Key Generators Can Strengthen Block Ciphers

Michael Stephen Fiske
Aemea Institute
San Francisco, CA, 94129, USA
mf@aemea.org

Abstract: The notion of *key generators* is introduced to symmetric cryptography. Key generators help eliminate the dependence of a block cipher's security on a single, static key. If one of the dynamic keys is leaked to the adversary, then this compromise does not reveal future keys and prior keys used by the block cipher to encrypt distinct blocks of plaintext. A practical, key generator updating algorithm is provided that enhances the cryptographic strength of block ciphers and in particular the AES block cipher.

Keywords: AES, block cipher, complexity, dynamic key, key generator, regularity, static key assumption

1 Introduction

Typically, cryptographic methods [1, 2, 3, 4, 5] imply or use a static key throughout the entire execution of the encryption algorithm. (Herein *algorithm* refers to any method that can be implemented with a Turing machine [6].) In symmetric cryptography, the encryption and decryption use the same private key. For example, a block cipher algorithm $\mathcal{E}_{\mathcal{A}} : \{0,1\}^m \times \{0,1\}^\kappa \rightarrow \{0,1\}^m$ uses an κ-bit private key K as a parameter and encrypts an m-bit block of plaintext $\mathcal{M}$, denoted as $\mathcal{E}_{\mathcal{A}}(\mathcal{M}, K)$. The block cipher's key space $\{0,1\}^\kappa$ has size 2^κ. The block cipher's message space $\{0,1\}^m$ has size 2^m. The decryption algorithm $\mathcal{D}_{\mathcal{A}} : \{0,1\}^m \times \{0,1\}^\kappa \rightarrow \{0,1\}^m$ has the inverse property that $\mathcal{D}_{\mathcal{A}}(\mathcal{E}_{\mathcal{A}}(x, K), K) = x$ for every plaintext $x \in \{0,1\}^m$ and every key $K \in \{0,1\}^\kappa$.

AES is a popular block cipher with block size 128 bits, standardized by NIST [7, 8]. In recent years, various attacks on the AES cipher have demonstrated some weaknesses. Practical oracle padded attack [9] can capture the plaintext from ciphertext that has been encrypted by AES. At least part of the weakness in AES is due to slow diffusion in the key scheduling [10, 11, 12]. A feasible attack on AES-256 has been demonstrated with only one fewer round, 13 rounds instead of 14, in the key schedule [13]. In [14], a cache-timing attack captured the static key in one day using 2×10^8 $(\texttt{plaintext}, \texttt{ciphertext}, \texttt{time})$ triples. Additional attacks were also recently discovered [15, 16, 17, 18].

In prior methods, the *static key assumption* is implied by some attacks (cited in the previous paragraph, including the cache-timing attack) whose goal is to capture the key. In the algebraic embedding of AES in BES, the static key assumption is further implied by the counting [19, 20] of the number of equations that the AES encryption satisfies. In another example, [21] demonstrates a generic attack on all block ciphers such that the time complexity for finding a static key of size k bits is $2^k(1-\epsilon)$, where $\epsilon > 0$.

This paper's primary contribution introduces *key generators* to symmetric cryptography. A key generator is defined as a sequence $\Gamma : \mathbb{N} \rightarrow \{0,1\}^n$. Key generators help eliminate the dependence of a block cipher's security on a single, static key. In our algorithms 2 and 3, a new dynamic key is derived for each block of plaintext from the key generator so that the block cipher no longer depends upon a single, static key. Even if one of the dynamic keys is leaked to the adversary, then this compromise does not reveal prior keys and future keys used by the block cipher to encrypt distinct blocks of plaintext.

A second contribution defines a one-way preimage function, based on our introduction of concrete complexity, and uses these functions to define a general key generator updating algorithm 1 and cryptographic algorithm 2. Algorithms 1 and 2 help strengthen the underlying block cipher: when a one-way preimage hash function with digest size q satisfies our regularity condition and helps implement our key generator updating algorithm, the compromise of prior keys and future keys in algorithm 2 requires a preimage attack with complexity at least 2^{n-q}, where $n > q$. For appropriate values of q and n, this complexity 2^{n-q} can be substantially greater than the size of the keyspace, used by the underlying block cipher.

Lastly, algorithm 3 contributes a practical instantiation of algorithm 2. Speed tests, presented in section 7, demonstrate the feasibility of key generators in modern cryptographic applications.

ISBN: 978-1-61804-297-2

2 Key Generator Updating and Dynamic Key Derivation

Key generator updating requires Alice and Bob agreeing upon the next sequence element $\Gamma(i)$ of key generator Γ. Even though an uncountable number of key generators are Turing incomputable, herein our attention is on computable key generators because computability helps simplify the coordination of updating between Alice and Bob.

Our *one-way preimage hash functions* are not intended for message authentication. Consequently, in section 4, our formal definition does not include collision resistance. The reason for this is that the hash digest of Ψ in algorithm 2 is not readily available to Eve and that a collision does not provide a feasible method for Eve computing prior and future keys, as further explained in remark 22. With this in mind, the key generator updating and cryptography algorithms are presented first.

In key generator updating algorithm 1, Φ is a one-way preimage hash function with digest size q. $\Gamma : \mathbb{N} \to \{0,1\}^n$ is a key generator such that $n > q$. The jth bit of $\Gamma(i)$ is $\Gamma_{i,j}$. Alice and Bob can establish a shared element $\Gamma(0)$, by executing a signed, Diffie-Hellman-Merkle exchange with elliptic curves [22, 23, 24, 25]. (See Theorem 2.1 in [24] to establish shared secrets with $n > 256$.)

Algorithm 1 *Key Generator Updating*

Alice and Bob execute a signed, DHM exchange to establish a shared 0th sequence element $\Gamma(0) = \Gamma_{0,0} \ldots \Gamma_{0,n-1}$.

Initialize $i = 0$
while(next element $\Gamma(i+1)$ is requested)
{
  $(\Gamma_{i+1,0} \ldots \Gamma_{i+1,q-1}) = \Phi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,q-1})$
  Set $\Gamma_{i+1,j} = \Gamma_{i,j}$ for each j such that $q \le j \le n-1$
  Increment i
}

Algorithm 1 is designed to generate a high dimensional orbit on the first q bits $\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,q-1}$, induced by the avalanche properties [26] of function Φ; to keep the remaining $n-q$ bits invariant for all i; and to assure that no information from the last $n-q$ bits contributes to the orbit of the first q bits. In a typical case, the adversary Eve never has access to any bits of Alice's $\Gamma(i)$. This is analogous to Eve not having access to any bits of Alice's static key, used in traditional, symmetric cryptography.

