44}, 128)$, $(T/2, 128)$, $(T/2, 128)$ and $(T/2, 128)$-space hardness, while previous results claim weak KP-$(T/4, 128)$-space hardness [9]. All variants SPNbox-8,-16,-24 and -32 achieve weak $(T/4, 64)$-space hardness in known, chosen and adaptively-chosen space attacks, which is a reasonable security level for practical applications. Also, all variants achieve strong $(T/2.3, 64)$ to $(T/32, 64)$-space hardness in known/chosen space attacks.

### 4.10 Advanced Side Channel Attacks

**Differential Computation Analysis.** Bos et al. proposed a new class of side channel attacks called differential computation analysis [11]. This attack exploits memory access patterns during the software execution of whitebox AES [15, 24, 40] with the aid of a binary instrumentation framework such as PIN and Valgrind. Since the software execution traces contain time demarcated physical addresses of memory locations being read/written into, they essentially leak the values of the inputs to the various tables accessed, and can be used as side-channel information to extract the key.

This attack basically utilizes the fact that each table depends on only a fraction of the key, e.g. 8 and 16 bits of key [15, 24, 40]. A small part of the key is efficiently extracted using side-channel leakages. On the other hand, any table of SPNbox contains full 128-bit key information. Thus, even if the adversary can fully monitor the memory access patterns for the target key-dependent table, there are $2^{128}$ possible candidates of corresponding memory access patterns for

**Table 2.** Comparison of SPACE, SPNbox: Lower bounds of the required space with respected to the success probability $2^{-64}$ and $2^{-128}$

| cipher | $T$ | Weak Space hardness | | | | Strong Space hardness | |
|---|---|---|---|---|---|---|---|
| | | $Z = 64$ | | $Z = 128$ | | $Z = 64$ | $Z = 128$ |
| | | KS/CS | ACS | KS/CS | ACS | KS/CS | KS/CS |
| SPACE-8 [9] | 3.84 KB | $T/2^{0.22}$ | $T/2^{0.22}$ | $T/2^{0.44}$ | $T/2^{0.44}$ | $T/2^{0.64}$ | $T/2^{0.86}$ |
| SPACE-16 [9] | 918 KB | $T/2^{0.5}$ | $T/2^{0.5}$ | $T/2$ | - | $T/2^{1.5}$ | $T/2^{2}$ |
| SPACE-24 [9] | 218 MB | $T/2^{0.5}$ | $T/2^{0.5}$ | $T/2$ | - | $T/2^{1.5}$ | $T/2^{2}$ |
| SPACE-32 [9] | 51.5 GB | $T/2^{0.5}$ | $T/2^{0.5}$ | $T/2$ | - | $T/2^{1.5}$ | $T/2^{2}$ |
| SPNbox-8 | 256 B | $T/2^{0.40}$ | $T/2^{0.40}$ | $T/2^{0.80}$ | $T/2^{0.80}$ | $T/2^{1.20}$ | $T/2^{1.60}$ |
| SPNbox-16 | 132 KB | $T/2^{0.81}$ | $T/2^{0.81}$ | $T/2^{1.61}$ | - | $T/2^{2.40}$ | $T/2^{3.20}$ |
| SPNbox-24 | 50.3 MB | $T/2^{1.28}$ | $T/2^{1.28}$ | $T/2^{2.57}$ | - | $T/2^{3.68}$ | $T/2^{4.96}$ |
| SPNbox-32 | 17.2 GB | $T/2^{1.60}$ | $T/2^{1.60}$ | $T/2^{3.20}$ | - | $T/2^{4.80}$ | $T/2^{6.40}$ |

KS: Known-space attack
CP : Chosen-space attack
ACS: Adaptive-chosen-space Attack

each key value. Therefore, a differential computational attack on SPNbox is computationally infeasible.

**Differential Fault Attacks.** Sanfelix et al. propose a differential fault attack on whitebox AES and DES [35]. This attack modifies the specific byte position of internal states by injecting a fault. In the case of the AES, the fault injection targets the MixColumn operation in the 9-th round.

The tables of SPNbox compose of small block ciphers, and the internals of the small block ciphers are inaccessible in whitebox setting. Thus, any fault injection attack reduces to a differential attack on a small block cipher in the blackbox setting. Since the underlying cipher is secure against a differential attack in the blackbox setting as estimated in Sect. 3.2, SPNbox is secure against differential fault attacks.

