On *LANE* Hashing Modes of Operation

Elena Andreeva

SCD-COSIC, Dept. of Electrical Engineering, Katholieke Universiteit Leuven

1 Basic Definitions

Let $H : \{0,1\}^* \to \{0,1\}^n$ be an infinite-domain keyless hash function and $\mathrm{F} : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a finite-domain compression function. For practical reasons we set a bound λ on the domain size of H where $\lambda < 2^{64}$. Possible additional hash function input parameters are a salt value S and a bit counter value C of lengths s and c, respectively. Then $\mathrm{F} : \{0,1\}^b \times \{0,1\}^n \times \{0,1\}^c \times \{0,1\}^s \to \{0,1\}^n$.

KEYS VERSUS SALTS. These play very different roles both in formal security definitions and in practice. The key K is publicly known and fixed parameter and formally selects a single function from a hash function family, while the salt S adds randomness per message. Our hash function proposal does not make use of fixed keys but offers an alternative randomized mode of operation that allows the use of salt values. We motivate this choice with the fact that known fixed keys in the hash function setting do not serve as a tool for randomization and once they are fixed they become known constants for the hash function algorithm. On the other hand, salts provide the necessary randomization tool per message at the expense of additional transmission and storage overhead in certain applications.

SECURITY NOTIONS. For a hash function H and a security property X, we measure the adversarial advantage in breaking X by $\mathbf{Adv}_H^X$, where $X \in \mathit{Properties}$. Some of the properties X are parametrized by the message length $\lambda \in \mathbb{N}$, $X[\lambda]$, for reasons having to do with uniform sampling from the message space. The set of $\mathit{Properties} = \{\mathrm{Coll}, s\mathrm{Coll}, sw\mathrm{Coll}, \mathrm{Sec}, s\mathrm{Sec}, \mathrm{eSec}, \mathrm{Pre}, s\mathrm{Pre}^1, s\mathrm{Pre}^2\}$ includes the properties of collision security, salted collision security, salted weak collision security, second preimage security, salted second preimage security, everywhere second preimage security, preimage security and salted preimage security variants one and two, respectively. The adversarial advantage for each of these security notions is measured as follows:

1. **Collision Security (**Coll**)**

$$\mathbf{Adv}_H^{\mathrm{Coll}}(A) = \Pr\left[(M, M') \xleftarrow{\$} \mathsf{A} \ : \ \begin{array}{c} M \neq M' \text{ and} \\ H(M) = H(M') \end{array} \right]$$

2. **Salted Collision Security (**sColl**)**

$$\mathbf{Adv}_H^{s\mathrm{Coll}}(A) = \Pr\left[(M, S, M', S') \xleftarrow{\$} \mathsf{A} \ : \ \begin{array}{c} M \neq M' \text{ and} \\ H(M, S) = H(M', S') \end{array} \right]$$

   Intuition: The adversary has a full control over the salt values S and S' where either $S = S'$ or $S \neq S'$.
   Remark: Notice that colliding tuples (M, S) and (M', S') with $M = M'$ and $S \neq S'$ have no impact in any practical scenario. An adversary that outputs a collision for identical messages $M = M'$ and distinct salts $S \neq S'$ does not violate the Coll security of any practical hash function application. Therefore, we impose the practically grounded and mild condition that a collision is only valid when the colliding M and M' are distinct (independently of the salt values).
3. **Salted Weak Collision Security (**$sw\mathrm{Coll}[\lambda]$**)** This notion coincides with the enhanced target collision security of Halevi and Krawczyk [6].

$$\mathbf{Adv}_H^{sw\mathrm{Coll}[\lambda]}(A) = \Pr\left[\begin{array}{c} M \xleftarrow{\$} \{0,1\}^\lambda \\ (S, M', S') \xleftarrow{\$} \mathsf{A}(M) \end{array} \ : \ \begin{array}{c} M \neq M' \text{ and} \\ H(M, S) = H(M', S') \end{array} \right]$$

Intuition: An adversary is given a target message M and has to output (S, M', S') where either $S = S'$ or $S \neq S'$. In practical scenarios this means that an adversary is able to influence (modify) the challenge salt S value according to its choice prior to the target message hashing. Notice that the computational complexity of a swColl attack against an ideal hash function H in this case is of order $2^{n/2}$.
(Note also that this notion is not equivalent to sColl. There exists a trivial counterexample function H for which it is hard to find collisions in the swColl sense but easy to find collisions in the sColl sense, e.g. $H(M, S) = 0^n$ when $M = 1^b$ and $H(M, S) = G(M, S)$ otherwise, where G is swColl secure compression function and $|M| = b$ with $b \geq n$).

4. **Second Preimage Security (**$\mathrm{Sec}[\lambda]$**)**

$$\mathbf{Adv}_H^{\mathrm{Sec}[\lambda]}(\mathsf{A}) = \Pr\left[\begin{array}{c} M \xleftarrow{\$} \{0,1\}^\lambda \\ M' \xleftarrow{\$} \mathsf{A}(M) \end{array} : \begin{array}{c} M \neq M' \text{ and} \\ H(M) = H(M') \end{array}\right]$$

5. **Salted Second Preimage Security (**$s\mathrm{Sec}[\lambda, s]$**)**

$$\mathbf{Adv}_H^{s\mathrm{Sec}[\lambda,s]}(A) = \Pr\left[\begin{array}{c} S \xleftarrow{\$} \{0,1\}^s; M \xleftarrow{\$} \{0,1\}^\lambda \\ (M', S') \xleftarrow{\$} \mathsf{A}(M, S) \end{array} : \begin{array}{c} M \neq M' \text{ and} \\ H(M, S) = H(M', S') \end{array}\right]$$

Intuition: An adversary is given a target tuple (M, S) and has to produce a colliding tuple (M', S') where either $S = S'$ or $S \neq S'$. The sSec notion depicts a real-world scenario in which the adversary does (can) not influence the challenge salt value S.

6. **Everywhere Second Preimage Security (**$\mathrm{eSec}[\lambda, s]$**)**

$$\mathbf{Adv}_H^{\mathrm{eSec}[\lambda,s]}(A) = \Pr\left[\begin{array}{c} M \leftarrow A\,;\ S \xleftarrow{\$} \{0,1\}^s; \\ (M', S') \xleftarrow{\$} \mathsf{A}(S, St) \end{array} : \begin{array}{c} M \neq M' \text{ and} \\ H(M, S) = H(M', S') \end{array}\right]$$

This notion is formally introduced in [10]. A here is a stateful adversary with a state St.

7. **Preimage Security (**$\mathrm{Pre}[\lambda]$**)**

$$\mathbf{Adv}_H^{\mathrm{Pre}[\lambda]}(\mathsf{A}) = \Pr\left[\begin{array}{c} M \xleftarrow{\$} \{0,1\}^\lambda\,;\ Y \leftarrow H(M) \\ M' \xleftarrow{\$} \mathsf{A}(Y) \end{array} : H(M') = Y\right]$$

8. **Variant One for Salted Preimage Security (**$s\mathrm{Pre}^1[\lambda, s]$**)**

$$\mathbf{Adv}_H^{s\mathrm{Pre}^1[\lambda,s]}(\mathsf{A}) = \Pr\left[\begin{array}{c} (M, S) \xleftarrow{\$} \{0,1\}^\lambda \times \{0,1\}^s \\ Y \leftarrow H(M, S)\,;\ (M', S') \xleftarrow{\$} \mathsf{A}(Y) \end{array} : H(M', S') = Y\right]$$

Intuition: The adversary is given as a challenge only the hash value Y and has to come up with a valid tuple (M', S'), such that $H(M', S') = Y$.
Commitment schemes: commit to a value Y and later reveal M and S.

9. **Variant Two for Salted Preimage Security (**$s\mathrm{Pre}^2[\lambda, s]$**)**

$$\mathbf{Adv}_H^{s\mathrm{Pre}^2[\lambda,s]}(\mathsf{A}) = \Pr\left[\begin{array}{c} (M, S) \xleftarrow{\$} \{0,1\}^\lambda \times \{0,1\}^s \\ Y \leftarrow H(M, S)\,;\ (M', S') \xleftarrow{\$} \mathsf{A}(Y, S) \end{array} : H(M', S') = Y\right]$$

Intuition: The adversary is given as a challenge tuple the hash value Y and the salt S and has to come up with a valid tuple (M', S'), such that $H(M', S') = Y$ where either $S = S'$ or $S \neq S'$.
Applications: Salted password protection and commitment schemes: given S and Y an adversary has to come up with M', such that $H(M', S') = Y$ (here $S' = S$).

We say that A is (t, ϵ) X-secure if no adversary running in time at most t has advantage more than ϵ. When giving results in the random oracle model, we will talk about $(t, q_{\mathrm{RO}}, \epsilon)$ atk-secure schemes, where q_{RO} is the total number of queries that A makes to its random oracles.

Note 1: The notions that are parameterized by the λ message length do exhibit ideal security with respect to the relevant security notion only when $\lambda \geq n$. Therefore, in our security analysis we explicitly consider messages of length $\lambda \geq n$. (e.g. a more intuitive argument is that no ideal function exhibits Sec security of order 2^n when its domain size is smaller than its range size, namely n bits)

Note 2: Notice that in some security notions (Sec, Pre and variants of these) we make use of uniformly distributed target messages (Sec cases) or hash values that are the hash result from the computation over random messages (Pre cases). A popular technique in many standard reduction-based Sec and Pre security proofs makes use of the uniformity assumption on the input message string M in the following way. If a message block size substring m from M where $M \xleftarrow{\$} \{0,1\}^{\lambda}$ is substituted with a new random substring of the same size, then it is clear that as long as m is chosen uniformly at random and independently from M the new resulting message string M' (after the substitution) is also uniformly distributed over $\{0,1\}^{\lambda}$. This fact is used in some of our security proofs.