Algorithm 2 derives a dynamic key K_i for block cipher $\mathcal{A}$ from each sequence element $\Gamma(i)$ of the key generator. Let Ψ be a one-way preimage hash function whose digest size is r bits. Ψ hashes a concatenation of the dynamic part $\Gamma_{i,0}, \Gamma_{i,1}, \ldots \Gamma_{i,q-1}$ of $\Gamma(i)$ and the invariant part $\Gamma_{i,q-1}, \ldots \Gamma_{i,n-1}$ in order to derive a distinct key K_i for each block that is encrypted. The expression $\mathcal{E}_{\mathcal{A}}(\mathcal{M}, K)$ represents block cipher $\mathcal{A}$ encrypting plaintext block $\mathcal{M}$ with key K, and $\mathcal{D}_{\mathcal{A}}(\mathcal{C}, K)$ represents block cipher $\mathcal{A}$ decrypting ciphertext $\mathcal{C}$ with key K. The key size $|K|$ of the block cipher is κ bits and satisfies $\kappa \le r$. Define the projection map $\pi_\kappa : \{0,1\}^r \to \{0,1\}^\kappa$ where $\pi_\kappa(x_1, x_2, \ldots, x_r) = (x_1, x_2, \ldots, x_\kappa)$.

Algorithm 2 *Block Cipher $\mathcal{A}$ uses Dynamic Keys*

Alice's Encryption Algorithm:

Alice executes with Bob a signed, DHM exchange to share secrets $\Gamma(0)$ and $\mathcal{C}_{-1}$

Initialize $i = 0$
while(more plaintext $\mathcal{M}_i$ to encrypt)
{
  Dynamic key $K_i = \pi_\kappa \circ \Psi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,n-1})$
  Encrypt $\mathcal{C}_i = \mathcal{E}_{\mathcal{A}}(\mathcal{M}_i \oplus \mathcal{C}_{i-1}, K_i)$
  ($\mathcal{M}_i \oplus \mathcal{C}_{i-1}$ is encrypted with key K_i)
  Algorithm 1 computes element $\Gamma(i+1)$
  Increment i
}

Bob's Decryption Algorithm:

Bob executes with Alice a signed, DHM exchange to share secrets $\Gamma(0)$ and $\mathcal{C}_{-1}$

Initialize $i = 0$
while(more ciphertext $\mathcal{C}_i$ to decrypt)
{
  Dynamic key $K_i = \pi_\kappa \circ \Psi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,n-1})$
  Decrypt $\mathcal{M}_i = \mathcal{C}_{i-1} \oplus \mathcal{D}_{\mathcal{A}}(\mathcal{C}_i, K_i)$
  ($\mathcal{C}_i$ is decrypted with dynamic key K_i)
  Algorithm 1 computes element $\Gamma(i+1)$
  Increment i
}

Note that cipher block chaining is used.

3 AES-128 and SHA-512 Key Generator Updating

A variation of algorithm 2 is described. AES-128 is the block cipher. SHA-512 [27] acts as the one-way hash function Φ that performs the key generator updating and one-way hash Ψ that derives the dynamic key. Two steps of algorithm 2 change slightly:

ISBN: 978-1-61804-297-2

Dynamic key $K_i = \pi_\kappa \circ \Psi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,n-1})$ and Algorithm 1 computes element $\Gamma(i+1)$.

One SHA-512 digest creates four distinct 128-bit keys. The encryption and decryption execution speeds can be increased by performing these two steps only when $i \bmod 4 \equiv 0$. Define function $\Pi : \{0,1,2,3\} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$ as $\Pi(a, (x_0, x_1, \ldots, x_{511})) = (x_{128a}, x_{128a+1}, \ldots, x_{128a+127})$ where $a \in \{0,1,2,3\}$ and $(x_0, x_1, \ldots, x_{511}) \in \{0,1\}^{512}$. Set $n = 1024$ so that for all i and for each $j \in \{512, \ldots, 1023\}$ then $\Gamma_{i,j} = \Gamma_{i+1,j}$. Set $\Phi = \Psi =$ SHA-512.

Algorithm 3 *AES-128 uses a SHA-512 Key Generator Update*

Alice's Encryption Algorithm:

Alice executes with Bob a signed, DHM exchange to share secrets $\Gamma(0)$ and $\mathcal{C}_{-1}$

Initialize $i = 0$
while(more plaintext $\mathcal{M}_i$ to encrypt)
{
  Set $a = i \bmod 4$
  if $(a == 0)$ then compute $\beta_{i/4} = \Psi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,1023})$
  Set dynamic key $K_i = \Pi(a, \beta_{i/4})$
  Encrypt $\mathcal{C}_i = \mathcal{E}_{AES}(\mathcal{M}_i \oplus \mathcal{C}_{i-1}, K_i)$
  If $(a == 0)$ then algorithm 1 computes element $\Gamma(i+1)$ from $\Gamma(i)$
  Increment i
}

Bob's Decryption Algorithm:

Bob executes with Alice a signed, DHM exchange to share secrets $\Gamma(0)$ and $\mathcal{C}_{-1}$

Initialize $i = 0$
while(more ciphertext $\mathcal{C}_i$ to decrypt)
{
  Set $a = i \bmod 4$
  if $(a == 0)$ then compute $\beta_{i/4} = \Psi(\Gamma_{i,0}\, \Gamma_{i,1} \ldots \Gamma_{i,1023})$
  Set dynamic key $K_i = \Pi(a, \beta_{i/4})$
  Decrypt $\mathcal{M}_i = \mathcal{C}_{i-1} \oplus \mathcal{D}_{AES}(\mathcal{C}_i, K_i)$
  If $(a == 0)$ then algorithm 1 computes element $\Gamma(i+1)$ from $\Gamma(i)$
  Increment i
}

Note that cipher block chaining is used.

The use of key generator updating in algorithm 3 should not be confused with the existing block cipher modes of operation such as ECB, CBC or CTR. First, each of these modes still relies on a static key. Even CTR – where $K_i = E_A(\text{nonce} \,||\, i,\, K)$ and the ith block of ciphertext is $C_i = M_i \oplus K$ – relies on the static key K. Second, key generator updating uses values of n for the key generator that can be substantially greater than the block and static key size. That is, usually $n \gg |\mathcal{M}_i|$ and $n \gg \kappa$. For example, in algorithm 3, $n = 1024$, while the key and block size = 128. As explained in section 5, the periodicity of the orbit of dynamic keys produced by a key generator can be substantially greater than 2^κ.

Each of these modes puts an upper bound on the amount of entropy increase, based on the block size or key size. In the case of ECB, no entropy increase occurs. In the case of CBC, the entropy increase is bounded above by the size of the message space. In the case of CTR, the nonce concatenated with the counter i is bounded above by the size of the message space and the resulting key orbit is bounded above by the size of the key space. Since n can be substantially greater than the key or block size, a greater entropy increase can occur with key generator updating. Furthermore, nothing precludes combining key generator updating with the CBC mode or the CTR mode. Both algorithms 2 and 3 show key generator updating combined with the CBC mode.

