

Conditions for sound tree hashing modes

Guido BERTONI¹ Joan DAEMEN¹ Michaël PEETERS²
Gilles VAN ASSCHE¹

¹STMicroelectronics

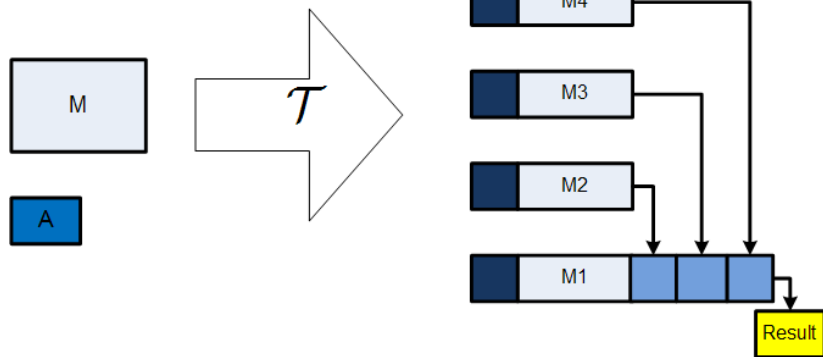
²NXP

13 January 2008 — Dagstuhl Cryptography Seminar

Tree hashing mode

- Can hash parts of a message independently and in parallel
- Calls n -bit compression function \mathcal{F}
- Input:
 - Message (length) $|M|$
 - Values of tree parameters A
- Output: Hashing template T :
 - Array of nodes N_i , containing blocks of types:
 - Parameter block: contents fully determined by $|M|$ and A
 - Message block: sequence of bits mapped from message M
 - Chaining block: $\mathcal{F}(N_i)$ for some other node.
 - T : graph with edges from nodes to chaining blocks
 - T fully determined by A and $|M|$

Tree hashing mode: example

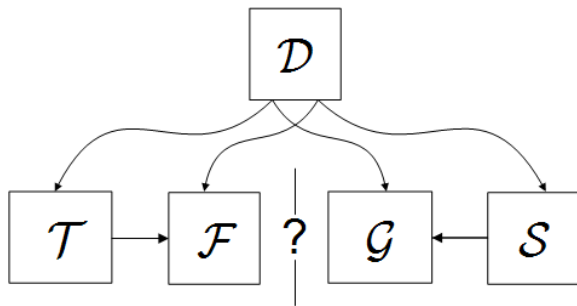


Sound tree hashing mode

- Sound tree hashing mode should
 - introduce no weaknesses
 - behave like a random oracle if \mathcal{F} is a random oracle
 - ... up to some limit
- Indifferentiability framework:
 - Maurer et al., TCC 2004; Coron et al., CRYPTO 2005
 - Provides upper bound for advantage in differentiating \mathcal{T} calling \mathcal{F} from random oracle
 - Upper bound applies to generic attacks.
- What do you win: secure tree hashing if (non-tree) \mathcal{F} is secure

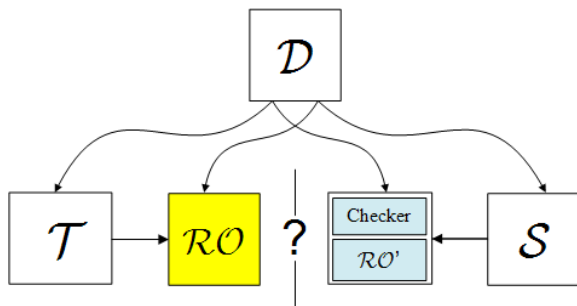
Indifferentiability

- Quantifies effort to distinguish mode of use \mathcal{T} calling ideal compression function \mathcal{F} from another ideal component \mathcal{G} and a simulator \mathcal{S}

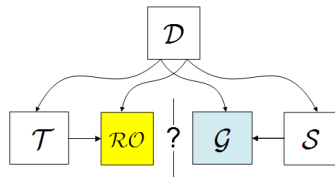


Indifferentiability

- Quantifies effort to distinguish mode of use \mathcal{T} calling ideal compression function \mathcal{F} from another ideal component \mathcal{G} and a simulator \mathcal{S}



The simulator



- We build a simulator \mathcal{S} with the following properties:
 - \mathcal{T} -consistency: output gives no proof for $(\mathcal{S}, \mathcal{G}) \neq (\mathcal{F}, \mathcal{T})$
 - \mathcal{G} -Completeness: \forall query to \mathcal{G} , same response can be obtained by sequence of queries to \mathcal{S} , with similar cost
 - Maximum-randomness: within those constraints, \mathcal{S} responses maximize randomness
- Differentiability advantage determined by difference in output distribution between \mathcal{S} and \mathcal{F} , i.e. a random oracle.

