IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 3, pp. 362–366. DOI:10.13154/tosc.v2020.i3.362-366

Errata to Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers

Aldo Gunsing, Joan Daemen and Bart Mennink

Digital Security Group, Radboud University, Nijmegen, The Netherlands
aldo.gunsing@ru.nl, joan@cs.ru.nl, b.mennink@cs.ru.nl

Abstract. In ToSC 2018(4), Daemen et al. performed an in-depth investigation of sound hashing modes based on arbitrary functions, permutations, or block ciphers. However, for the case of invertible primitives, there is a glitch. In this errata, we formally fix this glitch by adding an extra term to the security bound, $q/2^{b-n}$, where q is query complexity, b the width of the permutation or the block size of the block cipher, and n the size of the hash digest. For permutations that are wider than two times the chaining value this term is negligible. For block cipher based hashing modes where the block size is close to the digest size, the term degrades the security significantly.

Keywords: hash functions · tree hashing · sufficient conditions · indifferentiability · errata

1 Introduction

In [DMA18], Daemen, Mennink, and Van Assche performed a thorough investigation of cryptographic hashing modes. They considered a very large class of hashing modes built on top of arbitrary functions, permutations, or block ciphers, and derived sufficient conditions for these modes to be hard to differentiate from a random oracle. Their analysis generalized earlier attempts of Dodis et al. [DRRS09] and Bertoni et al. [BDPV14]. Most importantly, the contribution of Daemen et al. consisted of cleaner sufficiency conditions and analyses for permutation based cryptographic hashing modes. While the conceptually cleaner sufficiency conditions simplified the security analyses, a level of complication was introduced by the fact that more general modes than in [DRRS09, BDPV14] were taken into consideration.

After publication of the original article, it turned out that there was an error in the proof of the mode for a truncated permutation or a block cipher [Nev19]. At a high level, the attack consisted of (i) querying the construction oracle for an arbitrary message $M\|m$ to get a hash digest h, (ii) querying the inverse primitive on input of the hash outcome h (possibly appended, as truncation is involved), and (iii) using the previous result to compute the hash of $M\|m'$, without having to know M but only h, m and m'. Intuitively, the attack would succeed after 2^{b-n} attempts, where b is the width of the permutation or block length of the block cipher, and n the hash digest size. The problem is discussed at a higher level of technicality in Section 2.

Fortunately, the glitch is quite simple to fix. In this errata to the original article of Daemen et al. [DMA18], we correct the analysis for the case of modes based on a permutation or block cipher. The original analysis carries over with an additional term $q/2^{b-n}$, where b is the width of the primitive or block size of the block cipher, and n the hash digest size. The updated indifferentiability bounds for the relevant hashing modes are given in Table 1. The updated analysis is described in Section 3. For truncated permutations,

Received: 2020-06-01 Accepted: 2020-08-01 Published: 2020-09-28

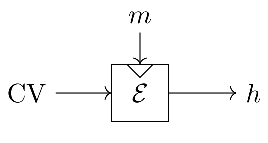
Table 1: Updated indifferentiability bounds for hashing modes of an arbitrary function, a truncated permutation or a (truncated) block cipher. The rectangles denote the additional terms. The conditions SF, RD, MD, and LA stand for subtree-freeness, radical-decodability, message-decodability, and leaf-anchoring, respectively (see the original article [DMA18] for their definitions). q is the adversarial complexity expressed as the number of primitive queries either direct or indirect, n the CV length, and $b \geq n$ the width of the permutation resp. the block length of the block cipher.