4 Concrete Complexity and One-Way Preimage Functions

Based on Turing machines, this section introduces concrete complexity and then defines a one-way preimage hash function. The first goal of our new definitions is to avoid the difficulty that asymptotic definitions of complexity cannot model one-way hash functions used in practice. A second longer term goal is to further develop an appropriate framework to characterize one-wayness, by applying powerful tools from dynamical systems to the Turing machine.

As a brief review, a *Turing machine* is a triple $(\mathcal{Q}, \Sigma, \eta)$ where $\mathcal{Q}$ is a finite set of states that does not contain a unique halting state h. When machine execution begins, the machine is in an initial state $s \in \mathcal{Q}$. Σ is a finite alphabet whose symbols are read from and written to a tape $\mathcal{T} : \mathbb{Z} \rightarrow \Sigma$. The alphabet symbol in the kth tape square is $\mathcal{T}(k)$. -1 and $+1$ represent advancing the tape head to the left or right tape square, respectively. η is a *program* function, where $\eta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q} \cup \{h\} \times \Sigma \times \{-1, +1\}$.

For each q in $\mathcal{Q}$ and α in Σ, the instruction $\eta(q, \alpha) = (r, \beta, x)$ specifies how the machine executes one *computational step*. When in state q and

ISBN: 978-1-61804-297-2

reading alphabet symbol α on the tape: the machine jumps to state r. On the tape, the machine replaces alphabet symbol α with symbol β. If $x = -1$ or $x = +1$, then the machine moves its tape head one square to the left or right, respectively, and subsequently reads the symbol in this new square. If $r = h$, the machine reaches the halting state and stops executing.

Definition 4 *Concrete Complexity*

For machine input $u \in \Sigma^*$, let $|u|$ be the length of u. Let $g : \mathbb{N} \to \mathbb{N}$ be a function of $|u|$. Machine $\mathcal{M} = (\mathcal{Q}, \Sigma, \eta)$ has concrete, complexity $\mathcal{C}(g, \sigma, \varrho, |u|)$ if the following three conditions hold: (1) On input u, machine $\mathcal{M}$ takes at least $g(|u|)$ computational steps to halt. (2) $\mathcal{M}$'s alphabet satisfies $|\Sigma| \leq \sigma$. (3) $\mathcal{M}$'s states satisfy $|\mathcal{Q}| \leq \varrho$.

Remark 5

Parameters σ and ϱ impose limits on the size of the Turing machine program η in order to eliminate precomputations (table lookups). Precomputations are assumed to be encoded into η and/or the input u; otherwise, our theoretical definitions won't adequately model security practice.

Remark 6

Observe that *prior complexity definitions depend on the meaning of algorithm*. For any given algorithm, there can be an infinite number of Turing machines that implement the algorithm, where each of these machines have Shannon's `State × Symbol` complexity [28] such that $|\mathcal{Q}||\Sigma| > \varrho\sigma$. The distinction between a machine's implementation of an algorithm and an abstract algorithm can lead to deep subtleties [29, 30, 31]. In [32], a blackbox is constructed with a self-modifying, parallel machine that utilizes quantum randomness; this incomputable method raises more questions about the differences between an algorithm and the "machine" that executes it. Also, see [33].

Remark 7

From a practical perspective, side channel attacks typically exploit the particular machine implementation of an algorithm. (For example, see [14].) This further supports our position that a complexity definition should be based on the machine, not the algorithm.

Informally, $h : \{0,1\}^{<N} \to \{0,1\}^q$ is an (N, σ, ϱ, r) one-way, preimage function if A and B hold:

A. A Turing machine $\mathcal{M}$ exists that on input x outputs $h(x)$ in a feasible number of computational steps.

B. Any probabilistic, Turing machine $\mathcal{P}$ – that is given $y \in \{0,1\}^q$ as input and searches for an inverse image point $x \in h^{-1}(y)$ – only succeeds with exponentially low probability under the following 3 conditions: (1) Turing machine $\mathcal{P}$ has at most σ alphabet symbols. (2) Turing machine $\mathcal{P}$ has at most ϱ states. (3) There is some fixed degree r and each success takes at least $|x|^r$ steps.

In our formal definition, no assumptions are made about collision resistance.

Definition 8 *One-Way Preimage Hash Function*

Let $\sigma, \varrho, r \in \mathbb{N}$. Let $N \in \mathbb{N} \cup \{\omega_0\}$. A function $h : \{0,1\}^{<N} \to \{0,1\}^q$ is called an (N, σ, ϱ, r) one-way, preimage hash function with digest size q if the following two conditions hold:

A. **Easy to evaluate**: There exists a polynomial time Turing machine $\mathcal{A}$ such that $\mathcal{M}(x) = h(x)$ for every $x \in \{0,1\}^{<N}$.

B. **Computationally hard to invert**: For every probabilistic Turing machine $\mathcal{P}$ and every n such that $q \leq n < N$ the probability of the set $\{x \in \{0,1\}^n : \mathcal{P}(h(x)\ 1^n) \in h^{-1}(h(x))$ `and` $\mathcal{P}$ `has concrete complexity` $\mathcal{C}(n^r, \sigma, \varrho, |h(x)\ 1^n|)\ \} \leq 2^{-\frac{n}{2}}$.

The following remarks help clarify definition 8.

Remark 9

ω_0 is the first countably infinite ordinal. When $N = \omega_0$, this implies the domain of h is $\{0,1\}^*$ and in this case definition 8 is asymptotic.

Remark 10

The adversary's machine $\mathcal{P}$ receives $h(x)$ as input and the auxiliary input 1^n which is the binary length of x. The purpose of the auxiliary input 1^n is to eliminate the possibility that a function is speciously considered one-way because machine $\mathcal{P}$ does not have enough time to print its output. For example, the function $h(x) = y$, where $y = \log n$ of the least significant bits of x with $|x| = n$. No machine can find a point of $h^{-1}(y)$ in time polynomial in $|h(x)|$; however, there is a machine which finds a point of $h^{-1}(y)$ in time polynomial in $|x|$.

Remark 11

For our purposes only $n \geq q$ is needed in algorithms 1 and 2. There is some $k < q$ such that the adversary can brute force compute $h(x)$ for every $x \in \{0,1\}^j$ whenever $j \leq k$. The number k obviously depends on the adversary's computational resources.

ISBN: 978-1-61804-297-2

Remark 12
The one-way notion is probabilistic. The definition does not state that it is impossible for the adversary's machine $\mathcal{P}$ to find a point in the inverse image $h^{-1}\big(h(x)\big)$; it says that $\mathcal{P}$ has a probability $\leq 2^{-\frac{n}{2}}$ of finding a point in the inverse image, where the machine takes at least n^r computational steps to find it. Here $g(|u|) = (|u| - q)^r$, where $u = h(x)\ 1^n$. To "succeed", the adversary's machine $\mathcal{P}$ only has to find some point in $h^{-1}\big(h(x)\big)$. $\mathcal{P}$ is not required to find the x that machine $\mathcal{M}$ used. Furthermore, the probability distribution is uniform over the input x and the possible coin tosses of the adversary's machine $\mathcal{P}$.