The simulator realizes this by:

- Acting deterministic:
 - Keeping all received queries and the returned responses.
 - When receiving a query again, return the same response
- Avoiding internal collisions:
 - Every response to a new query is different from all previous ones
- Acting \mathcal{T} -consistent
 - When receiving a query, see whether, combined with received queries, it can be turned into a \mathcal{G} -query using the tree scheme rules
 - If so, get the response from \mathcal{G}

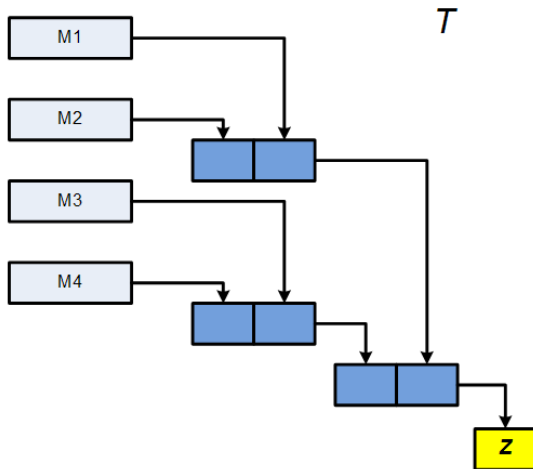
\mathcal{T} -consistency: illustration



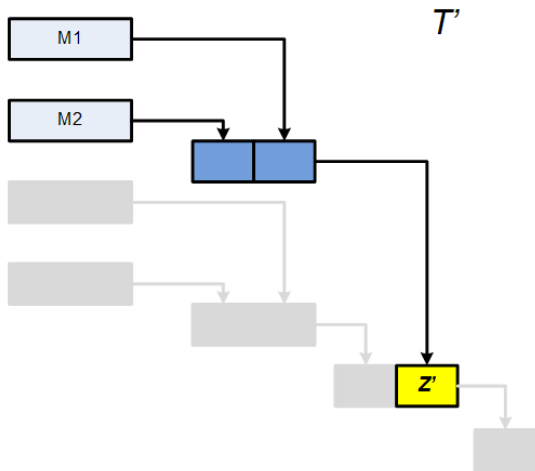
The final-internal-node distinguisher

- Take $(|M|, A) \rightarrow T$ and $(|M'|, A') \rightarrow T'$ where T' is a *subtree* of T
- Then the distinguisher can mount the following attack:
 - Query \mathcal{G} with (M, A) resulting in Z
 - Query \mathcal{G} with (M', A') resulting in Z'
 - Perform queries to \mathcal{S} simulating the query (M, A) to \mathcal{G} , where the calls due to $\text{subtree}(M', A')$ are replaced by chaining value Z' . Let the result be Z''
- \mathcal{S} cannot get response from \mathcal{G} because he cannot reconstruct the \mathcal{G} -query, so can only guess $Z'' (\neq Z)$.
- In the case of \mathcal{T} and \mathcal{F} we have $Z'' = Z$.

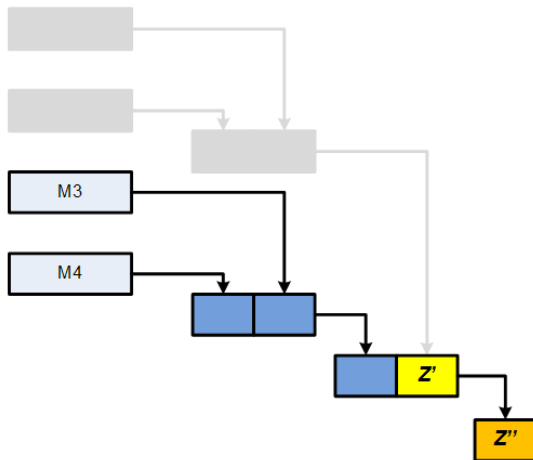
The final-internal-node distinguisher: illustration



