Starfish

Version 1.2

Final

Source	Marlin Developer Community
Date	August 22, 2006

Notice

Intellectual Property

A commercial implementation of this specification requires a license from the Marlin Trust Management Organization.

Contact Information

Feedback on this specification should be addressed to: editor@marlin-community.com

Contact information for the Marlin Trust Management Organization can be found at: http://www.marlin-trust.com/

Contents

1 Introduction

1.1 Goal and Scope

This specification describes Starfish, which is the Marlin broadcast encryption scheme based on HBES (Hierarchical Hash-Chain Broadcast Encryption Scheme). This specification documents HBES and the usage of a key tree structure and a Broadcast Key Block structure to provide a secret Broadcast Key to all Leaf Nodes (e.g., representing devices) in the tree except for ones that are excluded (for example, due to a security compromise).

1.2 Document Organization

This specification is organized as follows:

Introduction
Introductory information, including lists of namespaces, abbreviations, definitions, and references.

Goals
A brief description of the Starfish goals.

HBES Architecture
An overview of the HBES architecture, including the HBES tree structure, the sets of keys assigned in advance to Nodes in the tree, and the Broadcast Key Block that specifies which Nodes are excluded and supplies a Broadcast Key to Nodes that are not excluded.

BKB Encoding
A description of the two mechanisms for encoding and formatting a Broadcast Key Block.

1.3 Conformance Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

These capitalized key words are used to unambiguously specify requirements and behavior that affect the interoperability and security of implementations. When these key words are not capitalized they are meant in their natural-language sense.

All elements of this specification are considered Normative unless specifically marked Informative. All Normative Elements are Mandatory to implement, except where such an element is specifically marked OPTIONAL. Finally, where Normative elements are described as OPTIONAL, they MAY be omitted from an implementation, but when implemented, they MUST be implemented as described.

1.4 Namespaces and Identifiers

This specification defines a schema conforming to XML Schemas [Schema] and normative text to describe the syntax and semantics of XML-encoded objects and protocol messages. In cases of disagreement between the schema document and the schema listing in this specification, the schema document takes precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema document.

1.4.1 Namespaces and Notation

The following table summarizes the normative schema defined by this specification and its XML namespace [XMLns] URI. This URI MUST be used by implementations of this specification:

Prefix	XML Namespace	Schema File Name	Description
sf:	http://marlin-drm.com/starfish/1.2	Starfish.xsd	Starfish schema

In addition to the schema defined by this specification, we leverage existing schemas to achieve our design goals. The following table summarizes the external schemas used in this specification:

Prefix	XML Namespace	Reference
ds:	http://www.w3.org/2000/09/xmldsig#	[xmldsig]
xs:	http://www.w3.org/2001/XMLSchema	[Schema]

1.5 Abbreviations

bK	Broadcast Key
BKB	Broadcast Key Block
HBES	Hierarchical Hash-Chain Broadcast Encryption Scheme
HKT	HBES Key Tree
HNK	HBES Node Key Set
KEK	Key Encryption Key
npid	Node Path ID
devid	Device ID

1.6 Terms and Definitions

Ancestor Node	A Node on the path from a given Node up to the root Node, not including the given Node itself.
Broadcast Key	A key made available by a Broadcast Key Block to Non-Excluded Nodes.
Broadcast Key Block	A data structure used to exclude some Nodes and provide a Broadcast Key to Non-Excluded Nodes.
Completely Excluded Node	An Excluded Node with no Non-Excluded descendants.
d	The depth (number of levels) of an HBES Key Tree.

Descendant Node	A Node that is on one of the (multiple) paths from a given Node down to the Leaf Nodes, not including the given Node itself.
Device ID	A bit pattern that uniquely identifies a Leaf Node by specifying the path from the HBES Key Tree root to the Leaf Node.
Device Key Set	A set that includes the HBES Node Key set of a Leaf Node and the HBES Node Key sets of its Ancestor Nodes.
E	The number of Representative Excluded Leaf Nodes in an HBES Key Tree .
Excluded Node	A Node that is Excluded. An ancestor of an Excluded Node is also an Excluded Node.
Group	The t immediate Descendant Nodes of a given Node, or the t Nodes of Layer 0 of an HKT.
HBES Key Tree	A key management structure for HBES.
HBES Node Key Set	A unique set of keys assigned to a Node.
Interval	A set of cyclically consecutive Non-Excluded Nodes in a Group.
Key Encryption Key	A key associated with an Interval and used in a Broadcast Key Block to encrypt a Broadcast Key.
KSIZE	The byte size of the Broadcast Key.
Leaf Node	A Node (of an HBES Key Tree) that has no Descendant Nodes. Used in Starfish to represent a device.
Maximal Completely Excluded Node	A Completely Excluded Node whose parent is not a Completely Excluded Node.
Node	An object in an HBES Key Tree hierarchy that may be excluded.
Node Path ID	The ordinal number of a Node within its Group.
Non-Excluded Node	A Node that is not excluded.
Representative Excluded Leaf Node	A Leaf Node for which either (1) it is a Maximal Completely Excluded Node, or (2) it is the leftmost descendant Leaf Node of a Maximal Completely Excluded Node.
t	The number of Nodes in each Group in a tree. Thus, the term "t-ary tree" means that each Group has t Nodes.