1.1 Additional Properties on the Compression Function F

In addition to the standard security properties (Coll, Sec and Pre) and the newly introduced salted security properties of hash functions we use two stronger variants of the Sec and Pre security notions. Let $\mathrm{F} : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ be a compression function with $b > 1$ and A an adversary against the *cp*Sec (chosen partial input second preimage security) and *cp*Pre (chosen partial input preimage security) of F. Then we measure the adversarial advantage in breaking these properties by the advantage functions $\mathbf{Adv}_{\mathrm{F}}^{cp\mathrm{Sec}}(\mathsf{A})$ and $\mathbf{Adv}_{\mathrm{F}}^{cp\mathrm{Pre}}(\mathsf{A})$, respectively. Let m_1 and m_2 be of lengths at most b and at least n bits respectively and $|m_1 + m_2| = b + n$. Let A be the compression function F adversary (stateful with a state St) for the indicated notion. The adversarial advantage for each of these security notions is measured as follows:

$$\mathbf{Adv}_{\mathrm{F}}^{cp\mathrm{Sec}}(\mathsf{A}) = \Pr\left[\begin{array}{c}(m_1) \xleftarrow{\$} \mathsf{A} \text{ where } |m_1| \leq b \\ m_2 \xleftarrow{\$} \{0,1\}^{\geq n} \\ m' \xleftarrow{\$} \mathsf{A}(m_2, St)\end{array} : \begin{array}{c} m' \neq (m_1, m_2) \text{ and} \\ \mathrm{F}(m_1, m_2) = \mathrm{F}(m') \text{ where } |m'| = b + n\end{array}\right]$$

$$\mathbf{Adv}_{\mathrm{F}}^{cp\mathrm{Pre}}(\mathsf{A}) = \Pr\left[\begin{array}{c}(m_1) \xleftarrow{\$} \mathsf{A} \text{ where } |m_1| \leq b \\ m_2 \xleftarrow{\$} \{0,1\}^{\geq n} \\ \mathrm{F}(m_1, m_2) = Y \; ; \; m' \xleftarrow{\$} \mathsf{A}(Y, St)\end{array} : \mathrm{F}(m') = Y \text{ where } |m'| = b + n\right]$$

The (m_1, m_2) input to F may not always be parsed as the straightforward concatenation of both strings. In some cases we would only require that the pair is a valid input string up to reordering, e.g. $m_1 = m_1' \| m_1''$ and (m_1, m_2) shall be parsed as $m_1' \| m_2 \| m_1''$. Normally, this is clear from the specification of the compression function.

Notice that as long as the size of m_2 is lower bound by the size of the output hash value n an adversary against *cp*Sec or *cp*Pre needs to perform 2^n evaluations of an ideal compression function F to find a second preimage or preimage message, respectively. If F is instantiated as $\mathrm{F} : \{0,1\}^{b+n+c} \rightarrow \{0,1\}^n$ with $b > 1$ and $c > 1$ then the lengths of m_2 and m_1 are bound as $|m_2| \geq n$ and $|m_1| \leq b + c - n$.

1.2 Min Entropy and Hash Function Balance

Definition 1. ***Min Entropy*** *The min-entropy of a discrete random variable X taking values on a finite set $S = \{x_1, \ldots, x_n\}$ where $p_1, \ldots, p_n$ are the corresponding outcome probabilities (or probability distribution*

over S) is $\mathbf{H}_{min}(X) = \min_{i=1}^{n}(-\log p_i) = -(\max_i \log p_i) = -\log \max_i p_i$. *A random variable has high(est) min entropy when the distribution over S is uniform, or for any i then* $\Pr[X = x_i] = p_i = 1/n$.

Definition 2. ***Min Entropy Preserving Function*** *Let* F *be a function with domain* $D = \{0,1\}^d$ *and range* $R = \{0,1\}^n$ *where* $d > n$. *Let* X *be the random variable taking values on* D *and* $\mathbf{H}_{min}(X) = k$ *and* $Y = F(X)$ *be the random variable taking values on* R *where* $n \leq k \leq d$. *Then we call the function* F *mapping* X *to* Y *a min entropy preserving function if* $\mathbf{H}_{min}(Y) = n$.

Definition 3. ***Hash Function Balance****: Let a hash function have* 2^n *range points* Y_i *(* $i = 1$ *to* 2^n*). Let* $H^{-1}(Y_i)$ *be the preimage of* Y_i *under* H. *Let* $|H^{-1}(Y_i)| = d_i$ *be the size of the preimage set of* Y_i *under* H *and let* d *be the size of the hash function domain* D. *The balance of the hash function* H *function is then defined as* $r(H) = \log_{2^n}[\frac{d^2}{d_1^2+\ldots d_{2^n}^2}]$.

When the balance of a function is one, then the hash function is called a regular function or $d_i = d/2^n$ for every i ($i = 1$ to 2^n). If X is a random variable taking values from the domain D and D_U is the uniform distribution over D, then the resulting distribution D_R on R by a regular function mapping X to Y with Y taking values on R is again the uniform distribution.

1.3 Some Conventions on the Modes of Operation:

The message M is parsed as $m_1 \| m_2 \| \ldots \| m_l$ where $|m_i| = b$ for $i = 1$ to $l - 1$ and $|m_l| = z$ with $z = (|M| \bmod b)$. Let S be a salt value of length s bits. With C_i where $|C_i| = c$ we denote a message bit counter value representing the binary encoding of the integer variable accounting for the number of bits hashed up to the i-th message block (including). The counter C is a multiple of the block size b when the message fills the whole message block, and not otherwise. If a final extra $(l+1)$st input block contains no message bits (e.g. due to some padding rules), then the counter is $C_{l+1} = 0^c$ (the concatenation of c 0 bits).

2 Counter-Based Merkle-Damgård Hash Function with Output Transformation ($\mathcal{LANE}$)

The counter-based Merkle-Damgård hash function with an output transformation ($\mathcal{LANE}_{\mathrm{F}}$) is a Merkle-Damgård [8, 5] based hash function that borrows a lot of its design characteristics from the HAIFA design [3].

$\mathcal{LANE}_{\mathrm{F}}$ makes use of a finite-domain compression function $\mathrm{F} : \{0,1\}^{n+b+c} \rightarrow \{0,1\}^n$. Let $\phi_i = \langle i \rangle_8$ be the binary encoding of $i \geq 0$ in eight bits. For $z = (|M| \mod b)$ then let pad be a padding function that takes as input the message M and returns the string $M \| 0^{b-z}$ if $z \neq 0$ and M if $z = 0$.

Let $\langle |M| \rangle_{64}$ is the 64-bit binary encoding of the message length (in bits). Let n be either 224, 256, 384 or 512 bits. The initial value IV_n is given by $IV_n = \mathrm{F}(0^n, \phi_2 \| \langle n \rangle_{32} \| 0^{b-40}, 0^c)$ where $\phi_2 = \langle 2 \rangle_8$.

In Figure 1 we describe the $\mathcal{LANE}_{\mathrm{F}}$ algorithm.

Algorithm 1 $\mathcal{LANE}_{\mathrm{F}}(M)$:

$m_1 \| \ldots \| m_{\ell-1} \leftarrow \mathsf{pad}(M)$
$h_0 = IV_n$; $C_\ell = 0^c$
for $i = 1$ to $\ell - 1$ **do**
  $h_i = \mathrm{F}(h_{i-1}, m_i, C_i)$
end for
$h_\ell = \mathrm{F}(h_{\ell-1}, \phi_0 \| \langle |M| \rangle_{64} \| 0^{b-72}, C_\ell)$
return h_ℓ

2.1 $\mathcal{LANE}$ Compared to Known Hash Functions

The main differences of the $\mathcal{LANE}$ scheme compared to the Merkle-Damgård design [8, 5] are:

1. IV_n **derivation:** Similarly to HAIFA [3], in $\mathcal{LANE}$ the initialization vector for distinct hash digest sizes is derived as $IV_n = \mathrm{F}(0^n, \phi_2 \| \langle n \rangle_{32} \| 0^{b-40}, 0^c)$. Thus, hashing messages with different digest sizes requires the use of distinct initialization vectors. This measure precludes attacks that try to find correlation on the hash outputs of messages hashed to distinct digest sizes, e.g. hashing two identical messages to different digest sizes results in uncorrelated digest values.
2. **Hash function inputs:**
   (a) *prefix-free* inputs to $\mathcal{LANE}$: If no input message M of length $|M| < |M'|$ is a prefix of a distinct input message $M' \neq M$, such that $M' = M \| S$, then the inputs are prefix-free.
   Now the hash function $\mathcal{LANE}$ processes a pad-padded message in a sequence of b blocks. Each block m_i is fed to the fixed-input length compression function F. But F also takes as inputs a chaining variable h_i and a counter C_i. For a computed and fixed IV_n value, the counter inputs to all of the compression functions used in the iteration are distinct. The inclusion of $C_\ell = 0^c$ counter and flag byte ϕ in the final block ensures the prefix-free property of the inputs to $\mathcal{LANE}$.
   (b) *suffix-free* inputs to $\mathcal{LANE}$: If no input message M of length $|M| < |M'|$ is a suffix of a distinct input message $M' \neq M$, such that $M' = P \| M$, then the inputs are suffix-free.
   We use a standard technique of adding the message length encoding in bits to provide a suffix-free property in the $\mathcal{LANE}$ hash function.
   (c) *distinct* inputs to F (*domain separation*): A distinctive feature of the $\mathcal{LANE}$ hash function borrowed from the HAIFA [3] design is the inclusion of a counter to every input of F. The counter values together with the addition of the byte ϕ input to the first and final F applications provides a *domain separation* to all compression functions in the iteration. More precisely, no two inputs to F in $\mathcal{LANE}$ repeat.
   Then, in some cases we make use of this fact and by $\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ we explicitly indicate the distinct calls to F needed for processing a padded message $m_1 \| \ldots \| m_{\ell-1} \leftarrow \mathsf{pad}(M)$.