Remark 13
The intuitive reason for the upper bound $2^{-\frac{n}{2}}$ on the probability stems from the birthday paradox: given a randomly selected digest y, it should be computationally more difficult for Eve to find a preimage point $x \in h^{-1}(y) \cap \{0,1\}^n$, than for Eve to randomly select preimage points $x_1, x_2, \ldots, x_m$, compute $h(x_1), h(x_2), \ldots, h(x_m)$ and search for a collision in $\{h(x_1), h(x_2), \ldots, h(x_m)\}$. The formal reason is based on number theory and dynamical systems tools applied to Turing machines, which is beyond the scope of this paper.

Example 14
Let $\Phi_{512} : \{0,1\}^{<2^{128}} \rightarrow \{0,1\}^{512}$ denote `SHA-512`. For Φ_{512}, $N = 2^{128}$ and $q = 512$. It is unknown whether `SHA-512` satisfies our definition of one-way preimage function, for some values of r, σ and ϱ. In this regard, it is helpful to mention the recent biclique preimage attack [34] on a reduced 50 rounds of Φ_{512}: their preimage complexity estimate of $2^{511.5}$ still supports this possibility and is far beyond today's computing power. In practice, input strings $\geq 2^{128}$ bits do not arise. However, based on prior, typical definitions of one-wayness, `SHA-512` fails to be a one-way hash function because its domain is not $\{0,1\}^*$ and consequently cannot satisfy their asymptotic requirements.

5 Some Analysis of Algorithms 1, 2 and 3

Let $f : X \rightarrow X$ be a function on some topological space X. The orbit of the point $p \in X$ is $\mathcal{O}(p, f) = \{p,\ f(p),\ f \circ f(p),\ \ldots\ f^n(p),\ \ldots\}$. In general, the orbit may be an infinite set. In algorithms 1, 2 and 3, the space $X = \{0,1\}^m$ for some $m \in \mathbb{N}$, so our key orbits and key generator orbits are finite. Point $p \in X$ is a *periodic* point if there exists $j \in \mathbb{N}$ such that $f^j(p) = p$. Point $x \in X$ is *eventually periodic* if there exists $k \in \mathbb{N}$ such that $f^k(x) = p$ and p is a periodic point.

Suppose $f : \{0,1\}^m \rightarrow \{0,1\}^m$ is a function. The pigeonhole principle implies that every point $x \in \{0,1\}^m$ is eventually periodic with period at most 2^m. Each function $f : \{0,1\}^m \rightarrow \{0,1\}^m$ induces an equivalence relation on the set $\{0,1\}^m$ as follows. If x and y are eventually periodic in the same orbit with respect to f, then x and y are called *eventually periodic equivalent*, expressed as $x \underset{f}{\sim} y$. Let $[x]$ denote the equivalence class $\{y \in \{0,1\}^m : x \underset{f}{\sim} y\ \}$.

The *key generator orbit* $\mathcal{O}(\Gamma, \Phi, \mathcal{A}_1) = \{\pi_q \circ \Gamma(i) \in \{0,1\}^q : \Gamma(i)$ `is computed by Algorithm` $1\}$. The *dimension* of the key generator orbit is the number of points in $\mathcal{O}(\Gamma, \Phi, \mathcal{A}_1)$. Also, $\mathcal{A}_2$ and $\mathcal{A}_3$ denote algorithms 2 and 3, respectively.

Definition 15 *Let $\phi : \{0,1\}^{<N} \rightarrow \{0,1\}^q$ be a hash function with digest size q. (No assumption is made about ϕ's one-wayness.) ϕ has a periodic point $p \in \{0,1\}^q$ with period m if m is the smallest, positive integer such that $\phi^m(p) = p$.*

The periodic orbit contained in $\mathcal{O}(\Gamma, \Phi, \mathcal{A}_1)$ has a period $\leq |\mathcal{O}(\Gamma, \Phi, \mathcal{A}_1)|$. One of our tools uses theorem 16 to provide an algorithm for finding a preimage attack on Φ based on the eventually periodic equivalence classes.

When $q > \kappa$ where $\kappa = |K_j|$, there is an important subtlety to mention. At a first glance, one might expect that the sequence of dynamic keys $K_1, K_2, \ldots$ should always have a period $\leq 2^\kappa$ because the set $\{K_1, K_2, \ldots K_{2^\kappa+1}\}$ must have a collision. This is even further magnified by the birthday paradox that it is likely for the sequence $K_1, K_2, \ldots, K_{2^{\lceil \frac{\kappa}{2} \rceil}}$ to contain two identical dynamic keys. If this dynamic key sequence were produced by a discrete, autonomous dynamical system $f : \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$, then the first collision would determine the periodicity of the key sequence. Instead the orbit $\mathcal{O}(\Gamma,\ \Phi,\ \mathcal{A}_1) \subset \{0,1\}^q$ is used to derive dynamic keys $K_1, K_2, \ldots$. Thus, the dimension of $\mathcal{O}(\Gamma,\ \Phi,\ \mathcal{A}_1)$ can be much greater than 2^κ, particularly when q is substantially greater than κ. Note that in algorithm 3, $\kappa = 128$ and $q = 512$. This subtlety leads us to theorem 16.

Theorem 16 *Suppose $z \in \{0,1\}^q$ has period m with respect to ϕ. Then z has a preimage attack, by computing $m - 1$ iterations of ϕ.*

PROOF. Compute $x = \phi^{m-1}(z)$. Then $\phi(x) = \phi^m(z) = z$. $\square$

ISBN: 978-1-61804-297-2

The following definition helps analyze algorithms 2 and 3.

Definition 17 *A hash function* $\phi : \{0,1\}^{<N} \rightarrow \{0,1\}^q$ *is regular on its subdomain* $\{0,1\}^k$ *with* $k \geq q$ *if for every* $y \in \{0,1\}^q$*, then the intersection of the inverse image* $\phi^{-1}(y)$ *and* $\{0,1\}^k$ *have the same number of points. This means that for every* $y \in \{0,1\}^q$*, then* $|\phi^{-1}(y) \cap \{0,1\}^k| = 2^{k-q}$.

Theorem 18 *Suppose hash function* $\phi : \{0,1\}^{<N} \rightarrow \{0,1\}^q$ *is regular on subdomain* $\{0,1\}^q$*. Then every point in* $\{0,1\}^q$ *is a periodic point and lies in a unique periodic orbit with respect to* ϕ*.*

PROOF. By reductio ad absurdum, suppose $x \in \{0,1\}^q$ is not a periodic point. Let k be the smallest positive natural number such that $y = \phi^k(x)$ is a periodic point. Let m be the period of y. Then $\phi^{-1}(y)$ contains at least two points $\phi^{m-1}(y)$ and $\phi^{k-1}(x)$. These two points contradict the regularity condition of ϕ. The uniqueness of x's periodic orbit immediately follows from the equivalence relation $\underset{\phi}{\sim}$ that ϕ induces on $\{0,1\}^q$. □

When ϕ satisfies the regularity condition on subdomain $\{0,1\}^q$, theorems 16 and 18 are useful because there is no need to search for clever preimage attacks. Instead, the size and number of the periodic orbits of ϕ on $\{0,1\}^q$ can be studied. Corollary 19 states that 2^q equals the sum of the periods of each periodic orbit with respect to ϕ.