1.7 Starfish Values

d	16
KSIZE	16
t	16

1.8 References

[AES]	NIST FIPS 197: Advanced Encryption Standard (AES), November 2001, http://csrc.nist.gov /publications/fips/fips197/fips-197.pdf
[CS]	D.Naor, M.Naor and J.Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", in Advances in Cryptology – CRYPTO 2001, LNCS 2139, 2001, 41-62

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.
[RFC4051]	D. Eastlake 3rd. *Additional XML Security Uniform Resource Identifiers (URIs)*. IETF RFC4051. April 2005. http://www.ietf.org/rfc/rfc4051.txt
[RSA]	PKCS #1 : RSA Encrytion Standard (RSA) Version 1.5 http://www.rsasecurity.com/rsalabs/node.asp?id=2125
[Schema]	XML Schema Part 1: Structures. W3C Recommendation. D. Beech, M. Maloney, N. Mendelsohn, H. Thompson. May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ XML Schema Part 2: Datatypes. W3C Recommendation. P. Biron, A. Malhotra. May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
[SHA-1]	NIST FIPS 180-1: Secure Hash Standard (SHA-1), Apr. 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm
[xmldsig]	*XML-Signature Syntax and Processing W3C Recommendation* http://www.w3.org/TR/xmldsig-core/
[xml-exc-c14n]	*Exclusive XML Canonicalization, Version 1.0, W3C Recommendation*. *18 July 2002* http://www.w3.org/TR/xml-exc-c14n/
[XMLns]	Namespaces in XML. W3C Recommendation. T. Bray, D. Hollander, A. Layman. January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114

2 Goals

The Starfish goals are the following:

- Device Exclusion (Note: “Exclusion” is the proper term, rather than “revocation,” as revocation implies disablement of functionality. In general, services will only exclude devices from access to new content.)
  - Exclusion of specific device ID(s)
  - Exclusion of a family of devices
- The amount of permanent device storage needed to store exclusion-related information should be minimal.
- The size of the transmitted exclusion-related information should be minimal.

3 HBES Architecture

3.1 Overview

HBES is a symmetric key mechanism that makes a Broadcast Key (bK) cryptographically available to a large set of Nodes (the Non-Excluded Nodes) in a tree structure, while preventing access by a smaller set of Excluded Nodes. A Broadcast Key Block (BKB) is derived from the Broadcast Key and the set of Excluded Nodes. A Non-Excluded Node may recover the Broadcast Key from a BKB and, for example, use it to decrypt a media Content Key.

One goal of HBES is to minimize the length of a BKB, the computing time to recover a bK, and the size of the Device Key Set for a Leaf Node. The Device Key Set for a Leaf Node includes the Node Keys for that Node and the Node Keys of all its Ancestor Nodes. All keys are assigned in advance.

With most broadcast encryption schemes, if for a fixed number of exclusions the transmission overhead decreases linearly, then the storage increases exponentially. With HBES, however, the storage size (the size of the Device Key Set) increases linearly because of its usage of a hash chain. Thus, HBES has lower transmission overhead than other broadcast encryption schemes for similar storage costs.

3.2 Initial Configuration

3.2.1 Tree structure

The HBES Key Tree (HKT) is a t-ary tree. In Starfish, t is equal to 16. The HKT is used to manage Node exclusion. Each Leaf Node has a unique Device Key Set consisting of its t keys (its HBES Node Key Set) and the HBES Node Key Sets of all its Ancestor Nodes. The HKT provides the basis for calculating the Device Key Set for each Leaf Node and for creating a BKB structure that can be used to exclude one or more Nodes.