3. **Output transformation:** The input $m_\ell = (\phi_0 \| \langle \lambda \rangle_{64} \| 0^{0^{b-72}})$ to the output transformation function in $\mathcal{LANE}$ is fixed. This measure strengthens the security of the $\mathcal{LANE}$ hash function by foiling any attacks that exploit differences on the last inputs blocks.

2.2 Collision Security of $\mathcal{LANE}$.

We prove the $\mathcal{LANE}$ hash function Coll secure under the standard Coll assumption on the compression function and derive a tight security bound. Rather than defining Coll security through the non-existence of an algorithm A, we follow Rogaway's human-ignorance approach [9] and use the Coll advantage function as a metric to relate the advantage of an adversary A against the hash function $\mathcal{LANE}_{\mathrm{F}}$ to that of an adversary B against the compression function F. The general proof follows the proof of the standard Merkle-Damgård [8, 5] design.

Theorem 1. *If there exists an explicitly given adversary* A *that* (t, ϵ)*-breaks the* Coll *security of* $\mathcal{LANE}_{\mathrm{F}}$*, then there exists an explicitly given adversary* B *that* (t', ϵ')*-breaks the* Coll *security of* F *for* $\epsilon' \geq \epsilon$ *and* $t' \leq t + 2\ell \cdot \tau_{\mathrm{F}}$*. Here,* τ_{F} *is the time required for the evaluation of* F *and* $\ell = \lceil \lambda / b \rceil + 1$ *where* λ *is the maximum message length of the two messages output by* A*.*

Proof. Given a Coll adversary A against the iterated hash $\mathcal{LANE}_{\mathrm{F}}$, we construct a Coll adversary B against the compression function F. B runs A on no inputs and A outputs a colliding pair (M, M'), such that $\mathcal{LANE}_{\mathrm{F}}(M) = \mathcal{LANE}_{\mathrm{F}}(M')$. We investigate the following two cases:

1. If $|M| \neq |M'|$, then the inputs to the last compression function differ (due to the present message length encoding in m_ℓ) and therefore a collision on the final F occurs, or $m_\ell \neq m'_{\ell'}$ where $\mathrm{F}(h_{\ell-1}, m_\ell, C_\ell) = \mathrm{F}(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$. B then outputs $(h_{\ell-1}, m_\ell, C_\ell)$ and $(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ as a valid colliding pair.
2. Else if $|M| = |M'|$, then $\ell' = \ell$, $m'_{\ell'} = m_\ell$ and $(h_{\ell-1}, m_\ell, C_\ell) = (h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$. Then $h'_{\ell'} = h_\ell$ and B proceeds in the following way.
   B parses the inputs to the $(\ell - 1)$st application of F as $(h_{\ell-2}, m_{\ell-1}, C_{\ell-1})$ and $(h'_{\ell-2}, m'_{\ell-1}, C_{\ell-1})$. If these inputs differ, then they constitute a valid collision pair for B. Else, B goes one step back. Following the iteration principle of $\mathcal{LANE}_{\mathrm{F}}$ B proceeds in the same manner backwards.
   The inequality of the message inputs M and M' guarantees the existence of an index $i > 0$, such that $(h_{i-1}, m_i, C_i) \neq (h'_{i-1}, m'_i, C_i)$ where $\mathrm{F}(h_{i-1}, m_i, C_i) = \mathrm{F}(h'_{i-1}, m'_i, C_i)$. B outputs then the colliding pair (h_{i-1}, m_i, C_i) and (h'_{i-1}, m'_i, C_i) for the $\max(i)$ satisfying the former statement.

B succeeds with the same advantage as A. The time complexity of B is at most the time complexity of A plus two evaluations of $\mathcal{LANE}_{\mathrm{F}}$ over messages M and M' taking time $2\ell \cdot \tau_{\mathrm{F}}$. □

2.3 Preimage Security of $\mathcal{LANE}$.

As mentioned in Section 2.1, the inclusion of counter C and ϕ values provides *domain separation* for every F in $\mathcal{LANE}_{\mathrm{F}}$. An alternative way to denote then $\mathcal{LANE}_{\mathrm{F}}$ is $\mathcal{LANE}_{\mathrm{F,G}}$ where we model the output transformation compression function as an instantiation of a distinct function G with the same domain and range size as F. Or $\mathcal{LANE}_{\mathrm{F,G}}(M) = \mathrm{G}(\mathcal{LANE}_{\mathrm{F}}^{-1}(M), \phi_0 \| \langle |M| \rangle_{64} \| 0^{b-72}, 0^c)$ where $\mathcal{LANE}_{\mathrm{F}}^{-1}$ is the iterative hash function $\mathcal{LANE}_{\mathrm{F}}$ up to the final output transformation function (excluding).

Then we exhibit a Pre security proof for $\mathcal{LANE}_{\mathrm{F,G}}$ modeling the output transformation function G as *cp*Pre secure function and the iterative function $\mathcal{LANE}_{\mathrm{F}}^{-1}$ as either a random oracle, regular function or a min entropy preserving function.

Theorem 2. *If the compression function* G *is* (t', ϵ') *cp*Pre *secure and* $\mathcal{LANE}_{\mathrm{F}}^{-1}$ *is either a random oracle, a regular function or a min entropy preserving function, then the iterated function* $\mathcal{LANE}_{\mathrm{F,G}}$ *is* $(t, \epsilon, q_{\mathrm{RO}})$ Pre$[\lambda]$ *secure for* $\epsilon \leq \epsilon'$ *and* $t \geq t' - \tau_{\mathcal{LANE}_{\mathrm{F}}^{-1}}$*. Here,* $q_{\mathrm{RO}} < 2^n$ *is the number of random oracle queries by the adversary on* $\mathcal{LANE}_{\mathrm{F,G}}$ *and* $\tau_{\mathcal{LANE}_{\mathrm{F}}^{-1}}$ *is the time required for the evaluation of* $\mathcal{LANE}_{\mathrm{F}}^{-1}$*.*

Proof. Given a Pre[λ] adversary A on $\mathcal{LANE}_{\mathrm{F,G}}$ and $\mathcal{LANE}_{\mathrm{F}}^{-1}$ modeled as either a random oracle, a min entropy preserving or a regular function, we build the *cp*Pre adversary to the final compression function G of $\mathcal{LANE}_{\mathrm{F,G}}$.

On known λ then B computes m_ℓ (the final message input block to G) where $m_\ell = \phi_0 \| \langle \lambda \rangle_{64} \| 0^{b-72}$. B outputs (m_ℓ, C_ℓ) where $C_\ell = 0^c$ and gets as a challenge the target hash value Y where $Y = \mathrm{G}(h_{\ell-1}, m_\ell, C_\ell)$. Note that $h_{\ell-1}$ is a random string of n bits.

Now, B sends Y to A. We argue that the distribution on Y induced by $\mathcal{LANE}_{\mathrm{F,G}}$ on input a message $M \xleftarrow{\$} \{0,1\}^\lambda$ chosen uniformly at random is identical to the distribution on Y induced by G on input $(h_{\ell-1}, m_\ell, C_\ell)$. Note that A expects $Y = \mathrm{G}(\mathcal{LANE}_{\mathrm{F}}^{-1}(M), m_\ell, C_\ell)$. But $\mathcal{LANE}_{\mathrm{F}}^{-1}$ is either a random oracle, a regular or min entropy preserving function and therefore it always outputs a random string $h_{\ell-1}$ on input a random message $M \xleftarrow{\$} \{0,1\}^\lambda$. Thus, we can conclude that $\mathcal{LANE}_{\mathrm{F}}^{-1}(M)$ has the same distribution as the original $h_{\ell-1}$ (the hidden challenge for B). To generate a preimage message M', algorithm A makes queries to the function $\mathcal{LANE}_{\mathrm{F,G}}$. In the case when $\mathcal{LANE}_{\mathrm{F}}^{-1}$ is modeled as a random oracle, then B simulates the random oracle responses by lazy sampling strings of n bits. If A succeeds and outputs a preimage M', then B also succeeds in finding the preimage $(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ where $h'_{\ell'-1} = \mathcal{LANE}_{\mathrm{F}}^{-1}(M')$. The time complexity of B is at most the time complexity of A plus one evaluation of $\mathcal{LANE}_{\mathrm{F}}^{-1}$ over message M' taking time $\tau_{\mathcal{LANE}_{\mathrm{F}}^{-1}}$. □

Notice that modeling only the $(\ell-1)$st compression function F as a random oracle or min entropy preserving function is sufficient to exhibit the same result.

2.4 Security Proofs of $\mathcal{LANE}$ in the Random Oracle Model.

If the compression function F is modeled as a random oracle, then the $\mathcal{LANE}_{\mathrm{F}}$ is Sec secure up to 2^n queries to F and Pre secure up to 2^{n+1} queries to F.

Our security claims in the random oracle model consider information theoretic adversaries. We give a lower bound on the attack complexity in terms of the number of queries q_{RO} made by the adversary to the random oracle. In our model, we assume that the compression function $\mathrm{F} : \{0,1\}^{b+n+c} \to \{0,1\}^n$ behaves as a random oracle. That means F is a publicly computable function chosen uniformly at random from the set of all functions with the respective domain and range space.