compression function type	SF+RD+MD	LA	bound
arbitrary function	✓	—	$\frac{\binom{q}{2}}{2^n}$
truncated permutation	✓	✓	$\frac{\binom{q}{2}+1}{2^n} + \frac{\binom{q}{2}}{2^b} + \boxed{\frac{q}{2^{b-n}}}$
truncated block cipher	✓	✓	$\frac{\binom{q}{2}+1}{2^n} + \frac{\binom{q}{2}}{2^b} + \boxed{\frac{q}{2^{b-n}}}$

the impact of the additional term is negligible if $b - n \geqslant n$, i.e., if $b \geqslant 2n$. This is the case for wide permutations such as Keccak-p[1600] or Keccak-p[800], or the permutation used in MD6. For lightweight permutations, the presence of the term becomes problematic. However, these do not lend themselves easily to the tree hashing modes considered in this work in the first place, as at least two CVs should fit in a single permutation input, hence $b \geqslant 2n$. For the modes based on block ciphers b is the block length. In most cases there is no truncation, i.e., implying $b = n$ and all security evaporates. However, if there is truncation and $n \leqslant b/2$, the additional term is negligible. For example, using the block cipher underlying SHA-512/256 [SHA15] we get $b = 512$ and $n = 256$, which gives 128 bits of security both without and with the extra term. Concluding, the extra term leads to an extra requirement for modes to be secure, in that sufficient truncation has to be done.

2 Problem in Original Analysis

For simplicity, consider a block cipher based tree hashing mode without truncation, i.e., with $b = n$, for the processing of a message $M\|m$, where M is of arbitrary length and m of fixed length. The computation of the final node when computing $\mathcal{T}(M\|m) = h$ would typically be of the following form:



Here, CV is the intermediate result when compressing M. Because the block cipher is invertible, an attacker can compute $\mathrm{CV} = \mathcal{E}_m^{-1}(h)$. It can use this information to compute $\mathcal{T}(M\|m') = h' = \mathcal{E}_{m'}(\mathrm{CV})$ based on just h, m and m', thus without any knowledge of M. It can use this trick to differentiate the hashing mode from a random oracle.

In short, an attacker can do the following:

1. query $\mathcal{T}(M\|m) = h$,
2. query $\mathcal{E}_m^{-1}(h) = \mathrm{CV}$,
3. query $\mathcal{E}_{m'}(\mathrm{CV}) = h'$,

4. verify $h' = \mathcal{T}(M \| m')$.

The problem for the simulator is that it does not remain consistent. We assume that at the end of the interaction, the distinguisher will always verify its queries to the random oracle. In above case, it will query the simulator for the compression of M. The simulator will return a random value CV', which is unlikely to be equal to CV. However, to be consistent with the random oracle, the simulator has to return h for $\mathcal{E}_m(\mathrm{CV}')$ as well. This means that it is no longer consistent as a block cipher, as both CV and CV' are mapped to h under $\mathcal{E}_m$. Note that the simulator is already inconsistent without querying $\mathcal{E}_{m'}(\mathrm{CV})$, but we do need that additional query to exploit the weakness in a real hashing mode.

The same problem holds when a truncated permutation is used and $b - n$ is small. In general, the attacker gets an advantage of $q/2^{b-n}$ by guessing the truncated bits.

In general, the problem is caused by the fact that the final primitive call in the hashing mode is invertible and that the attacker succeeds in inverting it in 2^{b-n} attempts, as it must correctly guess the $b-n$-bit truncated part. The attack therefore does not affect the analysis of [DMA18] for arbitrary functions, but only for permutations and block ciphers.

3 Updated Analysis

As a reference, we first restate the simulator and bad views of [DMA18, Theorem 2] for the case of a permutation. The simulator is given in Algorithm 3. For the bad views, we denote by $\mathcal{M} = \{(M_1, Z_1, h_1), \ldots, (M_r, Z_r, h_r)\}$ the view seen by distinguisher $\mathcal{D}'$ on interaction with the construction oracle, and by $\text{Ł} = \{(x_1, y_1), \ldots, (x_q, y_q)\}$ the view seen by $\mathcal{D}'$ on interaction with the primitive oracle. The set Ł is split into forward queries $\mathcal{L}^{\mathrm{fwd}}$ and inverse queries $\mathcal{L}^{\mathrm{inv}}$. We further split $\mathcal{L}^{\mathrm{fwd}}$ into