Corollary 19 *Let function* $\phi : \{0,1\}^{<N} \rightarrow \{0,1\}^q$ *be regular on subdomain* $\{0,1\}^q$*. Then* $\sum_{[x]} |[x]| = 2^q$ *where the sum ranges over each equivalence class* $[x]$ *induced by* $\underset{\phi}{\sim}$ *and* $|[x]|$ *is the number of points in* $[x]$*. That is,* $|[x]|$ *is the period of* x *with respect to* ϕ*.*

PROOF. $\underset{\phi}{\sim}$ is an equivalence relation on $\{0,1\}^q$. Apply theorem 18. □

Corollary 19 creates a counting tool for finding the probability that a point lies in a periodic orbit with period m. As a simple example, let $\mathcal{S} : \{0,1\}^8 \rightarrow \{0,1\}^8$ denote the substitution box used in AES. Then $\underset{\mathcal{S}}{\sim}$ induces the five equivalence classes $[0], [1], [4], [11], [115]$ on $\{0,1\}^8$. The equivalence class $[0]$ has 59 elements. This implies $\mathcal{S}^{59}(0) = 0$ since S is a bijection. Observe that $[11] = \{43, 241, 161, 50, 35, 38, 247, 104, 69, 110, 159, 219, 185, 86, 177, 200, 232, 155, 20, 250 \; 45, 216, \; 97, 239, 223, 158\}$. Also, $|[1]| = 81$, $|[4]| = 87$, $|[11]| = 27$ and $|[115]| = 2$ and $|[0]| + |[1]| + |[4]| + |[11]| + |[115]| = 2^8$.

During a single execution of algorithm 2, there is a low probability of encrypting two distinct blocks with identical keys. In other words, when $i \neq j$, the event $K_i = K_j$ has a low probability. The following lemma helps sharpen the expression "low probability".

Lemma 20 *Suppose* $\Phi\{0,1\}^{<N} \rightarrow \{0,1\}^q$ *is a* $(N, \sigma, \varrho, r+m+2)$ *one-way preimage function satisfying the regularity condition on subdomain* $\{0,1\}^q$*, where* $r, m \geq 1$*,* $N = n+1$*, and* $\sigma = q$ *and* $\varrho = q^2$*. Suppose machine* $\mathcal{M}$ *computes* Φ *on any input* $x \in \{0,1\}^q$ *in at most* q^m *computational steps. Suppose Alice randomly chooses* $x \in \{0,1\}^q$ *and computes* $\Phi(x) = y$*. Suppose Eve only sees* y*. Set* $\mathcal{S} = \{x \in \{0,1\}^q : |\mathcal{O}(\Gamma, \Phi, \mathcal{A}_1)| < q^r \text{ and } \pi_q \circ \Gamma(0) = x\}$*. Then* $|\mathcal{S}| \leq 2^{-\frac{q}{2}}$.

PROOF Outline. Using machine $\mathcal{M}$, Eve computes the orbit $[y, \Phi(y), \Phi^2(y), \dots]$ with at most q^r iterates. After completing the computation of each iterate $\Phi^k(y)$, Eve searches for a collision in $\{y, \Phi(y), \Phi^2(y), \dots, \Phi^k(y)\}$. If a collision is found, Eve's machine halts. If Eve's machine reaches $\Phi^{q^r}(y)$ and does not find a collision, then Eve's machine halts.

When there is a collision in $\{y, \Phi(y), \dots, \Phi^k(y)\}$, by theorem 18, the regularity condition implies that y lies in this periodic orbit (equivalence class). Let $a = |[y]|$. Then theorem 16 implies $x = \Phi^{a-1}(y)$ is the preimage point sought by Eve. If $|\mathcal{S}| > 2^{-\frac{q}{2}}$, then Eve's machine will find preimage point x in less than $q^{r+m-1} \log q$ computational steps with probability greater than $2^{-\frac{q}{2}}$, contradicting that Φ is a $(N, \sigma, \varrho, r+m+2)$ one-way preimage function. □

Example 21

Consider Φ_{512}, where $q = 512$. Assume $m = 3$ because 512^3 steps is a more conservative upper bound for a TM computing Φ_{512} on $x \in \{0,1\}^{512}$ than 512^2. If Φ_{512} satisfies the regularity condition on subdomain $\{0,1\}^{512}$ and Φ_{512} is a $(2^{128}, q, q^2, 9)$ pre-image hash function, then the probability is $\leq 2^{-256}$ that the key generator in algorithm 3 has an orbit satisfying $|\mathcal{O}(\Gamma, \Phi_{512}, \mathcal{A}_3)| < q^4$; with probability at least $1 - 2^{-256}$, whenever $j \neq k$, then $\Gamma(j) \neq \Gamma(k)$ for an encryption length up to 8.5 billion bytes. Seeing two identical keys that encrypt distinct blocks requires a SHA-512 collision after only 134,217,728 iterations of SHA-512. Although no proof exists of Φ_{512}'s one-wayness, $(2^{128}, q, q^2, 9)$ seems conservative based on

the biclique preimage attack [34] that depends on a reduced 50 rounds instead of 80.

Remark 22

Standard block cipher methods must not reveal the static key to Eve. This is equivalent to not revealing any dynamic key to Eve. To construct future dynamic keys K_k such that $k > j$, Eve must find the preimage point $\Gamma(j)$. In algorithm 3, suppose a processor backdoor leaks four consecutive 128-bit dynamic keys $\beta_{\frac{j}{4}} = K_j\ K_{j+1}\ K_{j+2}\ K_{j+3}$ to Eve. Even after the leak, constructing future keys requires Eve knowing $\Gamma(j)$. For algorithm 3, this involves considerably more computational steps than finding a single, preimage point $x \in \{0,1\}^{1024}$ such that $\Phi_{512}(x) = \beta_{\frac{j}{4}}$. If Φ_{512} is regular on subdomain $\{0,1\}^{1024}$, then $|{\Phi_{512}}^{-1}(\beta_{\frac{j}{4}})| = 2^{512}$. The regularity condition implies Eve must guess $\Gamma(j)$ from 2^{512} possible preimage points. When Eve attempts to find dynamic keys that precede K_j, she has even less information available than when she is attempting to construct future keys. While the last $n-q$ bits of $\Gamma(j)$ are invariant, even if Eve knows $\Gamma(j)$, this doesn't enable her to immediately capture $\Gamma(j-1)$ because $\Phi_{512}(\Gamma_{j-1,0} \ldots \Gamma_{j-1,q-1}) = \Gamma_{j,0} \ldots \Gamma_{j,q-1}$.