The final-internal-node distinguisher: illustration



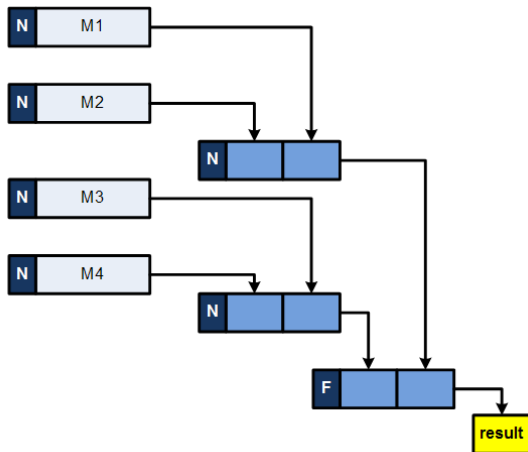
The final-internal-node distinguisher: illustration



Avoiding the final-internal-node distinguisher

- It shall be impossible to find $(|M|, A) \rightarrow T$ and $(|M'|, A') \rightarrow T'$ with T' a subtree of T
- Several approaches:
 - Use domain separation between final and other nodes
 - Limit the topology of the trees, e.g. fix the number of levels
 - Do some specific processing with the final node
 - ...
- Domain separation
 - results in simplest proofs
 - costs only 1 bit of input

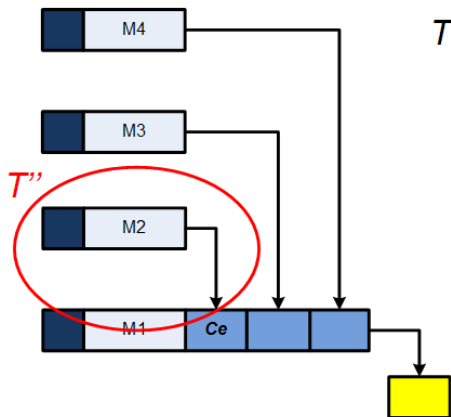
Final node domain separation



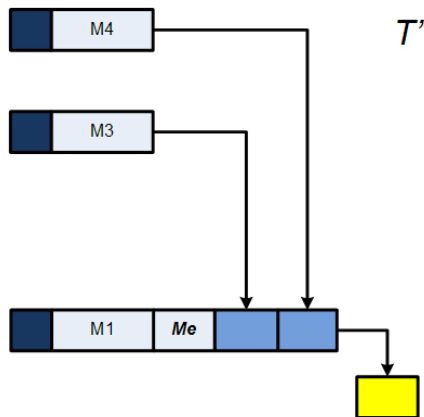
The chaining-message-block distinguisher

- Take $(|M|, A) \rightarrow T$ and $(|M'|, A') \rightarrow T'$ with T having a chaining value C_e with subtree T_e where T' has an n -bit message block M_e .
- Then the distinguisher can mount the following attack:
 - Query \mathcal{S} for the chaining value corresponding with T'' , resulting in Z''
 - Query \mathcal{G} with (M, A) resulting in Z
 - Query \mathcal{G} with (M', A') with Z'' in location M_i , resulting in Z'
- Queries (M, A) and (M', A') to \mathcal{G} are different and hence probably $Z' \neq Z$
- Sending these queries to \mathcal{T} and \mathcal{F} results in $Z'' = Z$.

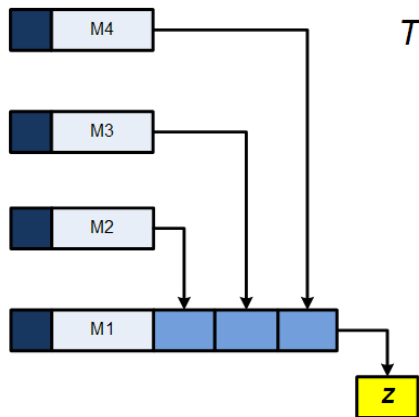
The chaining-message-block distinguisher: illustration



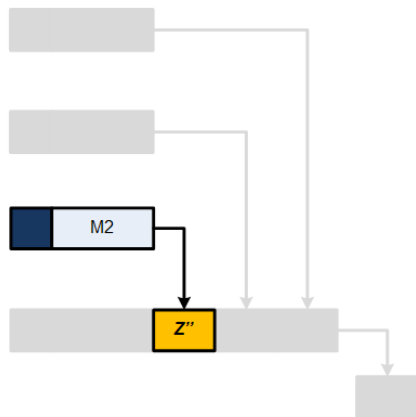
The chaining-message-block distinguisher: illustration