## 5 Efficient Software Implementations

### 5.1 Setting

In this section, we discuss implementation characteristics of the SPNbox family of block ciphers. We also present experimental measurements based on our optimised high-performance software implementations and compare them to equivalent instances of the SPACE family of whitebox ciphers proposed at CCS 2015 [9]. Altogether, this provides a comprehensive implementation study of all proposed variants both in the blackbox and the whitebox setting. As target platforms for the server-side, we chose the recent Skylake generation of Intel microprocessors which support the AES-NI instruction set [19] and SSE instructions up to AVX2. As a mobile platform, we use the ARMv8 (AArch64) microarchitecture with NEON instructions.

For the blackbox implementations, we specifically focus on constant-time implementations without key-dependent table lookups on recent Intel platforms. Whenever possible, we realise the small block ciphers with AES-NI instructions.

For the whitebox implementations, both on Intel and ARM, the small block ciphers are implemented as table lookups, while the linear mixing of the table lookups is implemented using AVX2 (Intel) and NEON (ARMv8) instructions.

### 5.2 Implementation Characteristics of SPNbox

The SPNbox ciphers can efficiently utilize the parallelism offered by both standard SIMD and the AES instructions on contemporary microprocessors. With block sizes of $n = 128$ or $n = 120$ bit, one block fits naturally in the 128/256-bit SSE/AVX registers on Intel, or the 128-bit NEON registers on ARMv8. Additionally, the parallel and independent application of the S-boxes $S_{n_{in}}$, realised by the small internal block ciphers, offers opportunities for exploiting parallelism, both inside one block and across blocks of a longer message.

**In the Black Box.** In the blackbox setting on Intel platforms, the small block ciphers are implemented in a round-based fashion using the AES-NI instructions for the individual transformations. The composition of $\mathsf{MC}_{n_{in}} \circ \mathsf{SB}$ can be realised by first using the `pshufb` instruction reordering the bytes of the state equivalent to inverse ShiftRows, followed by an `aesenc` instruction for one full AES round. For $n_{in} = 32$, this is already sufficient. For $n_{in} = 24, 16$, we note that by construction, the matrices $A_{24}$ and $A_{16}$ are submatrices of $A_{32}$ such that their multiplication with the state corresponds to $(x, y, z, 0) \times A_{32}$ and $(x, y, 0, 0) \times A_{32}$, respectively (the last 8 resp. 16 bits are ignored). We can therefore realize the round function of the small block ciphers by XOR-ing the values $(0, 0, 0, \mathtt{52}_x)$ or $(0, 0, \mathtt{52}_x, \mathtt{52}_x)$ before applying inverse ShiftRows and the AES round, with $\mathtt{52}_x$ being the inverse of 0 through the AES S-box. This allows the efficient re-use of Intel's AES-NI instructions also for smaller block sizes. For $n_{in} = 8$, the linear mixing step is the identity mapping, so can be omitted.

For the implementation of the linear layer $\theta$ in the outer rounds, it is beneficial to re-organise the internal state such that the $i$-th S-boxes of multiple message blocks are collected in one 128-bit register. This allows an efficient parallel execution of the finite field arithmetic, which vastly outweighs the overhead imposed by the input and output conversion to and from this format.

Additionally, on the Skylake platform, the AES round function has a latency of 4 cycles with a throughput of 1. Altogether, this implies that in order to both fully utilize the AES-NI instruction pipeline and fill the SSE/AVX registers for SIMD operations, our implementations for $n_{in} = 32, 24, 16, 8$ process 8/4/8/16 consecutive blocks at a time, respectively (which is possible in any parallelizable mode, in particular ECB or CTR). By reordering the round keys accordingly, the implementation of the internal block ciphers can remain unchanged.

*Efficient and Constant-Time Parallel Finite Field Arithmetic.* Since we explicitly aim for constant-time implementations in the black box, the conditional polynomial reduction has to be carried out without branching. For this, we employ an optimized variant of the technique introduced in [25], which allows a simultaneous doubling of 4 elements of $\mathrm{GF}(2^{32})$ and $\mathrm{GF}(2^{24})$, or 8 elements of $\mathrm{GF}(2^{16})$ or 16 elements of $\mathrm{GF}(2^8)$ with just four instructions with a latency of 3 and a throughput of 1.