The HKT has a layered structure. In Starfish, the depth d (the number of layers) is 16. The first layer descending from the root contains the highest t Nodes and is called Layer 0. The immediate layer below Layer 0 consists of t^2 Nodes and is called Layer 1. The Nodes at the bottom of the HKT, those at Layer d-1, are the Leaf Nodes.

All Nodes that are on the path from a Node up to the root Node (excluding the Node itself) are called Ancestor Nodes of that Node. All Nodes that are on a path from a Node down to and including a Leaf Node (but excluding the Node itself) are called Descendant Nodes of that Node. Formally, a Leaf Node is a Node with no Descendant Nodes. HBES defines a Group to be the t immediate Descendant Nodes of a given Node. In other words, a Group consists of a set of t Nodes sharing the same parent (immediate Ancestor Node). The children of a Node are its immediate Descendant Nodes.

3.2.2 Node Path IDs and Device IDs

Each Node in a Group is identified by a unique Node Path ID, denoted by **npid**. The value of an **npid** is in the range 0 to t-1. A Node Path ID is assigned sequentially from the leftmost Node in the Group to the rightmost Node in the Group. Thus, the leftmost

Node is Node 0, the Node to its right is Node 1, and the rightmost Node is Node t-1. The size of the Node Path ID is $\log_2 t$ bits.

Every Leaf Node has a unique Device ID, denoted by **devid**. The value of a Leaf Node's **devid** is the concatenation of all Ancestor Node **npid**s with the Leaf Node's **npid**. That is, the leftmost $\log_2 t$ bits of a **devid** are the **npid** of the Leaf Node's Ancestor Node in Layer 0. The next $\log_2 t$ bits are the **npid** of the Leaf Node's ancestor node in Layer 1, and so on. The last $\log_2 t$ bits of the **devid** are the **npid** of the Leaf Node itself.

See Figure 2 in Appendix B for an example of a 4-ary HKT (t=4). In a 4-ary HKT, each npid is 2 bits ($\log_2 4$) long, and there are 4 layers in the example , so a devid is 8 bits long. In Starfish, a Device ID is 64 bits (= 4 bits per npid * 16 layers).

3.3 Key Generation and Pre-distribution

HBES functions as follows: Each Node in a t-ary HBES Key Tree is assigned in advance t keys, referred to as the HBES Node Key Set (HNK) for that Node. A Leaf Node Key Set contains its HNK and all the HNKs of its Ancestor Nodes. Thus, in Starfish, a device (represented by a Leaf Node) is assigned 256 keys (16 layers * 16 keys per layer).

There is a relationship, described below, between the keys assigned to a Node and those assigned to the other Nodes in its Group. The characteristics of this relationship enable efficient exclusion of Nodes. To provide a Broadcast Key to Non-Excluded Nodes only, a Broadcast Key Block containing one or more encryptions of the Broadcast Key (depending on how many Nodes are excluded) is sent to all Nodes. For each Broadcast Key encryption, the encryption is done using one of the keys in a Non-Excluded Node such that Excluded Nodes do not have and cannot calculate the key used for encryption, whereas Non-Excluded Nodes can.

3.3.1 Key Assignment

Each Node in a Group is assigned a seed value as one of its keys. The seed value for Node i in the Group is denoted by S_i for $0 \le i \le (t-1)$. All seed values should be randomly generated and pairwise independent.

The keys of a given Node (its HBES Node Key Set) consist of the Node's seed and t-1 keys that are the result of applying the HBES SHA-1 hash function (see Appendix A), denoted by h, one or more times to each of the seed values of the other Nodes in its Group. For each Node i, $h(S_i)$, that is, the result of performing the hash once on S_i, is assigned to the Node to the immediate "right" of Node i. In general, the Node to the right of Node i is Node i+1, except that the Nodes of a Group are viewed cyclically such that Node 0 is considered to be to the right of Node t-1. Thus, in an 8-ary tree, Node 0 is to the right of Node 7, and $h(S_7)$ is assigned to Node 0.

The result of performing the hash function twice, starting with the value S_i, is referred to as $h^2(S_i)$. This value is assigned as a key for the Node two Nodes to the right of Node i. The remaining keys are assigned in a similar fashion, with $h^3(S_i)$ assigned to the Node 3 Nodes to the right of Node i, and so on until $h^{t-1}(S_i)$ is assigned to the Node t-1 Nodes to the right (i.e., one Node to the left).