For the ease of exposition in our security proofs we will use the relevant domain separation notation for F in $\mathcal{LANE}_{\mathrm{F}}$ as indicated in Section 2.1. Thus, for a message of ℓ blocks we denote $\mathcal{LANE}_{\mathrm{F}}$ by $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$. Note that effectively there are $(\ell+1)$ evaluations of F but we do not account for the evaluation of F_0 because the IV_n value can be efficiently precomputed.

Theorem 3. *If the compression functions* F_i *for all* $i = 1$ *to* ℓ *are instantiated as random oracles, then the advantage of adversary* A *against the* Sec[λ] *security of* $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{\mathrm{Sec}[\lambda]}_{\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^n$ *where* q_{RO} *is the total number of queries to the random oracles* F_i *and* $\ell = \lceil \lambda/b \rceil + 1$.

Proof. Let A be a Sec[λ] adversary on the iterated hash function $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$. A is given a target message $M \xleftarrow{\$} \{0,1\}^\lambda$. The goal of A is to find a second preimage message M' for M, such that $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}(M) = \mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_{\ell'}}(M')$. Here $\ell = \lceil \lambda/b \rceil + 1$ and $\ell' = \lceil |M'|/b \rceil + 1$.

A is successful only when he makes consistent (with the iterative principle of $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$) queries to the random oracles F_i. Then to bound the maximal success probability of A we assume that A makes only such consistent queries to every F_i. Let us assume A outputs a valid second preimage message M'. Then we analyze the following cases:

1. If $|M| \neq |M'|$ then $\mathrm{F}_\ell(h_{\ell-1}, m_\ell, C_\ell) = \mathrm{F}_{\ell'}(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ and $m_\ell \neq m'_{\ell'}$ because these message blocks contain the binary encoding of the message lengths. The probability of finding a second preimage for the last input block $(h_{\ell-1}, m_\ell, C_\ell)$ is $1/2^n$ because $h_{\ell-1}$ is the random output of $\mathrm{F}_{\ell-1}$. However, to compute sufficient values $h'_{\ell'-1}$ under $\mathrm{F}_{\ell'}$ for the second preimage final block $(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ then A needs to make additionally at least that many evaluations of $\mathrm{F}_{\ell'-1}$ on inputs $(h'_{\ell'-2}, m'_{\ell'-1}, C_{\ell'-1})$. In this case A makes the least number of queries to F_i when $\ell' = 2$. Thus, the success probability of A in finding a second preimage for the last message block in q_{RO} queries to F_1 and $\mathrm{F}_{\ell'=2}$ is upper bound by $q_{\mathrm{RO}}/2^{n+1}$.

2. If $|M| = |M'|$ then we follow the argument of the standard collision security proof from Theorem 1. In this case A computes $h_{\ell-1} = \mathrm{F}_{\ell-1}(h_{\ell-2}, m_{\ell-1}, C_{\ell-1})$ which is the one but last chaining value of the original target message. Now, the optimal strategy for A is to search for a second preimage message m'_1 of a single block length, or that is the case when $\ell = \ell' = 2$. Then, all that A needs to do is to find m'_1, such that $\mathrm{F}_1(IV_n, m'_1, C_1) = h_1$ where h_1 is the one but last chaining value of the target message (as already computed $h_{\ell-1}$ with $\ell = 2$). Here the success probability of A for finding a second preimage colliding message is upper bound by $q_{\mathrm{RO}}/2^n$.

Now we argue that the optimal attack strategy for A is to find a second preimage for a single message block. The collision security preservation proof of the $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$ hash function shows that if two distinct messages M and M' hash to the same hash value, then a collision in at least one message block occurs. A collision in the Sec setting translates to A finding a second preimage for some F_i with $i \in \{1, \ldots, \ell\}$. However, all chaining variables h_{i-1}, but the first (which is treated as a constant for messages that hash to the same digest size), are output by the random oracle F_{i-1}. Thus, all h_{i-1}s are random strings of size n bits not known to A prior to the evaluation of the target message M. In the case of the evaluation of F_1 A gets a random message block m_1 (ensured by the fact that $M \xleftarrow{\$} \{0,1\}^\lambda$). As mentioned the counter C_i inclusion to every F_i provides random oracle domain separation. Or in order to find a single pair of colliding blocks under a fixed counter C_i value A needs to make 2^n random oracle queries to F_i. Then for a maximal number of q_{RO} queries the adversarial advantage is upper bound by $q_{\mathrm{RO}}/2^n$. □

Theorem 4. *If the compression functions* F_i *for all* $i = 1$ *to* ℓ *are instantiated as random oracles, then the advantage of adversary* A *against the* $\mathrm{Pre}[\lambda]$ *security of* $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{\mathrm{Pre}[\lambda]}_{\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^{n+1}$ *where* q_{RO} *is the total number of queries to the random oracles* F_i *and* $\ell = \lceil \lambda/b \rceil + 1$.

Proof. Let A be a $\mathrm{Pre}[\lambda]$ adversary on the iterated hash function $\mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$. Given a challenge hash value Y, the goal of A is to invert Y. Remember that Y is computed as $Y = \mathcal{LANE}_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}(M)$ for a message $M \xleftarrow{\$} \{0,1\}^\lambda$ chosen uniformly at random and $i = 1$ to $\ell = \lceil \lambda/b \rceil + 1$.

Adversary A's best strategy is to invert short messages to minimize the number of evaluations of F_i. Therefore, we analyze the adversarial advantage when $\ell' = 2$. In that case A first computes many values of h_1 which then would allow him verify if $h_2 = Y$. (An alternative strategy of the same complexity for A is to first invert Y to (h_1, m_2, C_2) for arbitrary h_1 and fixed $(m_2 = \phi_0 \| \langle |M'| \rangle_{64} \| 0^{b-72}, C_2 = 0^c)$, and then to invert h_1 to (IV_n, m_1, C_1) for some arbitrary M' where $m_1 = M' \| 0^{b-|M'|}$ with $|M'| \geq n$.) For a fixed precomputed initial vector A evaluates F_1 under different values for m_1 where $m_1 = M' \| 0^{b-|M'|}$ and $|M'| \geq n$. Thus, A exhaustively queries the random oracle F_1 on inputs (IV_n, m_1, C_1) to obtain distinct values for h_1. This subsequently allows A to compute $h_2 = \mathrm{F}(h_1, m_2, C_2)$ with $m_2 = \phi_0 \| \langle M' \rangle_{64} \| 0^{b-72}$ and $C_2 = 0^c$. The probability that h_2 equals the correct hash value Y in a single random oracle query to the final compression function F_2 is $1/2^n$. In total q_{RO} random oracle queries to F_1 and F_2 A succeeds to invert Y with probability $q_{\mathrm{RO}}/2^{n+1}$. As in the Sec security proof of Theorem 3 the domain separation (ensured by the counter inclusion) disallows A to reuse query/response under incorrect counter values. □

Long Message Sec Attacks on $\mathcal{LANE}$ The long message second preimage attacks of [7, 1] do not apply to $\mathcal{LANE}$ due to the counter inclusion. This result follows from Theorem 3. The problem for a second preimage adversary occurs in the linking step (dominant term in the attack complexity) of the attacks where the adversary does not know the correct counter value to connect to one of the chaining values of the original target message. Moreover, in the [7] attack scenario the adversary needs to produce expandable message which is not possible in the case of $\mathcal{LANE}$ due to the counter inclusion.

3 Randomized *LANE*

With *sLANE* we denote the randomized variant of the *LANE* hash function. *sLANE* also makes use of the finite-domain compression function $\mathrm{F} : \{0,1\}^{n+b+c} \to \{0,1\}^n$. Let S be a salt value of length $s = n$ and $\phi_i = \langle i \rangle_8$ again be the binary encoding of the integer i in eight bits. For $z = (|M| \mod b)$ the padding function pad (identically to *LANE*) takes as input the message M and returns the string $M \| 0^{b-z}$ if $z \neq 0$ and M if $z = 0$.

Let n be either 224, 256, 384 or 512 bits. The initial value IV_n is given by $IV_n = \mathrm{F}(0^n, \phi_3 \| \langle n \rangle_{32} \| 0^{b-s-40} \| S, 0^c)$. In Figure 2 we describe the *sLANE* algorithm.

Algorithm 2 $sLANE_{\mathrm{F}}(M)$:

$m_1 \| \ldots \| m_{\ell-1} \leftarrow \mathsf{pad}(M)$
$h_0 = IV_n$; $C_\ell = 0^c$
for $i = 1$ to $\ell - 1$ **do**
  $h_i = \mathrm{F}(h_{i-1}, m_i, C_i)$
end for
$h_\ell = \mathrm{F}(h_{\ell-1}, \phi_1 \| \langle |M| \rangle_{64} \| 0^{b-s-72} \| S, C_\ell)$
return h_ℓ

3.1 Collision security of *sLANE*.

The collision security proof is identical to the standard Merkle-Damgård proof but for the purpose of completeness we provide a rigorous analysis. Note that we use the salted *s*Coll security notion for the iterated hash function and make a standard Coll security assumption on the compression function F. Though the first and final blocks of the iterative hash function contain as input the salt value, these blocks are treated uniformly as a valid message input by F. Thus, it suffices to work with the non-salted standard Coll assumption on the compression function F.