The in-place multiplication by two of register `%xmm0` can be implemented in constant-time as follows:

```
vpcmpgtd  MSB4_M, %xmm0, %xmm1
vpslld    $1, %xmm0, %xmm0
vpand     REDPOLY4_M, %xmm1, %xmm1
vpxor     %xmm0, %xmm1, %xmm0
```

with `MSB4_M` containing four 32-bit copies of the value $\mathtt{7fffffff}_x$, and `REDPOLY4_M` containing four 32-bit copies of the reduction polynomial, i.e. $\mathtt{8d}_x$.

**In the White Box.** In the whitebox setting, the small block ciphers $S_{n_{in}}$ are implemented as lookup tables of size $n_{in} \cdot 2^{n_{in}}$ bytes. The linear layer $\theta$ of SPNbox is then implemented on top of these table lookups using AVX (Intel) or NEON instructions (ARMv8).

Again, we found it beneficial to re-organize the state to collect the $i$-th S-boxes of consecutive blocks in one SSE/NEON register for a SIMD execution of the finite field arithmetic. Compared to SPACE, we have 4,5,8 and 16 parallel independent table lookups in SPNbox-32,24,16 and 8, respectively. Since memory-XMM register transfers have a throughput of 0.5 on Skylake, two of these independent table lookups can be scheduled per cycle on Intel platforms. This has to be contrasted to the situation in the serial round function of SPACE, where no simultaneous table lookups were possible.

On ARM, the smaller caches and slower memory interface imply that lookups in larger tables tend to be relatively more expensive than on Intel platforms.

### 5.3 Performance Measurements

We provide performance measurements for SPNbox and SPACE in both the blackbox and the whitebox setting for the encryption of messages of length 2048 bytes. For the Intel platform, all measurements were taken on a single core of an Intel Core i7-6700 CPU at 3400 MHz with Turbo Boost and hyperthreading disabled, and averaged over 100000 repetitions, processing one message at a time. For the ARMv8 platform, a single Cortex-A57 core at 2100 MHz of a Samsung Exynos 7420 CPU as shipped in a Samsung Galaxy S6 mobile phone was used.

Our findings are summarised in Table 3 and Fig. 11 for the blackbox setting; and Table 4 for the whitebox setting. The whitebox performance is further illustrated in Fig. 12 (grouped by table size) and Fig. 13 (grouped by platform). All performance figures are given in cycles per byte (cpb).

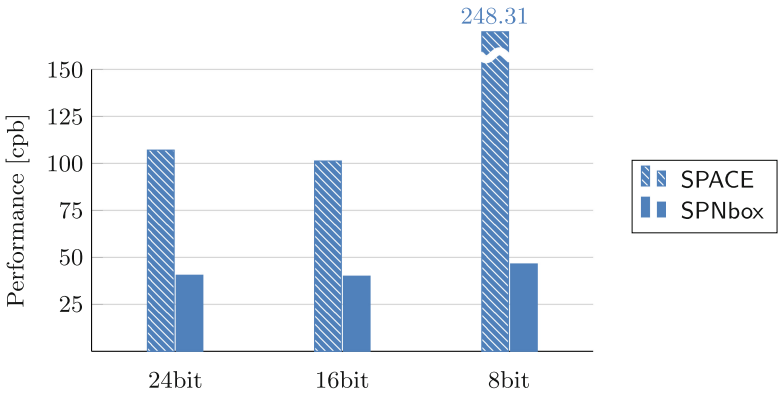


**Fig. 11.** Constant-time blackbox performance of SPACE and SPNbox on Intel Skylake platform for various table sizes in cycles per byte (lower is better).