Remark 23

A Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ can be expressed as a polynomial $f(x_1, \ldots, x_n) = \sum_{a \in \{0,1\}^n} c_a {x_1}^{a_1} \ldots {x_n}^{a_n}$ over $\mathbb{F}_2[x_1, \ldots, x_n] \ / \ (x_1^2 - x_1, \ldots, x^2 - x_n)$, where $c_a = \sum_{x \leq a} f(x_1, \ldots, x_n)$ and $x \leq a$ iff $x_i \leq a_i$ for each i. The algebraic degree of f is defined as $deg\ f = \max\{wt(a) : a \in \{0,1\}^n, c_a \neq 0\}$, where $wt(a)$ is the Hamming weight of a. Consider functions $f_1, f_2 \ldots f_n : \{0,1\}^n \rightarrow \{0,1\}$ and function $F : \{0,1\}^n \rightarrow \{0,1\}^n$, defined as $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$. The algebraic degree of $F = \max\{deg\ f_1,\ deg\ f_2,\ \ldots,\ deg\ f_n\}$. For a static AES key K, the AES encryption function $\mathcal{E}_K : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ has an algebraic degree ≤ 128 and $\mathcal{E}_K$ is a function of 128 Boolean variables. It is well-known that a Boolean function's resistance to differential cryptanalyis and higher order differentials depends on its algebraic degree and how quickly its degree can be reduced by taking discrete derivatives [35, 36, 37].

Set $M = |\mathcal{O}(\Gamma, \Phi, \mathcal{A}_3)|$. For each dynamic key K_i, let $\mathcal{E}_{K_i} : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$, denote the AES encryption function. During execution of algorithm 3, there are $4M$ distinct functions $\mathcal{E}_{K_0} \ldots \mathcal{E}_{K_{4M-1}}$, where encryption function $\mathcal{E}_{K_0}$ is applied to plaintext block $\mathcal{M}_0$, encryption function $\mathcal{E}_{K_1}$ is applied to block $\mathcal{M}_1$, and so on. This sequence of encryptions induces a function $f_\Gamma : \{0,1\}^{512M} \rightarrow \{0,1\}^{512M}$, where $f_\Gamma = (f_1, f_2, \ldots, f_{512M})$. As discussed in example 21, even for an extremely rare event such as a collision after only 134,217,728 iterations of SHA-512 (if such an orbit exists), the induced f_Γ will be a function of 68,719,476,736 Boolean variables versus 128 Boolean variables for $\mathcal{E}_K$. The cipher block chaining and key generator orbit create a composition of the AES encryption functions $\mathcal{E}_{K_0}$, $\mathcal{E}_{K_1}$, ...; for example, $\mathcal{C}_2 = \mathcal{E}_{K_2}(\mathcal{M}_2 \oplus \mathcal{E}_{K_1}(\mathcal{M}_1 \oplus \mathcal{E}_{K_0}(\mathcal{M}_0 \oplus \mathcal{C}_{-1})))$. Thus, $f_{1+128k}, \ldots f_{128(k+1)}$ are a function of the $128(k+1)$ variables $x_1, \ldots, x_{128(k+1)}$ for $0 \leq k < 4M$. Based on the work of Boura, Canteaut [38] and Biss [39], we conjecture that for most key generator orbits the degree of f_Γ is at least M.

6 Algorithms 2 and 3 Stop a Generic Block Cipher Attack

The dynamic keys, derived in algorithms 2 and 3, help stop Huang and Lai's generic block cipher attack [21], which is described below and shown in algorithm 24. The following list describes the symbols, used in their attack algorithm 24.

P `plaintext`
C `ciphertext`
n `block size`
K `master key`
k `master key size`
R `number of rounds`
S `non-linear layer`
L `linear layer`
K^r `subkey used in round` r
X^r `input block to round` r `where` $X^0 = P$
Y^r `output block of the key mixing in round` r
Z^r `output block of the nonlinear layer in round` r
${Z_i}^r$ i`th subblock in` Z^r

S_1 is the internal state that can be calculated from P only with k_1 bits of subkeys, where k_1 is the maximum smaller than k that can be obtained. Similarly, S_2 is the internal state that can be derived from C only with (other) k_1 bits of subkeys. For any block cipher, the states of S_1 and S_2 can be found. The attack algorithm has two stages:

1. A meet-in-the-middle stage generates the candidate list containing 2^{k-M} keys, where M is the met intermediate size.

ISBN: 978-1-61804-297-2

2. A check stage that examines the keys in the candidate list.

Line numbers have been added to the attack algorithm in [21] to help explain how algorithms 2 and 3 hinder this attack.

Algorithm 24 *Generic Block Cipher Attack*

Data: $\lceil \frac{k}{n} \rceil$+1 (plaintext, ciphertext) pairs

Result: the output key K

1 for each value in the 1st k_1 key bits {

2 compute S_1 from P with these k_1 bits

3 for each value in the remaining $k - k_1$ key bits {

4 compute $Z_0^{\lfloor \frac{R}{2} \rfloor}$ from S_1

5 store $Z_0^{\lfloor \frac{R}{2} \rfloor}$ in a table corresponding to the guessed key

6 }

7 }

8 for each value in the last k_1 key bits {

9 compute S_2 from C with these k_1 bits

10 for each value in the remaining $k - k_1$ key bits {

11 compute $Z_0^{\lfloor \frac{R}{2} \rfloor}$ from S_2

12 if $Z_0^{\lfloor \frac{R}{2} \rfloor}$ corresponding to the guessed key is in the table {

13 add guessed key to candidate list

14 move onto the next guess

15 }

16 else move onto the next guess

17 }

18 }

19 Check keys in candidate list with other $\lceil \frac{k}{n} \rceil$ (plaintext, ciphertext) pairs

Algorithm 24's method of using a candidate key list to find the static key of the block cipher is not effective against cryptographic algorithms 2 and 3. To illustrate this, in algorithm 3, after each 64 byte block is encrypted, the candidate list of keys changes because the next four 128-bit keys $K_j, K_{j+1}, K_{j+2}, K_{j+3}$ are derived from an updated key generator $\Gamma_{j,0} \ldots \Gamma_{j,1023}$ where the average Hamming distance between $\Gamma_{j,0} \ldots \Gamma_{j,511}$ and $\Gamma_{j-1,0} \ldots \Gamma_{j-1,511}$ is 256.