As a result, the HBES Node Key Set (HNK) of Node i, for $0 \le i \le (t-1)$, is a sequence of values $k_{i,n}$ for $0 \le n \le (t-1)$, where

$$k_{i,n} = h^{((i+t-n) \bmod t)}(S_n)$$

For example, for t=8, the HNK of Node i=0 is

$$k_{0,0}, k_{0,1}, k_{0,2}, k_{0,3}, k_{0,4}, k_{0,5}, k_{0,6}, k_{0,7}$$

In this case, i+t = 8 , so the HNK of Node 0 is

$$h^{((8-0) \bmod 8)}(S_0), h^{((8-1) \bmod 8)}(S_1), h^{((8-2) \bmod 8)}(S_2), h^{((8-3) \bmod 8)}(S_3),$$
$$h^{((8-4) \bmod 8)}(S_4), h^{((8-5) \bmod 8)}(S_5), h^{((8-6) \bmod 8)}(S_6), h^{((8-7) \bmod 8)}(S_7)$$

which is

$$S_0, h^7(S_1), h^6(S_2), h^5(S_3), h^4(S_4), h^3(S_5), h^2(S_6), h(S_7)$$

Similarly, the HNK of Node i=4 is

$$k_{4,0}, k_{4,1}, k_{4,2}, k_{4,3}, k_{4,4}, k_{4,5}, k_{4,6}, k_{4,7}$$

which, since i+t= 12, is

$$h^{((12-0) \bmod 8)}(S_0), h^{((12-1) \bmod 8)}(S_1), h^{((12-2) \bmod 8)}(S_2), h^{((12-3) \bmod 8)}(S_3),$$
$$h^{((12-4) \bmod 8)}(S_4), h^{((12-5) \bmod 8)}(S_5), h^{((12-6) \bmod 8)}(S_6), h^{((12-7) \bmod 8)}(S_7)$$

which is

$$h^4S_0, h^3(S_1), h^2(S_2), h^1(S_3), S_4, h^7(S_5), h^6(S_6), h^5(S_7)$$

Figure 3 in Appendix C shows an example of the HNK for each of the Nodes in a Group with eight Nodes.

3.3.2 Device Key Set

Each Leaf Node stores a unique Device Key Set, which consists of the HNK of the Leaf Node and the HNKs of all its Ancestor Nodes. For a 16-ary tree and a key size of 16 bytes, the Device Key Set size is 4K bytes long:

16 layers * 16 node keys * 16 bytes per key

The order of keys in a Device Key Set is the HNK of the Node’s ancestor in Layer 0, followed by the HNK of the Node’s ancestor in Layer 1, and so on. The ordering of the keys in the HNK for each Node i is as shown below:

$$k_{i,0}, k_{i,1}, k_{i,2}, k_{i,3}, \ldots , k_{i,(t-2)} , k_{i,(t-1)}$$

where the values $k_{i,n}$ for $0 \le n \le (t-1)$ are as defined in §3.3.1. The order of the Device Key Set MUST be preserved so that the receiver of a Broadcast Key Block can recover the Broadcast Key, based on the node exclusion information discussed in the next section.

3.4 Node Exclusion

Once a Broadcast Key, bK, is selected, bK is encrypted with Key Encryption Keys (KEKs). A KEK is a key used to encrypt bK such that only Non-Excluded Nodes have or can calculate the KEK and thereby decrypt the bK. Which keys are used, and how many are used, depend on which Nodes are excluded and how many Nodes are excluded.

An Interval is a set of cyclically consecutive Non-Excluded Nodes in a Group containing one or more Excluded Nodes. "Cyclically consecutive" means that Node 0 is considered to be to the right of Node t-1. Thus, if a Group contains one or more Excluded Nodes, and both Node 0 and Node t-1 are Non-Excluded, they are in the same Interval.

The start Node of an Interval is the unique Node in the Interval whose immediate left neighbor is an Excluded Node (and thus not in the Interval). In other words, the start Node of an Interval is the Node to the immediate right of the rightmost Excluded Node preceding the Interval, keeping in mind the fact that Node 0 is considered to be to the right of Node t-1.

A KEK is determined for each Interval in each Group containing at least one Excluded Node and at most t-1 Excluded Nodes. The KEK for an Interval is the result of hashing the start Node's seed value one time less than the number of Nodes in the Interval. The relationships between the keys stored by the Nodes in a Group (see Appendix C) are such that all Non-Excluded Nodes in the Interval either have the KEK or they have another key that can be hashed one or more times to calculate the KEK, but the Excluded Nodes do not have the KEK and cannot calculate it. See Figure 4 in Appendix D and its accompanying description for an example showing the determination of KEKs for a Group containing two Intervals.