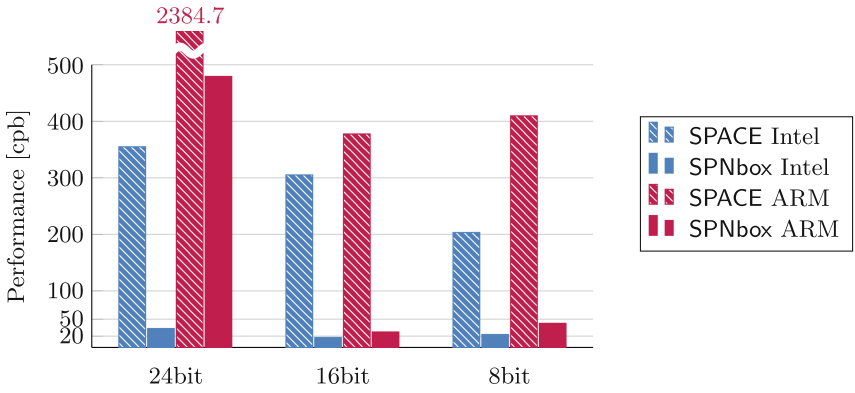


**Fig. 12.** Whitebox performance of SPACE and SPNbox on Intel Skylake and ARMv8 platforms for various table sizes in cycles per byte (lower is better).

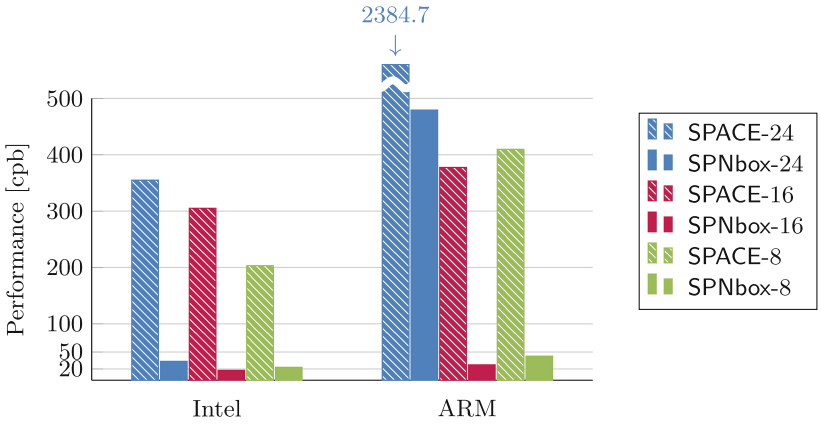


**Fig. 13.** Whitebox performance of SPACE and SPNbox on Intel Skylake and ARMv8 platforms for various table sizes in cycles per byte (lower is better).

**Discussion.** The blackbox constant-time implementation results in Table 3 indicate that for each variant with comparable space hardness, the SPNbox ciphers offer significantly increased performance compared to SPACE. Somewhat interestingly, the largest improvement (factor 4.5 speed-up) is obtained for the 32-bit variant offering the highest level of space hardness. This is due to inherent construction differences: While SPACE always uses the full AES transform and its performance is only affected by the number of Feistel rounds, SPNbox needs more and more rounds in its internal block ciphers to ensure sufficient key mixing when the block sizes becomes smaller. Additionally, the use of the AES round transformation implies increasing overhead with smaller block sizes, since increasing parts of the state are unused. For $n_{in} = 8$, the decrease in performance is caused by the heavy $16 \times 16$ MDS diffusion layer over $\mathrm{GF}(2^8)$.

**Table 3.** Software performance of the SPNbox and SPACE cipher families on the Intel Skylake platform in the blackbox setting. Numbers are given in cycles per byte (cpb).

| Algorithm | Rounds (outer) | Rounds (inner) | Performance [cpb] |
|---|---|---|---|
| SPNbox-32 | 10 | 16 | 15.09 |
| SPNbox-24 | 10 | 20 | 40.48 |
| SPNbox-16 | 10 | 32 | 39.98 |
| SPNbox-8 | 10 | 64 | 46.49 |
| SPACE-32 | 128 | 10 | 101.02 |
| SPACE-24 | 128 | 10 | 107.01 |
| SPACE-16 | 128 | 10 | 101.21 |
| SPACE-8 | 300 | 10 | 248.31 |

**Table 4.** Software performance of the SPNbox and SPACE cipher families in the whitebox setting on Intel Skylake and ARMv8 platforms. Numbers are given in cycles per byte (cpb).

| Algorithm | Rounds (outer) | Table size | Performance Intel | [cpb] ARM |
|---|---|---|---|---|
| SPNbox-32 | 10 | 17.2 GB | 184.56 | — |
| SPNbox-24 | 10 | 50.3 MB | 33.48 | 479.38 |
| SPNbox-16 | 10 | 132 KB | 17.59 | 27.37 |
| SPNbox-8 | 10 | 256 B | 22.93 | 42.66 |
| SPACE-32 | 128 | 51.5 GB | 5535.01 | — |
| SPACE-24 | 128 | 218 MB | 354.86 | 2384.74 |
| SPACE-16 | 128 | 918 KB | 305.11 | 377.51 |
| SPACE-8 | 300 | 3.84 KB | 203.19 | 409.57 |

Regarding the performance of SPACE, our results largely confirm the estimation of $R$ cpb for $R$ rounds on an AES-NI platform provided in [9].