Consider algorithm 3, encrypting 25,600 bytes of voice data per second. At this rate, a one hour phone conversation requires a key generator orbit $(\Gamma_{0,0} \ldots \Gamma_{0,511})$, $\Phi_{512}(\Gamma_{0,0} \ldots \Gamma_{0,511})$, $\ldots$ ${\Phi_{512}}^{1440000}(\Gamma_{0,0} \ldots \Gamma_{0,511})$ with size 1,440,001. If a collision occurred in this orbit during a one hour phone call, then theorem 16 provides a devastating, preimage attack on SHA-512 with at most 1,440,000 iterations of SHA-512. Based on an extremely low probability of this rare event (such orbits may not even exist), a collision would also imply that SHA-512 does not satisfy any reasonable values of $(2^{128}, \sigma, \varrho, r)$ preimage complexity. "Reasonable" means not constraining Eve's machine $\mathcal{P}$ so much that she cannot compute, for example, SHA-512. Consider $\varrho = 1$, so machine $\mathcal{P}$ can have only one state.

Recall that the biclique preimage attack [34] – on a reduced 50 rounds of SHA-512 instead of the complete 80 – has an estimated preimage complexity of $2^{511.5}$. From this work, it is considerably more likely that an orbit $\mathcal{O}(\Gamma, \Phi_{512}, \mathcal{A}_3)$ has a size far greater than the number of SHA-512 iterations needed to provide a complete encryption for any foreseeable application. In this case, the assumption that there are $\lceil \frac{k}{n} \rceil$ (plaintext, ciphertext) pairs does not hold for algorithm 3. Furthermore, the lack of $\lceil \frac{k}{n} \rceil$ (plaintext, ciphertext) pairs invalidates the effectiveness of the loop composed of lines 1 through 7 and the loop composed of lines 8 through 18.

7 Speed Testing of Algorithm 3

Algorithm 3's execution speed is compared to standard AES-128. Figure 1 shows 10,000 speed tests, measured in microseconds, where AES-128 uses a static key to encrypt 64 bytes of random plaintext.

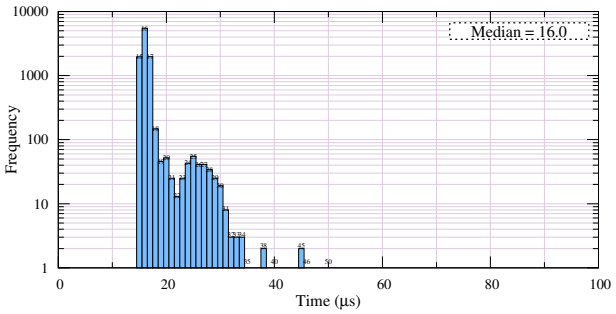

Figure 1: Static key AES-128 encrypts 64 bytes.

All speed tests were performed on an Apple Mac mini, running OSX 10.9.2 with a 2.5 Ghz Intel Core. The median was selected over the sample mean [40] in order to filter out the effects of OSX interrupts. All random plaintext, keys and key generators were created from `device/urandom`.

Figure 2 shows 10,000 speed tests, measured in microseconds. In each of these tests, algorithm 3 encrypts 64 bytes of random plaintext.

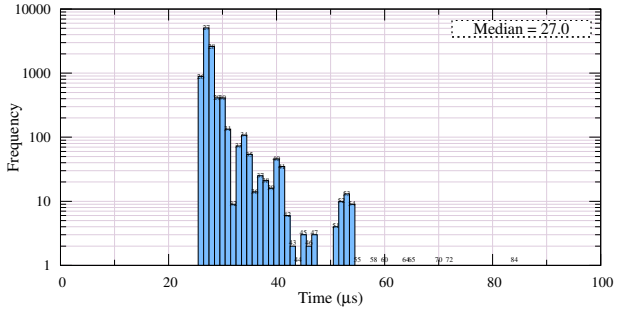

Figure 2: Algorithm 3 encrypts 64 bytes.

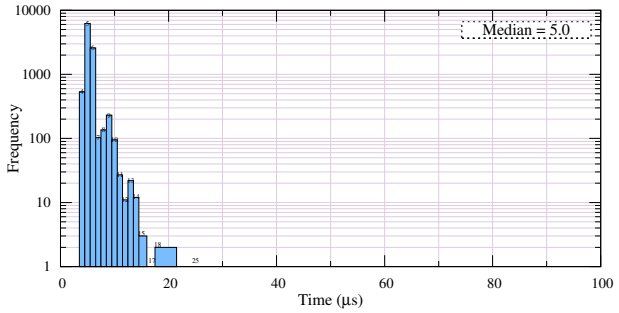

Figure 3: Four AES 128-bit key expansions.

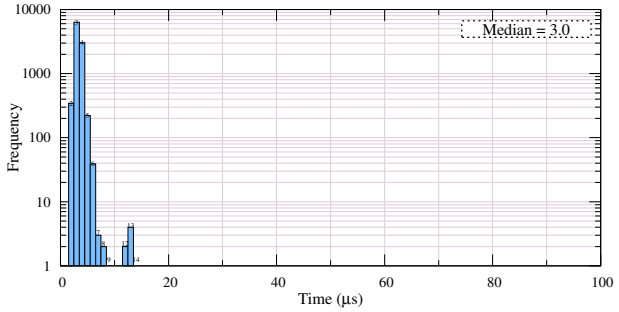

Figure 4: One key generator update with SHA-512.

When algorithm 3 is executed, the key expansion is executed for each 16 byte block because AES encrypts each block with a different 128-bit key. This motivates testing how much time is due to expansion of the 128-bit key versus how much time is due to key generator updating and dynamic key derivation from the 1024-bit key generator. Figures 3, 4 and 5 show these tests.

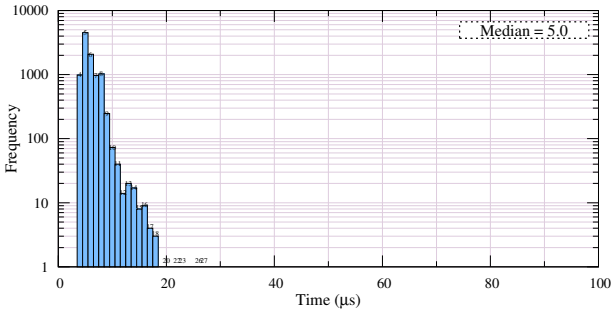

Figure 5: SHA-512 hash of a 1024-bit key generator.

Overall, algorithm 3 increases the execution time by almost 70 percent over standard AES-128 for these 64 byte tests, as indicated in figures 3, 4 and 5. The 128-bit key expansion uses almost 39 percent of this increase in execution time.

8 Summary and Future Research

In algorithms 2 and 3, a successful attack that obtains a sequence element $\Gamma(j)$ of the key generator requires at least a preimage attack on a one-way preimage hash function, where no direct information about the one-way preimage digest is revealed to Eve. When the one-way preimage hash function satisfies our regularity condition, obtaining $\Gamma(j)$ requires Eve guessing the correct preimage point from all possible preimage points; when the key generator element $\Gamma(j)$ has length n bits, and the digest size is q, there are 2^{n-q} possible preimage points. Furthermore, if Eve successfully captures $\Gamma(j)$, she still must find additional preimage attacks to obtain preceding dynamic keys. The complexity is lower for a standard block cipher because Eve is searching for a static key used directly by the key scheduling.