$$\begin{aligned} \mathcal{L}^{\mathrm{rad}} &= \{(x_i, y_i) \in \mathcal{L}^{\mathrm{fwd}} \mid S = \mathrm{radicalExtend}[\mathcal{L}^{\mathrm{fwd}}_{i-1}](x_i) \in \mathcal{S}^{\mathrm{rad}}_{\mathcal{T}}\}, \\ \mathcal{L}^{\mathrm{other}} &= \mathcal{L}^{\mathrm{fwd}} \setminus \mathcal{L}^{\mathrm{rad}}. \end{aligned}$$

Denote $\nu = (\mathcal{M}, \mathcal{L}^{\mathrm{rad}}, \mathcal{L}^{\mathrm{other}}, \mathcal{L}^{\mathrm{inv}})$. The set $\mathcal{V}$ denotes any attainable view that can be observed by $\mathcal{D}'$.

An attainable view ν is called *bad* if:

(i) There exist distinct $(x_i, y_i), (x_j, y_j) \in \mathcal{L}^{\mathrm{rad}}$ with $\lfloor y_i \rfloor_n = \lfloor y_j \rfloor_n$;

(ii) There exist distinct $(x_i, y_i), (x_j, y_j) \in \mathcal{L}^{\mathrm{other}}$ with $\lfloor y_i \rfloor_n = \lfloor y_j \rfloor_n$;

(iii) There exist $(x_i, y_i) \in \mathcal{L}^{\mathrm{rad}}$ and $(x_j, y_j) \in \mathcal{L}^{\mathrm{other}}$ with $i < j$ such that $\lfloor y_j \rfloor_n = \mathrm{radicalValue}[\text{Ł}_{i-1}](x_i)$;

(iv) There exist $(x_i, y_i) \in \mathcal{L}^{\mathrm{inv}}$ such that $\lfloor x_i \rfloor_n = \mathrm{IV}$;

(v) There exist $(x_i, y_i) \in \mathcal{L}^{\mathrm{inv}}$ and $(x_j, y_j) \in \mathcal{L}^{\mathrm{fwd}}$ such that $\lfloor x_i \rfloor_n = \lfloor y_j \rfloor_n$;

(vi) There are distinct $(x_i, y_i), (x_j, y_j) \in (\mathcal{L}^{\mathrm{fwd}} \cup \mathcal{L}^{\mathrm{inv}})$ with $x_i = x_j$ or $y_i = y_j$.

The error of [DMA18] happens in the analysis of bad event (vi). The original paper assumes that the simulator always returns random values from $\mathbf{Z}_2^b$ for new inputs, which leads to the term $\binom{|\text{Ł}|}{2}/2^b = \binom{q}{2}/2^b$. However, when a query completes a tree, the first n bits of its result are taken from the random oracle, whose bits are not random when the distinguisher has queried it earlier. Below a corrected analysis of bad event (vi) is given.

Let $(x_i, y_i), (x_j, y_j) \in (\mathcal{L}^{\mathrm{fwd}} \cup \mathcal{L}^{\mathrm{inv}})$ be arbitrary distinct queries with $i < j$. We do some case separation based on what kind of queries i and j are.