Theorem 5. *If there exists an explicitly given adversary* A *that* (t, ϵ)*-breaks the* *s*Coll *security of* $sLANE_{\mathrm{F}}$*, then there exists an explicitly given adversary* B *that* (t', ϵ')*-breaks the* Coll *security of* F *for* $\epsilon' \geq \epsilon$ *and* $t' \leq t + 2\ell \cdot \tau_{\mathrm{F}}$*. Here,* τ_{F} *is the time required for the evaluation of* F *and* $\ell = \lceil \lambda / b \rceil + 1$ *where* λ *is the maximum message length of the two colliding messages output by* A*.*

Proof. Given a *s*Coll adversary A against the iterated hash $sLANE_{\mathrm{F}}$, we construct a Coll adversary B against the compression function F. B runs A on no inputs and A outputs a colliding pair (S, M) and (S', M'), such that $sLANE_{\mathrm{F}}(S, M) = sLANE_{\mathrm{F}}(S', M')$. We investigate the following two cases:

1. If $|M| \neq |M'|$ or $S \neq S'$, then the inputs to the last compression function differ and therefore a collision on the final F occurs, or $m_\ell \neq m'_{\ell'}$ where $\mathrm{F}(h_{\ell-1}, m_\ell, C_\ell) = \mathrm{F}(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$. B then outputs $(h_{\ell-1}, m_\ell, C_\ell)$ and $(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ as a valid colliding pair.
2. Else if $|M| = |M'|$ and $S = S'$, then $\ell' = \ell$. If $(h_{\ell-1}, m_\ell, C_\ell) \neq (h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ then a collision occurs again in the last application of F. Note that if $m_\ell = m'_{\ell'}$ then $S = S'$ and therefore $IV_n = IV'_n$. Thus, when $(h_{\ell-1}, m_\ell, C_\ell) = (h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ then B proceeds in the following way.
   B parses the inputs to the $(\ell - 1)$st application of F as $(h_{\ell-2}, m_{\ell-1}, C_{\ell-1})$ and $(h'_{\ell-2}, m'_{\ell-1}, C_{\ell-1})$. If these inputs differ, then they constitute a valid collision pair for B. Else, B goes one step back. Following the iteration principle B proceeds in the same manner backwards.
   The inequality of the message inputs M and M' guarantees the existence of an index $i > 0$, such that $(h_{i-1}, m_i, C_i) \neq (h'_{i-1}, m'_i, C_i)$ where $\mathrm{F}(h_{i-1}, m_i, C_i) = \mathrm{F}(h'_{i-1}, m'_i, C_i)$. B outputs then the colliding pair (h_{i-1}, m_i, C_i) and (h'_{i-1}, m'_i, C_i) for the $\max(i)$ satisfying the former statement.

Whenever A succeeds, then B also succeeds with the same advantage. The time complexity of B is at most the time complexity of A plus two evaluations of $sLANE_{\mathrm{F}}$ over messages M and M' taking time $2\ell \cdot \tau_{\mathrm{F}}$. □

3.2 *sw*Coll Security of *sLANE*

Theorem 6. *If there exists an explicitly given adversary* A *that* (t, ϵ)*-breaks the sw*Coll *security of* $sLANE_{\mathrm{F}}$*, then there exists an explicitly given adversary* B *that* (t', ϵ')*-breaks the* Coll *security of* F *for* $\epsilon' \geq \epsilon$ *and* $t' \leq t + 2\ell \cdot \tau_{\mathrm{F}}$*. Here,* τ_{F} *is the time required for the evaluation of* F *and* $\ell = \lceil \lambda/b \rceil + 1$ *where* λ *is the maximum message length of the two messages output by* A*.*

Proof. The proof follows the analysis of the proof from Theorem 5. The only difference here is that the Coll adversary B on F runs the *sw*Coll adversary of *sLANE* on input a random message string M where $M \xleftarrow{\$} \{0,1\}^{\lambda}$. The rest of the proof follows from the proof of Theorem 5. □

3.3 *sLANE* Security Proofs in the Random Oracle Model.

If the compression function is modeled as random oracle, then the *sLANE* is sSec secure in up to 2^n random oracle queries and $s\mathrm{Pre}^1$ and $s\mathrm{Pre}^2$ secure in up to 2^{n+1} random oracle queries. The latter results follow from the proofs of Theorems 3 and 4, respectively.

Again we use the relevant domain separation notation outlined in section 2.1, which is also true in the case of $sLANE_{\mathrm{F}}$. Thus, for a message of ℓ blocks we denote $sLANE_{\mathrm{F}}$ by $sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$. Note that due to the salt inclusion, here, as opposed to *LANE*, we need to evaluate F_0 every time a new message is hashed.

Theorem 7. *If the compression functions* F_i *are instantiated as random oracles for all* $i = 0$ *to* ℓ*, then the advantage of adversary* A *against the* $s\mathrm{Sec}[\lambda, s]$ *security of* $sLANE_{\mathrm{F}_1,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{s\mathrm{Sec}[\lambda,s]}_{sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^n$ *where* q_{RO} *is the total number of queries to the random oracles* F_i *and* $\ell = \lceil \lambda/b \rceil + 1$*.*

Proof. Let A be a $s\mathrm{Sec}[\lambda, s]$ adversary on the iterated hash function $sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$. A is given target random message $M \xleftarrow{\$} \{0,1\}^{\lambda}$ and salt $S \xleftarrow{\$} \{0,1\}^{s}$. The goal of A is to find a second preimage pair (M', S') for (M, S) with $M' \neq M$, such that $sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}(M, S) = sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_{\ell'}}(M', S') = Y$ with $\ell' = \lceil |M'|/b \rceil + 1$. A maximizes his advantage only when he makes consistent (with the iterative principle of $sLANE_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$) queries to the random oracles F_i. Then to bound the maximal success probability of A we assume that A makes only such consistent queries to F_i. Let A output a valid second preimage pair (M', S').

We analyze the following cases:

1. If $|M| \neq |M'|$ or $S \neq S'$ then $\mathrm{F}_\ell(h_{\ell-1}, m_\ell, C_\ell) = \mathrm{F}_{\ell'}(h'_{\ell'-1}, m'_{\ell'}, C_{\ell'})$ and $m_\ell \neq m'_{\ell'}$ because these message blocks contain the binary encoding of the message lengths and the salt values. The probability of finding a second preimage for the last input block $(h_{\ell-1}, m_\ell, C_\ell)$ is $1/2^n$ for a single evaluation of $\mathrm{F}_{\ell'}$ because the $h_{\ell-1}$ part of the target block is the random output of $\mathrm{F}_{\ell-1}$ and not known to A prior to obtaining the message M. However, to compute a second preimage colliding block $(h'_{\ell'-1}, m_{\ell'}, C_{\ell'})$ under $\mathrm{F}_{\ell'}$, then B needs to make consistent query evaluations of $\mathrm{F}_{\ell'-1}$. The argument goes the same way backwards. Thus, A minimizes its query complexity when $\ell' = 2$.
   Now if the the final message blocks m_ℓ (of M) and m'_2 (of M') collide because $S \neq S'$, then in that case A evaluates F_1 on inputs (IV_n, m'_1, C_1) for $S' \neq S$ to get sufficient values for h'_1 which would then would be used for the computation of $h'_2 = \mathrm{F}_2(h_1, m_2, C_2)$. If $h'_2 = Y$ then A is successful. (Note that the alternative strategy of computing new values for IV_n under distinct S' salts and then evaluating the final compression function under these salts is less efficient for A than the one described here). Thus, in this case the success probability of A in finding a second preimage block for the last message block in q_{RO} queries to F_1 and F_2 is upper bound by $q_{\mathrm{RO}}/2^{n+1}$.
   When $S = S'$ and $|M| \neq |M'|$ then A has a fixed salt value in which case we repeat the argument of the proof of Theorem 3 for messages of distinct lengths where the adversarial advatage is also upper bound by $q_{\mathrm{RO}}/2^{n+1}$.
2. If $|M| = |M'|$ and $S = S'$ then we follow the argument of the standard collision security proof from Theorem 5. That is, a collision in at least one message block of M and M' occurs in $sLANE_{\mathrm{F}}$ because $M \neq M'$. Notice that here the IV_n value is fixed for A ($S' = S$) and the security argument follows the proof of Theorem 3 (when $|M| = |M'|$).

□

Theorem 8. *If the compression functions* F_i *are instantiated as random oracles for all* $i = 0$ *to* $\ell = \lceil \lambda/b \rceil + 1$*, then the advantage of adversary* A *against the* $\mathrm{eSec}[\lambda, s]$ *security of* $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{\mathrm{eSec}[\lambda,s]}_{s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^n + 1/2^{n-t}$ *and* q_{RO} *is the number of queries to the random oracles* F_i *and* $2^t < 2^n$*. Here, we allow the adversary precomputation efforts but bound them by* 2^n *evaluations of* F_i *for all* $i \in \{0, \ldots, \ell\}$*.*

Proof. Let A be an $\mathrm{eSec}[\lambda, s]$ adversary on the iterated hash function $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$. A outputs a target message M. Then A gets a random target salt $S \xleftarrow{\$} \{0,1\}^s$. The goal of A is to find a second preimage pair (M', S') for M, such that $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}(M, S) = s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_{\ell'}}(M', S')$. Again A can only be successful by making consistent (with the iterative principle of $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$) queries to the random oracle F. Then to bound the maximal success probability of A we assume that A makes only such consistent queries to F_i. Let us assume A outputs a valid second preimage pair (M', S').

According to the collision security proof of $s\mathcal{LANE}_{\mathrm{F}}$ if the pair (M', S') is a valid second preimage for (M, S) then a collision occurs for at least one input block of the evaluation of $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ and $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_{\ell'}}$.

Note that A makes the minimum evaluations on F_is when both messages M and M' are of single block length. Therefore, we investigate the case when $M = m_1$ and $M = m'_1$ for which we get the tightest security bound.