Future research will focus on the theoretical security of computable key generators, which depends on the existence of one-way hash functions and a better understanding of their dynamical behavior. In this regard, a number theoretic method has been designed that satisfies our regularity condition and the propagation criteria [41].

9 Acknowledgements

I would like to thank Babil Golam Sarwar for his helpful advice and comments.

References:

[1] Claude Shannon. Communication Theory of Secrecy Systems. 1949.

[2] Horst Feistel. Cryptography and Computer Privacy. Scientific American. **228**, No. 5, 15–23, 1973.

[3] Oded Goldreich. Foundations of Cryptography. I Basic Tools. Cambridge University Press. 2001.

[4] Mihir Bellare and Phillip Rogaway. Introduction to Modern Cryptography. 2005.

[5] T.W. Cusick and P. Stanica. Cryptographic Boolean Functions and Applications. Academic Press, 2009.

[6] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. Series 2 **42** (Parts 3 and 4), 230–265, 1936.

[7] Joan Daemen and Vincent Rijmen. The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer, 2002.

[8] NIST. Advanced Encryption Standard (AES), FIPS 197. Nov. 2001.

[9] Juliano Rizzo and Thai Duong. Practical Padding Oracle Attacks. Black Hat Conference. 2010.

[10] Alex Biryukov and D. Khovratovich: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: Asiacrypt. LNCS **5912**, Springer, 1–18, 2009.

[11] Alex Biryukov, Khovratovich, D., Nikolic, I. Distinguisher and Related-Key Attack on the Full AES-256. Advances in Cryptology - Crypto 2009. LNCS **5677**. Springer, 231–249, 2009.

[12] P. Derbez, Pierre-Alain Fouque and J. Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. Advances in Cryptology - Eurocrypt 2011. LNCS **7881**. Springer, 371–387, 2011.

[13] Alex Biryukov and Dmitry Khovratovich. Feasible Attack on the 13-round AES-256. 2010.

[14] Daniel Bernstein. Cache-timing attack on AES. 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[15] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. Advances in Cryptology - Asiacrypt 2011. LNCS **7073**, Springer, 344–371, 2011.

[16] Orr Dunkelman, Nathan Keller, Adi Shamir. Improved Single-Key Attacks on 8-round AES. Cryptology ePrint Archive, Report 2010:322, 2010.

[17] Jon Passki and Tom Ritter. An Adaptive-Ciphertext Attack against I $\oplus$ C Block Cipher Modes with Oracle. IACR Cryptology ePrint Archive 2012:292, 2012.

[18] Daniel Bernstein and Tanja Lange. Non-uniform cracks in the concrete: the power of free precomputation. Advances in Cryptology - Asiacrypt. LNCS **8270**. Springer, 321–340, 2013.

[19] Carlos Cid, Sean Murphy and Matthew Robshaw. Algebraic Aspects of AES. Berlin: Springer, 2006.

[20] Sean Murphy and Matthew Robshaw. Essential algebraic structures within the AES. Advances in Cryptography. Crypto 2002. LNCS, **2442**, Berlin: Springer, 1–16, 2002.

[21] Jialin Huang and Xuejia Lai. What is the Effective Key Length for a Block Cipher: an Attack on Every Block Cipher. Science China Information Sciences. **57**, Issue 7, Springer, 1–11, 2014.

[22] Ralph C. Merkle. Secure Communications over Insecure Channels. Communications of the ACM. **21** (4), 294299, April 1978. Rejected 1975 submission: http://merkle.com/1974/Puzzles1975.12.07.pdf

[23] Whitfield Diffie and Martin Hellman. New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654, 1976.

[24] Daniel Bernstein. Curve25519: new Diffie−Hellman speed records. Public Key Cryptography. LNCS **3958**. New York, Springer. 207–228, 2006.

[25] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, Bo-Yin Yang. High-speed high-security signatures. Journal of Cryptographic Engineering. **2**, 77-89, 2012.

[26] A.F. Webster and S.E. Tavares. On the Design of S-Boxes. Advances in Cryptology. CRYPTO 85 Proceedings. LNCS **218**. Springer, 523–534, 1986.

ISBN: 978-1-61804-297-2

[27] NIST. FIPS-180-2: Secure Hash Standard, August 2002.

[28] Claude Shannon. A universal Turing machine with two internal states. Automata Studies, C.E. Shannon and J. McCarthy (eds.). Princeton University Press, 129–153, 1956.

[29] Yiannis N. Moschovakis. What is an algorithm? In Mathematics Unlimited 2001 and beyond (eds. B. Engquist and W. Schmid), Springer. 919–936, 2001.

[30] Yiannis N. Moschovakis. Algorithms and Implementations. Tarski Lecture 1, March 3, 2008. http://www.math.ucla.edu/~ynm/lectures/tlect1.pdf

[31] Yuri Gurevich. What is an algorithm? In SOFSEM: Theory and Practice of Computer Science (eds. M. Bielikova et al.), LNCS **7147**. Springer. 31-42, 2012. http://research.microsoft.com/pubs/155608/209-3.pdf

http://link.springer.com/chapter/10.1007%2F978-3-642-27660-6_3#page-1

[32] Michael S. Fiske. Turing Incomputable Computation. Turing-100 Proceedings. Alan Turing Centenary. EasyChair 10, 69–91, 2012.

http://www.aemea.org/Turing100.

[33] Michael S. Fiske. Quantum Random Active Element Machine. UCNC 2013 Proceedings. LNCS **7956**, 252–254, Springer, 2013.

http://www.aemea.org/UCNC2013.

[34] Dmitry Khovratovich, Christian Rechberger and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family. FSE, 244–263, 2012.

[35] E. Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. Advances in Cryptology. CRYPTO 90 Proceedings. LNCS **537**. Springer, 2–21, 1990.

[36] Xuejia Lai. Higher Order Derivatives and Differential Cryptanalysis. In Communications and Cryptography: Two Sides of One Tapestry, R.E. Blahut et al., eds., Kluwer Adademic Publishers, 227–233, 1994.

[37] L. Knudsen. Truncated and higher order differentials. In Fast Software Encryption. Springer, 196–211, 1995.

[38] Christina Boura and Anne Canteaut. On the Influence of the Algebraic Degree of F^{-1} on the Algebraic Degree of $G \circ F$. IEEE Transactions on Information Theory. **59**, No. 1, 691–702, Jan. 2013.

[39] Daniel Biss. A lower bound on the number of functions satisfying the strict avalanche criterion. Discrete Math. **185**, 29–39, 1998.

[40] Bart Kosko. Noise: THE SAMPLE MEAN. What have you changed your mind about? The Edge, 2008.

[41] B. Preneel, W.V. Leekwijck, L.V. Linden, R. Govaerts, J. Vandewalle. Propagation characteristics of Boolean functions. Advances in Cryptology. EUROCRYPT 90 Proceedings. LNCS **473**. Springer, 161–173, 1991.