Algorithm 3

```
Interface: $\mathcal{S} : \mathbf{Z}_2^b \to \mathbf{Z}_2^b \ , \ \ x \mapsto y$
  if $x \notin \mathrm{dom}Ł$ then
    $S \leftarrow \mathrm{radicalExtend}[\mathcal{L}^{\mathrm{fwd}}](x)$        ▷ radical-extend tree from single node
    if $S \in \mathcal{S}_{\mathcal{T}}$ then                                   ▷ query completes tree
      $(M, Z) \leftarrow \mathrm{extract}(S)$                                   ▷ extract message and tree template
      $z \xleftarrow{\$} \mathbf{Z}_2^{b-n}$
      $y \leftarrow \mathcal{RO}(M, Z) \| z$
    else                                                                        ▷ query does not complete tree
      $y \xleftarrow{\$} \mathbf{Z}_2^b$
    end if
    $Ł(x) \leftarrow y$
  end if
  return $Ł(x)$

Interface: $\mathcal{S}^{-1} : \mathbf{Z}_2^b \to \mathbf{Z}_2^b \ , \ \ y \mapsto x$
  if $y \notin \mathrm{rng}Ł$ then
    $x \xleftarrow{\$} \mathbf{Z}_2^b$
    $Ł^{-1}(y) \leftarrow x$
  end if
  return $Ł^{-1}(y)$
```

1. Suppose $(x_j, y_j) \notin \mathcal{L}^{\mathrm{other}}$. Then y_j (for a forwards query) or x_j (for a backwards query) is chosen randomly from $\mathbf{Z}_2^b$, hence the probability of a collision is indeed $1/2^b$.

2. Suppose $(x_j, y_j) \in \mathcal{L}^{\mathrm{other}}$. Now we separate based on query i.

   (a) Suppose $(x_i, y_i) \in \mathcal{L}^{\mathrm{other}}$. By the negation of bad event (ii) we know that $\lfloor y_i \rfloor_n \neq \lfloor y_j \rfloor_n$, hence we cannot have a collision.

   (b) Suppose $(x_i, y_i) \notin \mathcal{L}^{\mathrm{other}}$. This is the problematic case. We merely know that the final $b - n$ bits of y_j are chosen randomly from $\mathbf{Z}_2^{b-n}$. If query j does not complete a tree, the other bits are also chosen randomly, but we can only guarantee the final $b - n$ bits for both cases combined. This means that the probability of a collision is bounded by $1/2^{b-n}$. Note that for a collision to happen we have the additional requirement that $\lfloor y_i \rfloor_n = \lfloor y_j \rfloor_n$.

As we only get a probability of $1/2^{b-n}$ when $(x_i, y_i) \notin \mathcal{L}^{\mathrm{other}}$, $(x_j, y_j) \in \mathcal{L}^{\mathrm{other}}$ and $\lfloor y_i \rfloor_n = \lfloor y_j \rfloor_n$, we can bound the number of problematic cases. For every query $(x_i, y_i) \notin \mathcal{L}^{\mathrm{other}}$ there is at most one query $(x_j, y_j) \in \mathcal{L}^{\mathrm{other}}$ with $\lfloor y_i \rfloor_n = \lfloor y_j \rfloor_n$ by bad event (ii). Hence the number of suitable pairs is at most $|Ł \setminus \mathcal{L}^{\mathrm{other}}| \leqslant q$. This means that we get the original term $\binom{|Ł|}{2}/2^b = \binom{q}{2}/2^b$ plus an additional one of $q/2^{b-n}$.

The fix to [DMA18, Theorem 3], for the block cipher based modes, is identical.

Acknowledgments

The authors would like to thank Samuel Neves, Gilles Van Assche, and the anonymous reviewers of ToSC for their valuable feedback. Aldo Gunsing is supported by the Netherlands Organisation for Scientific Research (NWO) under TOP grant TOP1.18.002 SCALAR. Joan Daemen is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA.

References

[BDPV14] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sufficient conditions for sound tree and sequential hashing modes. *Int. J. Inf. Sec.*, 13(4):335–353, 2014.

[DMA18] Joan Daemen, Bart Mennink, and Gilles Van Assche. Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2018(4):197–228, 2018.

[DRRS09] Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6. In Orr Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*, pages 104–121. Springer, 2009.

[Nev19] Samuel Neves. Personal communication, 2019.

[SHA15] National Institute of Standards and Technology. FIPS 180-4: Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-4, August 2015.