A follows different strategies to make use of precomputation:

1. Precomputation of the initial variable IV_n under distinct salt values S'. Here A produces 2^t colliding messages under distinct salts S' and S'', such that $\mathrm{F}_1(\mathrm{F}_0(0^n, \phi_3 \| \langle n \rangle_{32} \| 0^{b-s-40} \| S', C_0), m_1, C_1) = \mathrm{F}_1(\mathrm{F}_0(\phi_3 \| \langle n \rangle_{32} \| 0^{b-s-40} \| S', C_0), m'_1, C_1)$ in approximately $2^{t/2+n/2} < 2^n$ steps (evaluations of F_0 and F_1). Note that with the exception of S' and S'' all the input values to F_0 and F_1 are fixed. Then A outputs m_1 and subsequently gets the target salt S value. A succeeds when $S' = S$ with probability $1/2^{n-t}$. If $S \neq S'$ then A proceeds as shown in Theorem 7. In total, here the advatage of A is bound by $1/2^{n-t} + q_{\mathrm{RO}}/2^n$ with $t < n$ (by our precomputation bound).
2. A precomputes a collision leading to identical chaining value h_1 under distinct message blocks m_1 and m'_1. Let us assume that A stores a table of queries and responses $((IV^i_n, m_1, C_1), h^i_2)$ and $((IV^i_n, m'_1, C_1), h^i_2)$ for $i = 1$ to 2^t, such that $\mathrm{F}_1(IV^i_n, m_1, C_1) = \mathrm{F}_1(IV^i_n, m'_1, C_1)$. Note that to find such colliding pairs A makes at least $2^{n/2+t} < 2^n$ evaluations of F_1. In this case A outputs m_1.
   Now, A on input S has to compute $IV_n = \mathrm{F}(0^n, \phi_3 \| \langle n \rangle_{32} \| 0^{b-s-40} \| S, 0^c)$. If $IV_n \in \{IV^1_n, \ldots, IV^{2^t}_n\}$ then A has found a second preimage message for m_1 and outputs m'_1 and the same salt $S' = S$. However, if A has precomputed 2^t colliding messages, the probability that $IV_n \in \{IV^1_n, \ldots, IV^{2^t}_n\}$ is $1/2^{n-t}$. If $IV_n \notin \{IV^1_n, \ldots, IV^{2^t}_n\}$ then A has to do either: (i). 2^n evaluations of F_1 under and a new valid m'_1 and 2^n evaluations of F_2 under some $S' \neq S$; (ii). 2^n evaluations of F_1 for a new valid m'_1 under the same $S' = S$, such that $\mathrm{F}_1(IV_n, m'_1, C_1) = h_1$ where $\mathrm{F}_2(h_1, \phi_1 \| \langle |m'_1| \rangle_{64} \| 0^{b-s-70} \| S, 0^c) = s\mathcal{LANE}_{\mathrm{F}_0,\mathrm{F}_1,\mathrm{F}_2}(m_1, S)$.
   Thus, in this case A succeeds either to find a second preimage message under a precomputed salt $S' = S$ with probability $1/2^{n-t}$ or succeeds under the same salt value $S' = S$ with probability $q_{\mathrm{RO}}/2^n$. Using this strategy A's advantage is bound by $(1/2^{n-t} + q_{\mathrm{RO}}/2^n)$ where $2^t < 2^{n/2}$.

In case A does not do precomputations the best strategy for A is to act as described in (i). and (ii). subcases of the latter case 2. and therefore A's advantage is bound by $q_{\mathrm{RO}}/2^n$.

Theorem 9. *If the compression functions* F_i *are instantiated as random oracles for all* $i = 0$ *to* ℓ*, then the advantage of adversary* A *against the* $s\mathrm{Pre}^1[\lambda, s]$ *security of* $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{s\mathrm{Pre}^1[\lambda,s]}_{s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^{n+1}$ *where* q_{RO} *is the number of queries to the random oracles* F_i *and* $\ell = \lceil \lambda/b \rceil + 1$*.*

Proof. Let A be a $s\mathrm{Pre}^1[\lambda, s]$ adversary on the iterated hash function $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$. Given a challenge hash value Y, the goal of A is to invert Y and output some valid preimage pair (M', S') . Remember that Y is computed as $Y = s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}(M, S)$ for randomly chosen message $M \xleftarrow{\$} \{0,1\}^\lambda$ and salt $S \xleftarrow{\$} \{0,1\}^s$.

As in the proof Theorem 4 the best strategy for A is to first compute many h_1 which then would allow him to verify if $h_2 = Y$. Note that A needs to always make at least two evaluations of F_1 and F_2 to verify if $h_2 = Y$ (this is the case when the adversary tweaks only message m'_1 values and not salt values). Here, as compared to the proof Theorem 4, A needs to make an evaluation of IV_n for some randomly chosen S'. A is most efficient if only a single random oracle query to F_0 is made for the evaluation of IV_n and this way S' and IV_n are fixed. If, alternatively, A decides to arbitrarily choose and modify (tweak) the values of S', then he needs to make an additional evaluation F_0 to compute IV_n every time he needs new values for h_2. Thus, with a single query overhead compared to the proof of Theorem 4, A proceeds the same way as showed already in the proof of Theorem 4.

For the computed initial vector IV_n and fixed S' A tries different values for m'_1 where $m'_1 = M' \| 0^{b-|M'|}$ and $|M'| \geq n$. A exhaustively queries the random oracle F_1 on inputs (IV_n, m'_1, C_1) (modifying m'_1 every time) to obtain new values for h_1. This allows A to compute $h_2 = \mathrm{F}(h_1, m'_2, C_2)$ with fixed values $m'_2 = \phi_1 \| \langle M' \rangle_{64} \| 0^{b-s-72} \| S'$ and $C_2 = 0^c$. The probability that h_2 equals the correct hash value Y in a single random oracle query to the final compression function F_2 is $1/2^n$. In q_{RO} random oracle queries to F_1 and F_2 B succeeds to invert Y with probability $q_{\mathrm{RO}}/2^{n+1}$.

Note that here the domain separation (ensured by the counter and ϕ value inclusion) of every compression function does not allow A to reuse query/response pairs obtained by evaluations of distinct F_is (e.g. F_1 and F_2). □

Theorem 10. *If the compression functions* F_i *are instantiated as random oracles for all* $i = 0$ *to* ℓ*, then the advantage of adversary* A *against the* $s\mathrm{Pre}^2[\lambda, s]$ *security of* $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ *is* $\mathbf{Adv}^{s\mathrm{Pre}^2[\lambda,s]}_{s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}}(A) \leq q_{\mathrm{RO}}/2^{n+1}$ *where* q_{RO} *is the number of queries to the random oracles* F_i *and* $\ell = \lceil (\lambda)/b \rceil + 1$*.*

Proof. Let A be a $s\mathrm{Pre}^2[\lambda, s]$ adversary on the iterated hash function $s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}$ for $i = 0$ to ℓ. Given a challenge hash value Y and a salt S, the goal of A is to invert Y and output some valid preimage pair (M', S'). Remember that Y is computed as $Y = s\mathcal{LANE}_{\mathrm{F}_0,\ldots,\mathrm{F}_\ell}(M, S)$ for randomly chosen message $M \xleftarrow{\$} \{0,1\}^\lambda$ and salt $S \xleftarrow{\$} \{0,1\}^s$.

In the case when $S = S'$ then the proof follows the proof of Theorem 4. When $S \neq S'$, in which case A chooses arbitrarily S', the proof follows from the proof of Theorem 9.

Long Message Sec **Attacks on** $s\mathcal{LANE}$. Similarly to the basic underlying $\mathcal{LANE}$ hash function, the attacks of [7, 1] do not apply on $s\mathcal{LANE}$.

4 Prefix-free Guarantees and its Security Implications

Both $\mathcal{LANE}$ and $s\mathcal{LANE}$ hash designs ensure the prefix-free property on its inputs. Following the work of Coron et al. [4], we conclude that $\mathcal{LANE}$ guarantees indifferentiability from a random oracle when the underlying compression function is a publicly accessible random oracle. This result is an indication of the security of $\mathcal{LANE}$ and $s\mathcal{LANE}$ against extension attacks and lack of structural design flaws.

If on the other hand, if the compression function is modeled as a pseudorandom function keyed through the chaining values, then according to Bellare et al. [2], a prefix-free Merkle-Damgård domain extender (in this case $\mathcal{LANE}$ and $s\mathcal{LANE}$) is also a provably secure pseudorandom function.

5 Parallelizable Mode of Operation

Much of the design characteristics of the parallel mode *pMode* are borrowed from the work of Damgård [5]. Let $P_1, \ldots, P_{2^t}$ be $T = 2^t$ processors. The parallelizable mode of operation *pMode* makes use of a finite-domain hash functions $H_i : \{0,1\}^{\leq L} \rightarrow \{0,1\}^n$ for $i = 1$ to 2^t+1. The number of processors available for computation 2^t defines the degree(level) of parallelization of the *pMode*. We define a message padding function ppad that on inputs the degree of free parallelism $T = 2^t$ and the message $M = m_1 \| \ldots \| m_{l-1} \| m_l$ with $|m_i| = b$ for $i = 1$ to $l-1$ and $|m_l| = (M \mod b)$ returns the string $M' = M_1 \| \ldots \| M_{2^t}$. Here we explicitly assume that the interleave factor b_{int} that determines the block size in which the message will be split is b bits. Thus, for $i = 1$ to 2^t then $M_i = m_i \| m_{i+2^t} \| m_{i+2^{t+1}} \| \ldots \| m_{i+2^{t+x-1}} \| m_{i+2^{t+x}}$ where $x = \lfloor (l/2^t) \rfloor$. In Figure 3 we describe the *pMode* algorithm.