Also in the whitebox setting, SPNbox significantly outperforms SPACE for all variants, on both Intel and ARM platforms. One observes that any increases in pure lookup performance due to smaller table size is increasingly compensated for by the heavier linear MDS layers. The surprisingly good performance of SPNbox-32 can to some extent be attributed to the fact that our test platform had 16 GB of memory available.

Comparing the blackbox to the whitebox performances of each variant of SPNbox, it becomes apparent that from $n_{in} = 24$ and smaller, table-based implementations outperform round-based implementations. The latter, however, offer constant timing behaviour. Further optimizations of the constant-time implementations also remain possible.

Summarising, the constant-time blackbox performance of the proposed SPNbox ciphers outperforms the SPACE variants by factors of 2.5 to 6.5. In the

whitebox setting, the new SPNbox ciphers offer performance improvements by factors of 8 to 18 (on Intel) and 5 to 13 (on ARM) over SPACE, as illustrated in Figs. 12 and 13.

## 6 Conclusion and Outlook

In this paper, we proposed SPNbox, a new family of space-hard block ciphers, which significantly improves upon the SPACE ciphers. Employing an SPN-type design with efficient constant-time small block ciphers, the resulting parallelization opportunities allow significantly faster implementations both in the black box and in the white box. Instances of SPNbox achieve speed-ups of up to 6.5 times in the black box and up to 18 times in the whitebox setting, while offering comparable space hardness. Moreover, we formalized the security models of space hardness which are classified with respect to the adversary's abilities. We proved security bounds of space hardness in all adversarial models. We then applied this analysis to SPNbox, showing that SPNbox offers sufficiently high levels of space hardness in each adversary model.

Our work also raises a couple of open research questions and directions. Concerning the design of the small internal block ciphers, there seems to be an efficiency bottleneck regarding the key mixing: The smaller the block size, the more rounds are needed to avoid meet-in-the-middle attacks, which limits their efficiency. This raises the question of how to build more efficient block ciphers with very small block lengths and a relatively large key. Especially, fast key mixing and efficient key scheduling functions for small block ciphers are essentially unknown.

A possible solution for this efficiency problem is to use table lookups for secret S-boxes. This however introduces side-channel issues with key-dependent lookups, motivating further research into how to construct secret S-boxes of various sizes with efficient constant-time implementations.

## References

1. Adobe Systems Incorporated. Adobe Primetime Technical Primer for Operators (2014)
2. Akamai Technologies. Securing Cloud-Based Workflows for Premium Content (2014)
3. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 Proposal: ECHO. Submission to NIST (2009)
4. Barreto, P., Rijmen, V.: The Anubis Block Cipher. Submission to the NESSIE Project (2000)
5. Barreto, P., Rijmen, V.: The Khazad Legacy-level Block Cipher. Submission to the NESSIE Project (2000)
6. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a white box AES implementation. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 227–240. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30564-4_16

7. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 63–84. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45611-8_4
8. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. J. Cryptology **23**(4), 505–518 (2010)
9. Bogdanov, A., Isobe, T.: White-box cryptography revisited: space-hard ciphers. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1058–1069. ACM (2015)
10. Borghoff, J., Knudsen, L.R., Leander, G., Thomsen, S.S.: Slender-set differential cryptanalysis. J. Cryptology **26**(1), 11–38 (2013)
11. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: hiding your white-box designs is not enough. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 215–236. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53140-2_11
12. Bringer, J., Chabanne, H., Dottax, E.: White box cryptography: another attempt. IACR Cryptology ePrint Archive 2006:468 (2006)
13. Chong, K.-M.: The arithmetic mean-geometric mean inequality: a new proof. Math. Mag. **49**(2), 87–88 (1976)
14. Chow, S., Eisen, P., Johnson, H., Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 1–15. Springer, Heidelberg (2003). doi:10.1007/978-3-540-44993-5_1
15. Chow, S., Eisen, P., Johnson, H., Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003). doi:10.1007/3-540-36492-7_17
16. Delerablée, C., Lepoint, T., Paillier, P., Rivain, M.: White-box security notions for symmetric encryption schemes. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 247–264. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43414-7_13
17. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006). doi:10.1007/11681878_11
18. Gruss, D., Spreitzer, R., Mangard, S.: Cache template attacks: automating attacks on inclusive last-level caches. In: 24th USENIX Security Symposium, USENIX Security 15, pp. 897–912. USENIX Association (2015)
19. Gueron, S.: Intel Advanced Encryption Standard (AES) Instructions Set. Intel white paper, September 2012
20. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold boot attacks on encryption keys. In: Proceedings of the 17th USENIX Security Symposium, pp. 45–60. USENIX Association (2008)
21. Hawkes, P., O'Connor, L.: XOR and Non-XOR differential probabilities. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 272–285. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_19
22. Irazoqui, G., Eisenbarth, T., Sunar, B.: S$a: A shared cache attack that works across cores and defies VM sandboxing - and its application to AES. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, pp. 591–604. IEEE Computer Society (2015)

23. Irazoqui, G., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a minute! A fast, cross-VM attack on AES. In: Stavrou, A., Bos, H., Portokalidis, G. (eds.) RAID 2014. LNCS, vol. 8688, pp. 299–319. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11379-1_15
24. Karroumi, M.: Protecting white-box AES with dual ciphers. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 278–291. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24209-0_19
25. Käsper, E., Schwabe, P.: Faster and timing-attack resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 1–17. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04138-9_1
26. Lepoint, T., Rivain, M., Mulder, Y., Roelse, P., Preneel, B.: Two attacks on a white-box AES implementation. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 265–285. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43414-7_14
27. Link, H.E., Neumann, W.D.: Clarifying obfuscation: improving the security of white-box DES. In: International Symposium on Information Technology: Coding and Computing (ITCC 2005), vol. 1, pp. 679–684 (2005)
28. Minaud, B., Derbez, P., Fouque, P.-A., Karpman, P.: Key-recovery attacks on ASASA. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 3–27. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_1
29. Workgroup Mobey, H.C.E., Forum. The Host Card Emulation in Payments: Options for Financial Institutions (2014)
30. Mulder, Y., Roelse, P., Preneel, B.: Cryptanalysis of the xiao – lai white-box AES implementation. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 34–49. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35999-6_3
31. De Mulder, Y., Wyseur, B., Preneel, B.: Cryptanalysis of a perturbated white-box AES implementation. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 292–310. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17401-8_21
32. National Institute of Standards and Technology. Recommendation for Key Derivation Using Pseudorandom Functions. NIST Special Publication (SP) 800–108 (2009)
33. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication 202 (2015)
34. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, pp. 199–212. ACM (2009)
35. Sanfelix, E., Mune, C., de Haas, J.: Unboxing the white-box practical attacks against obfuscated ciphers. In: Black Hat Europe 2015 (2015)
36. Sim, S.M., Khoo, K., Oggier, F., Peyrin, T.: Lightweight MDS involution matrices. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 471–493. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48116-5_23
37. Alliance, S.C., Paper, W.: Host Card Emulation (HCE) 101 (2014)
38. Tiessen, T., Knudsen, L.R., Kölbl, S., Lauridsen, M.M.: Security of the AES with a secret S-Box. In: Leander, G. (ed.) Fast Software Encryption. LNCS, vol. 9054, pp. 175–189. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48116-5_9

39. Wyseur, B., Michiels, W., Gorissen, P., Preneel, B.: Cryptanalysis of white-box DES implementations with arbitrary external encodings. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 264–277. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77360-3_17
40. Xiao, Y., Lai, X.: A secure implementation of white-box AES. In: 2nd International Conference on Computer Science and its Applications (CSA2009) (2009)
41. Yarom, Y., Falkner, K.: FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In: Proceedings of the 23rd USENIX Security Symposium, pp. 719–732. USENIX Association (2014)