A Broadcast Key Block (BKB) contains the encryptions of the bK, one per KEK. The encryptions using the KEKs for Intervals in Groups in Layer 0 (if any) are stored first, followed by the encryptions using the KEKs for the Intervals in Groups in Layer 1, and so on. For a given Layer, the encryptions using the KEKs for the leftmost Group containing one or more Intervals are stored first, followed by the encryptions using the KEKs for the second-leftmost Group containing Intervals, and so on. For a given Group, the encryption using the KEK for the first Interval is stored first, followed by the encryption using the KEK for the second Interval, and so on. (The first Interval for a Group is the Interval with the lowest-numbered start Node. The second Interval is the one with the second lowest-numbered start Node, and so on.)

A BKB also contains a list of Excluded Node IDs. The Excluded Node IDs are an encoding identifying which Nodes are excluded at each Layer. They can be used to determine the Intervals in each Group that contains one or more (up to t-1) Excluded Nodes, and this information is sufficient to know which KEKs were used to encrypt the bK. See Appendix G for example pseudocode for determining a KEK that a Non-Excluded Node either has or can calculate, and then using that KEK to decrypt one of the encryptions and thereby extract the bK.

The Excluded Node IDs are constructed as described in the following. You may want to consult the example in Appendix D.2 as you read the description. First, a few definitions are in order: An Excluded Node is considered a *Completely Excluded Node* if either (1) its descendants are all Excluded Nodes, or (2) it is a Leaf Node (which has no descendants). A *Maximal Completely Excluded Node* is a Completely Excluded Node whose parent is not a Completely Excluded Node. A Leaf Node is a *Representative Excluded Leaf Node* if either (1) it is a Maximal Completely Excluded Node, or (2) it is the leftmost descendant Leaf Node of a Maximal Completely Excluded Node.

The first step in creating the Excluded Node IDs is to count the number of Representative Excluded Leaf Nodes, **E**. This is the same as the number of Maximal Completely Excluded Nodes.

The list of Excluded Node IDs contains **E** entries per Layer. The **E** entries for Layer 0 appear first, followed by the **E** entries for Layer 1, and so on through Layer d-1 (i.e., Layer 15 in Starfish). The first entry for a Layer provides information regarding the Node in that Layer that is the Ancestor Node of the first Representative Excluded Leaf Node (device), viewing the Leaf Nodes from left to right. The second entry for a Layer provides information regarding the Node in that Layer that is the Ancestor Node of the second Representative Excluded Leaf Node, and so on. (Note: The Ancestor Nodes of an Excluded Leaf Node are also excluded.)

Each entry is a 2-tuple (**gid**, **npid**), where both **gid** and **npid** are of size $\log_2 t$ bits (4 bits for the 16-ary Starfish tree, but only 2 bits for the 4-ary tree in the example).

The **npid** portion of each entry is the Node Path ID **npid** (see §3.2.2) of the specified Node in its Group.

Each **gid** is one of three possible values: 0, 1, and -1 ($11\ldots1_2$). The **gid**s are determined as follows:

1. The **gid** for the first entry for a Layer is $00\ldots0_2$, unless the Node currently being processed is a descendant of a Maximal Completely Excluded Node, in which case the **gid** is $11\ldots1_2$.
2. The **gid**s for subsequent entries for a Layer are the following:
    1) The **gid** is is $11\ldots1_2$ if the Node currently being processed is a descendant of a Maximal Completely Excluded Node. Otherwise:
    2) The **gid** is the same as the previous entry's **gid** if the previous entry's Node and the current Node are in the same Group.
    3) If they are in different Groups, the **gid** is the previous entry's **gid** + 1 (mod 2).

4 BKB Encoding

This specification defines two mechanisms for encoding and encapsulating the information which comprises the BKB:

- A binary representation, defined in §4.2.
- An XML representation, defined in §4.3, which builds upon elements of the binary representation.