Algorithm 3 $pMode_{H_1,\ldots,H_{2^t+1}}(M, 2^t)$:

$M_1 \| \ldots \| M_{2^t} \leftarrow \mathsf{ppad}(M)$
$C \leftarrow \varepsilon$
for $i = 1$ to 2^t (each P_i computes) **do**
  $h_i = H_i(M_i)$
  $C \leftarrow C \| h_i$
end for
$h_{2^t+1} = H_{2^t+1}(C)$
return h_{2^t+1}

Parallelism: The algorithm *pMode* has a 2^t degree of parallelism when 2^t processors are available. For $i = 1$ to 2^t each processor P_i processes the M_ith chunk of the original message with the hash function H_i. The parallelization degree 2^t is an input parameter of the scheme in the preprocessing part to specify the concrete message blocks that each processor needs to fetch in the computation. The hash values resulting from the computation of each processor are parsed sequentially as a concatenation of strings and input to the final hash function H_{2^t+1}.

Note that the $pMode_{H_1,\ldots,H_{2^t+1}}$ (when all H_i are Merkle-Damgård iterative hash functions) has the additional overhead of processing up to $2^t(b+n+64)$ extra message bits compared to a single Merkle-Damgård hash function processing messages of the same size. This is due to the inclusion of the initial vectors and standard MD padding in the computation of H_is over message parts M_is. This feature, on the positive side, allows the standard use of any hash function on a single processor level. Moreover, the hash values resulting from the computation of every processor are also processed by any chosen hashing method in the final phase. For messages of length $2^{t+x}+1$ the speedup of the parallel hashing compared to sequential processing is with a factor of approximately $2^x/x$.

5.1 Collision Security of *pMode*

Theorem 11. *If there exists an explicitly given adversary* A *that* (t, ϵ)*-breaks the* Coll *security of* $pMode_{H_1,\ldots,H_{2^t+1}}$*, then there exists an explicitly given adversary* B *that* (t', ϵ')*-breaks the* Coll *security of at least one of the* H_i *underlying hash functions where* $i \in \{1, \ldots 2^t+1\}$ *and* $\epsilon' \geq \epsilon$ *and* $t' \leq t + 2\sum_{i=1\ldots 2^t+1} \tau_{H_i}$*. Here,* τ_{H_i} *is the time required for the evaluation of* H_i *and* 2^t *is the maximal parallelization degree of the two colliding messages.*

Proof. Given a Coll adversary A against the hash function $pMode_{H_1,\ldots,H_{2^t+1}}$, we construct a Coll adversary B against the hash function H_i for some $i \in \{1, \ldots, 2^t+1\}$. B runs A on no inputs and A outputs a colliding pair (M, M') with respective parallelization degrees 2^t and $2^{t'}$, such that $pMode_{H_1,\ldots,H_{2^t+1}}(M, 2^t) = Y$ and

$pMode_{H_1,\ldots,H_{2^{t'}+1}}(M', 2^{t'}) = Y$. Let $M_1\|\ldots\|M_{2^t} \leftarrow \mathsf{ppad}(M, 2^t)$ and $M'_1\|\ldots\|M'_{2^{t'}} \leftarrow \mathsf{ppad}(M', 2^{t'})$. Then let $C = h_1\|h_2\|\ldots\|h_{2^t}$ and $C' = h'_1\|h'_2\|\ldots\|h'_{2^{t'}}$ where $h_i = H_i(M_i)$ for $i = 1$ to 2^t and $h'_j = H_j(M'_j)$ for $j = 1$ to $2^{t'}$. Then we know that $H_{2^t+1}(C) = H_{2^{t'}+1}(C')$.

If $2^t \neq 2^{t'}$, then $|C| \neq |C'|$ and therefore $C \neq C'$. Thus, a collision occurs and B outputs C and C' as a valid colliding pair. If $2^t = 2^{t'}$ and $C \neq C'$ then we repeat the latter argument. Else, if $C = C'$ then a collision has occurred in at least one H_i with $i = 1$ to 2^t. Note that when $C = C'$ then $|C| = |C'|$ and $2^{t'} = 2^t$. Then, we have that $H_i(M_i) = H_i(M'_i)$ for all $i = 1$ to 2^t and since $M \neq M'$ a collision occurs in at least one of the applications of H_i. Whenever A succeeds, then B succeeds to find a collision for at least one H_i with the same advantage. The time complexity of B is at most the time complexity of A plus two evaluations of all H_i taking time $2\sum_{i=1\ldots 2^t+1}\tau_{H_i}$. □

5.2 Second Preimage and Preimage Security of *pMode*

To exhibit Sec and Pre security proofs we build the respective Sec and Pre adversaries on the final hash function H_{2^t+1} in $pMode_{H_1,\ldots,H_{2^t+1}}$. Note, however, that a Sec adversary on H_{2^t+1} has to find a second preimage for $C = h_1\|\ldots\|h_{2^t}$ computed for a random input message $M \xleftarrow{\$} \{0,1\}^\lambda$ and random parallelization degree $2^t \xleftarrow{\$} \{2^0, 2^1, \ldots, 2^L\}$ (for practical reasons we upper bound the parallelization degree by 2^L). Here $h_i = H_i(M_i)$ for $i = 1$ to 2^t and $M_1\|\ldots\|M_{2^t} \leftarrow \mathsf{ppad}(M, 2^t)$. Now for an ideal hash function H_{2^t+1} the computational complexity of a second preimage search is of order 2^n when $\sum_{i=1,\ldots,2^t}(\mathbf{H}_{min}(h_i)) \geq n$. Note that the latter argument is valid only under the assumption that a second preimage adversary on H_{2^t+1} gets the message M chosen uniformly at random and computes its own target message $C = h_1\|\ldots\|h_{2^t}$. Thus, if for $i = 1$ to 2^t on average every H_i with input a random (and unpredictable to the adversary) message block M_i outputs h_i where $\mathbf{H}_{min}(h_i) \geq n/2^t$, then we can assume that the second preimage complexity for the adversary on H_{2^t+1} is of order 2^n. We call such instantiations of H_i $n/2^t$ min entropy preserving functions. The same argument holds for the Pre adversary on H_{2^t+1}.

Theorem 12. *If each hash function H_i is (t'_i, ϵ'_i)* Sec *secure (respectively) and $n/2^t$ min entropy preserving for all $i \in \{1, \ldots, 2^t\}$ and H_{2^t+1} is also (t'', ϵ'')* Sec *secure, then the hash function $pMode_{H_1,\ldots,H_{2^t+1}}$ is (t, ϵ) Sec secure (in the latter defined sense) for $\epsilon \leq \sum_{i=1,\ldots,2^t}(\epsilon'_i) + \epsilon''$ and $t' \leq t + 3\sum_{i=1\ldots 2^t+1}\tau_{H_i}$. Here, τ_{H_i} is the time required for the evaluation of H_i and 2^t is the maximal parallelization degree of the two colliding messages.*

Proof. Given a Sec$[\lambda, 2^t]$ (note that here the security notion is parameterized also by the parallelization degree) adversary A against the hash function $pMode_{H_1,\ldots,H_{2^t+1}}$, we construct the following Sec adversaries: B_1 against the hash function H_{2^t+1} and B_2 against either hash function H_i for some $i \in \{1, \ldots, 2^t\}$.

B_1 gets a random challenge message $M \xleftarrow{\$} \{0,1\}^\lambda$ and $2^t \xleftarrow{\$} \{2^0, 2^1, \ldots, 2^L\}$. B_1 computes $C = h_1\|\ldots\|h_{2^t}$ and has to find a second preimage message C' for C. On the same inputs $(M, 2^t)$ B_2's goal is to find a second preimage for at least one M_i under H_i where $M_1\|\ldots\|M_{2^t} \leftarrow \mathsf{ppad}(M, 2^t)$. Now B_1 and B_2 run A on inputs M and 2^t. A outputs a second preimage message M' with respective parallelization degree $2^{t'}$, such that $pMode_{H_1,\ldots,H_{2^t+1}}(M, 2^t) = Y$ and $pMode_{H_1,\ldots,H_{2^{t'}+1}}(M', 2^{t'}) = Y$. Let $M'_1\|\ldots\|M'_{2^{t'}} \leftarrow \mathsf{ppad}(M', 2^{t'})$. Then let $C' = h'_1\|h'_2\|\ldots\|h'_{2^{t'}}$ where $h'_j = H_j(M'_j)$ for $j = 1$ to $2^{t'}$. Then we know that $H_{2^t+1}(C) = H_{2^{t'}+1}(C')$.

Following the arguments from the collision security proof of *pMode* then we have the following scenarios: 1. a collision is present in either H_{2^t+1} or else, 2. a collision is present for at least one application of H_i. A collision in this setting translates to a valid second preimage message and therefore when A succeeds, then either B_1 or B_2 win. □

Theorem 13. *If the hash function H_{2^t+1} is (t', ϵ')* Pre *secure and each H_i hash function is $2^t/n$ min entropy preserving hash functions for all $i \in \{1, \ldots, 2^t\}$, then the hash function $pMode_{H_1,\ldots,H_{2^t+1}}$ is (t, ϵ) Pre secure for $\epsilon \leq \epsilon'$ and $t' \leq t + 2\sum_{i=1\ldots 2^t}\tau_{H_i}$. Here, τ_{H_i} is the time required for the evaluation of H_i and 2^t is maximal the parallelization degree of either preimage message(s) M and M'.*

Proof. Given a $\mathrm{Pre}[\lambda, 2^t]$ (note that here the security notion is parameterized also by the parallelization degree) adversary A against the hash function $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$, we construct the Pre adversary B against the hash function H_{2^t+1}. B gets a challenge hash value Y computed on inputs random $M \xleftarrow{\$} \lambda$ and $2^t \xleftarrow{\$} \{2^0, 2^1, \ldots, 2^L\}$ as $Y = H_{2^t+1}(C)$ where $C = h_1 \| h_2 \| \ldots \| h_{2^t}$ and $h_i = H_i(M_i)$ for $i = 1$ to 2^t. The $2^t/n$ min entropy assumption on all H_i with $i \in \{1, \ldots, 2^t\}$ ensures that B can not succeed in inverting Y faster than exhaustive search.