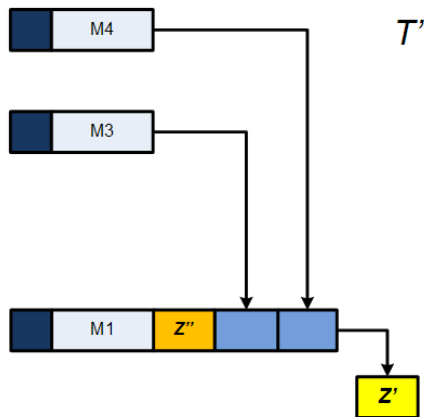
The chaining-message-block distinguisher: illustration



The chaining-message-block distinguisher: illustration



The chaining-message-block distinguisher: illustration



Avoiding the chaining-message-block distinguisher

- It shall be impossible to have $(|M|, A) \rightarrow T$ and $(|M'|, A') \rightarrow T'$ with T having a chaining value C_e with subtree T_e where T' has an n -bit message block M_e .
- Several approaches to make decoding unique
- Local coding: remove ambiguity in all nodes
 - Use Tag-Length-Value coding
 - Fix order and put information in parameter block
 - Have input nodes and nodes with chaining values and apply domain separation
- Final-node coding: remove ambiguity in parameter block of final node
 - Shall determine message-template bit mapping and not allow chaining-message block exchange

Indifferentiability proof sketch

- Proof by induction: additional query to \mathcal{S} preserves \mathcal{T} -consistency
- Implies two conditions on simulator \mathcal{S}
 - To avoid internal collisions: non-final node query excludes response for future use
 - To avoid responses to become \mathcal{T} -inconsistent in future:
 - Any query may require exclusion of potential chaining blocks
 - When using final-node coding, at most one per *interpretation*
 - \mathcal{T} can be designed such that at most 1 chaining block is excluded
- A query to \mathcal{S} excludes at most two response values for future use
- Advantage in distinguishing \mathcal{S} from \mathcal{F} : $\leq q^2/2^n$ with
 - q : number of queries to \mathcal{S}
 - n : length of the chaining blocks

Optimum value of n ?

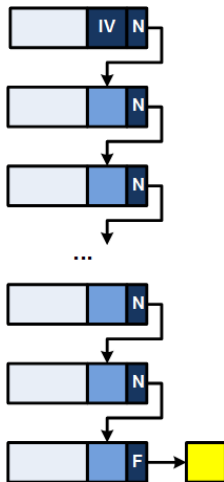
- Assume \mathcal{F} is a hash construction that is indifferentiable from a random oracle with advantage $N^2/2^{c+1}$
 - N : number of calls to the underlying primitive
 - c : *capacity* quantifying strength of construction
- Worst-case total advantage: $N^2/2^{c+1} + q^2/2^n$
 - Increasing n reduces the advantage
 - Increasing n reduces efficiency
 - Gain is small for $n > c$
- Optimum choice: $n \approx c$. But:
 - Requires F to have output with length up to c
 - This is often not the case (e.g. wide-pipe)
 - Solutions:
 - Use function \mathcal{F} with infinite-length output
 - Apply output extension tricks

Example: KECCAKTREE

- Tree parameters A
 - Tree growing mode: *leaf interleaving* or *final node growing*
 - Leaf block size B
 - Height of the tree H
 - Degree of the nodes D
- Mapping from $(|M|, A)$ to T
 - Domain separation between final node and other nodes
 - All parameters of A coded in final node
 - Case of *leaf interleaving*
 - Balanced tree of height H and degree D
 - Blocks of B bits spread on the D^H leaves cyclically
 - Case of *final node growing*
 - Balanced tree of height H and degree D except final node
 - One blocks of B bits per leaf

Example: conventional iterated hashing

- We can define a mode \mathcal{T} :
 - with A empty,
 - mapping to a set of nodes with fixed length
 - with each node having a single chaining block and a single message block.
 - First node has a parameter block containing fixed IV instead of the chaining block.
- This is simply:
 - Merkle-Damgård
 - with prefix-free coding
 - calling a fixed-input length compression function \mathcal{F} .



Questions?

Thanks for your attention!

Any questions?