Now B runs A on input Y. Note that the expected distribution of Y from A is identical. Then A outputs a preimage message M' and the respective parallelization degree $2^{t'}$. B computes $C' = h'_1 \| \ldots \| h'_{2^{t'}}$ where $h'_j = H_j(M'_j)$ for $j = 1$ to $2^{t'}$. C' is a valid preimage for B. When A succeeds in finding a preimage, then B also succeeds. □

5.3 *pMode* Security Proofs in the Random Oracle Model.

If the hash functions $H_1, \ldots, H_{2^t+1}$ are modeled as random oracles, then the $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$ is Sec and Pre secure in up to 2^{n-t} and 2^n, respectively, total number of queries to the random oracles $(H_1, \ldots, H_{2^t+1})$.

Theorem 14. *If all the hash functions $H_1, \ldots, H_{2^t+1}$ are instantiated as random oracles, then the advantage of adversary A against the $\mathrm{Sec}[\lambda, 2^t]$ security of $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$ is $\mathbf{Adv}^{\mathrm{Sec}[\lambda,2^t]}_{p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}}(A) \leq q_{\mathrm{RO}}/2^n$ and q_{RO} is the total number of queries to the random oracles $(H_1, \ldots, H_{2^t+1})$.*

Proof. Let A be a $\mathrm{Sec}[\lambda, 2^t]$ adversary on the hash function $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$. A is given a random target message $M \xleftarrow{\$} \{0,1\}^\lambda$ and $2^t \xleftarrow{\$} \{2^0, 2^1, \ldots, 2^L\}$. The goal of A is to find a second preimage message M' for M. Note, however, that A can be successful only when he makes consistent (with the *pMode* iterative principle) queries to the random oracles $(H_1, \ldots, H_{2^t+1})$. Then to bound the maximal success probability of A we assume that A makes only this type of queries to $(H_1, \ldots, H_{2^t+1})$. Let us assume A outputs a valid second preimage message M' with respective parallelization degree $2^{t'}$. Then $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}(M, 2^t) = Y$ and also $p\mathit{Mode}_{H_1,\ldots,H_{2^{t'}+1}}(M', 2^{t'}) = Y$. Let $M'_1 \| \ldots \| M'_{2^{t'}} \leftarrow \mathsf{ppad}(M', 2^{t'})$. Then let $C = h_1 \| h_2 \| \ldots \| h_{2^{lt}}$ where $h_i = H_i(M_i)$ for $i = 1$ to 2^t and $C' = h'_1 \| h'_2 \| \ldots \| h'_{2^{t'}}$ where $h'_j = H_j(M'_j)$ for $j = 1$ to $2^{t'}$. Then we know that $H_{2^t+1}(C) = H_{2^{t'}+1}(C')$.

Now, according to the collision security proof when $H_{2^t+1}(C) = H_{2^{t'}+1}(C')$ then a collision has occurred for some H_i with $i \in \{1, \ldots, 2^t + 1\}$. Let us observe the type of RO queries A needs to make to cause such collisions. To find a second preimage colliding message for C then A needs 2^n evaluations of the hash function $H_{2^{t'}+1}$. To find the second preimage C' under $H_{2^{t'}+1}$, however, A needs to make consistent queries. Thus, A needs to evaluate 2^n times some H_i for $i \in \{1, \ldots, 2^{t'}\}$ (to obtain new values for C'). Thus, A succeeds to find second preimage C' for C under $H_{2^{t'}+1}$ in 2^n queries to $H_{2^{t'}+1}$ and 2^n queries to some H_i for $i \in \{1, \ldots, 2^{t'}\}$.

Another way for A to proceed is to find a second preimage M'_j for some target message chunk M_i under H_i where $C' = h'_1 \| \ldots \| h'_{\ell'}$ equals to $C = h_1 \| \ldots \| h_\ell$. In that case A uses as target points of either $h_1, \ldots, h_\ell$ resulting from the hashing of the original message. In this case A needs to make 2^n evaluations of H_i. Then for distinct (domain separated) instantiations of $(H_1, \ldots, H_{2^{t'}+1})$ the success probability of A is upper bound by $q_{\mathrm{RO}}/2^n$. □

Theorem 15. *If the hash functions $H_1, \ldots, H_{2^t+1}$ are instantiated as random oracles, then the advantage of adversary A against the $\mathrm{Pre}[\lambda, 2^t]$ security of $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$ is $\mathbf{Adv}^{\mathrm{Sec}[\lambda]}_{p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}}(A) \leq q_{\mathrm{RO}}/2^{n+1}$ and q_{RO} is the is the total number of queries to the random oracles $(H_1, \ldots, H_{2^t+1})$.*

Proof. Let A be a $\mathrm{Pre}[\lambda, 2^t]$ adversary on the tree hash function *pMode*. Given a challenge hash value Y. The goal of A is to invert Y, which is computed as $Y = H_{2^t}(C)$ as specified in the $p\mathit{Mode}_{H_1,\ldots,H_{2^t+1}}$.

A queries the random oracles $(H_1, \ldots, H_{2^{t'}})$ to get $C' = h'_1, \ldots, h'_{2^{t'}}$. This costs in total $2^{t'}$ evaluations of all H_is. Then A computes $Y = H_{2^{t'}+1}(h'_1 \| h_2 \| \ldots \| h'_{2^{t'}})$. Now after evaluating all $(H_1, \ldots, H_{2^{t'}+1})$ the probability in a single query to $H_{2^{t'}+1}$ that $H_{2^{t'}+1}(C') = Y$ is $1/2^n$.

A needs to do at most 2^n oracle queries to $H_{2^{t'}+1}$. However, in order to make consistent queries to $H_{2^{t'}+1}$ then A needs to evaluate either of $H_1, \ldots, H_{2^{t'}}$ anew also 2^n times. Then for distinct (domain separated) instantiations of $(H_1, \ldots, H_{2^{t'}+1})$ the success probability of A is upper bound by $q_{\mathrm{RO}}/2^{n+1}$. □

6 Summary of the Security Results

To assess the security of the *LANE* iteration we have used the reduction-based provable security approach. More precisely, we state security claims on the *LANE* iteration under some concrete assumptions on the underlying compression function. Following this approach we are able to show concrete security bounds on the computational complexity of an adversary against the *LANE* iteration. We also exhibit information theoretic results on the *LANE* iteration which indicate security against generic attacks under the assumption that the compression function is an ideal primitive.

Similarly to the known Merkle-Damgård iterative principle [8, 5], we show that both the non-salted and salted versions of the *LANE* iteration are provably collision secure. We follow Rogaway's human-ignorance approach [9] and relate the advantage of an adversary against the *LANE* hash function to that of another adversary against the *LANE* compression function to derive a tight security bound.

The upper bounds on the advantage of information theoretic adversaries against the second preimage and preimage security, respectively, of the non-salted *LANE* hash function indicate that 2^n evaluations of the underlying ideal compression function need to be performed to break the respective security property. Moreover, making a (variant of) preimage security assumption on the output transformation of LANE and adopting some randomness extraction and regularity properties on the iterative portion of the non-salted *LANE* hash function, we exhibit tight preimage security bound on the *LANE* iteration.

For the salted version of the *LANE* hash function we develop a broad set of security notions that capture most of the important attack scenarios of randomized hashing. In addition, we also take into account the possibility of attacks under equal or distinct and known or secret salt values. We obtain the same (as for the non-salted *LANE* hash function) security against information theoretic adversaries in the second preimage case and even stronger (with a factor of 2^n) in the preimage case.

The latter information theoretic results show that no generic attacks on the second preimage and preimage security on both salted and non-salted *LANE* hash function variants succeed with probability one in up to 2^n number of evaluations of an ideal compression function.

Another important security feature of the *LANE* iterative design in both salted and non-salted versions is the security against extension attacks and lack of structural flaws ensured by the prefix-free property of the processed inputs. The latter design characteristic of *LANE* ensures its indifferentiability from a random oracle according to the work of Coron et al. [4] and pseudorandom function behavior according to Bellare et al. [2].

The suggested parallel processing method *pMode* is also shown to be collision secure when the underlying hash functions are collision secure. Under the second preimage/preimage security and some mild assumptions on the min entropy extraction properties of the hash functions we also show that the parallel mode of operation is second preimage/preimage secure.

Acknowledgements

The author thanks Sebastiaan Indesteege, Orr Dunkelman, Sebastian Faust, Markulf Kohlweiss and Bart Preneel for their valuable feedback. The author is supported by a Ph.D. Fellowship from the Flemish Research Foundation (FWO - Vlaanderen) and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and the Interdisciplinary Institute for BroadBand Technology (IBBT), and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir, and S. Zimmer. Second preimage attacks on dithered hash functions. pages 270–288, 2008.
2. M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. pages 514–523, 1996.
3. E. Biham and O. Dunkelman. A framework for iterative hash functions – HAIFA. Second NIST Cryptographic Hash Workshop, 2006.
4. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. pages 430–448, 2005.
5. I. Damgård. A design principle for hash functions. pages 416–427, 1990.
6. S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. pages 41–59, 2006.
7. J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. pages 474–490, 2005.
8. R. C. Merkle. A certified digital signature. pages 218–238, 1990.
9. P. Rogaway. Formalizing human ignorance: Collision-resistant hashing without the keys. In *Vietcrypt 2006*, volume 4341, 2006.
10. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. pages 371–388, 2004.