4.1 BKB Fields

A BKB consists of nine fields, defined as follows:

BKB Length	The entire length in bytes of the BKB (including the length of this field), when the BKB is in its binary format.
Structure Version	The number of times the BKB format has been revised. Each time the BKB format is changed, the Structure Version is incremented by one. The initial Structure Version number is 0.
Revocation Version	A revocation version number. BKBs are issued as an ordered series. Each time one or more additional Nodes are excluded, the Revocation Version of the BKB is incremented by one. The initial Revocation Version number is 0.
Key Check Data	A hashed value[1] of the Broadcast Key. A processor MAY verify the correctness of the Broadcast Key obtained from the BKB by executing the following steps: Step 1: Compute the Broadcast Key by decrypting one of the encryptions using a KEK. Step 2: Hash the Broadcast Key. Step 3: Compare the Key Check Data value and the hashed value. Step 4: If the two values are equal, the computed Broadcast Key is correct. Step 5: Otherwise, the processor may retry Step 1 or stop this process. The size of this field is equal to the size of a Broadcast Key. The byte size of the Broadcast Key is denoted by **KSIZE**.
Reserved	Free space set aside for future use. The value of this field for this version of the BKB structure must be set to 0x0 for all bytes.
Number of Representative Excluded Leaf Nodes	The number of Representative Excluded Leaf Nodes, as described in §3.4. This value is denoted by **E** in the following, for describing the components of the BKB that may vary relative to the value of **E**.

[1] This specification uses HBES SHA-1 (defined in Appendix A) as the hash function.

Excluded Node IDs — A sequence of the Excluded Node IDs, as described in §3.4. The size of this field is 128**E** bits (= 16**E** bytes).

Signature — A signature covering all the fields preceding this field, specifically, BKB Length, Structure Version, Revocation Version, Key Check Data, Reserved, Number of Representative Excluded Leaf Nodes, and the sequence of Excluded Node IDs.

Encrypted Broadcast Keys — A sequence of Broadcast Keys, each encrypted by a KEK that is determined by an Interval, as described in §3.4. The size of each encrypted Broadcast Key is equal to **KSIZE**. The total size of this field is dependent on **E** (the number of Representative Excluded Leaf Nodes).

4.2 Binary BKB Encoding

A BKB contains information that enables all Non-Excluded Nodes to compute the Broadcast Key. Table 4-1 defines the binary format of the BKB. Refer to Appendix E for a detailed example of this format.

4.2.1 Bit/Byte ordering

All data variables in this specification are presented with the most significant bit (or byte) on the left-hand side and the least significant bit (or byte) on the right-hand side. Where a variable is broken down into a number of substrings, the leftmost (most significant) substring is numbered 0, the next most significant is numbered 1, and so on through to the least significant.

4.2.2 Binary BKB Format

Table 4-1 shows the binary BKB format. The size of the BKB depends on the number of Representative Excluded Leaf Nodes.

Bytes	Size	Field Name	Collection
0 ~ 3	4 bytes	BKB Length	Revocation Information Fields
4 ~ 7	4 bytes	Structure Version	
8 ~ 11	4 bytes	Revocation Version	
12 ~ 27	16 bytes	Key Check Data	
28 ~ 35	8 bytes	Reserved	
36 ~ 39	4 bytes	Number of Representative Excluded Leaf Nodes (= **E**)	
40 ~ 16**E** + 39	variable	Excluded Node IDs in Layer 0 (8**E** bits) ||[2] … Excluded Node IDs in Layer 15 (8**E** bits)	
16**E**+40 ~	128	Signature[3]	

[2] The "||" symbol is meant to represent concatenation of octet sequences.
[3] The signature covers the fields in the gray area, i.e., BKB Length ~ Excluded Node IDs.

16**E**+167	bytes		
16**E**+168 ~ the value of BKB Length minus one	variable	bK encrypted using the KEK generated from the first Interval in Layer 0 || … bK encrypted using the KEK generated from the last Interval in Layer 0 || … bK encrypted using the KEK generated from the first Interval in Layer 15 || … bK encrypted using the KEK generated from the last Interval in Layer 15	Encrypted Broadcast Keys

Table 4-1 Binary BKB Format

The signature MUST follow the guidance given in PKCS #1 version 1.5 [RSA], with the exception of the digest algorithm, which must be SHA-1 (i.e., SHA-1 with RSA). The binary representation of the BKB limits the signature field to 128 bytes. Thus the modulus MUST be 1024 bits (i.e., 128 bytes). The binary BKB format only represents the signature value; neither the algorithm nor the signer information is conveyed by that encoding. This specification defines the algorithm information (in this paragraph), but the signer information is out of scope for this specification.

The algorithm used to encrypt the Broadcast Key in the Encrypted Broadcast Keys collection must be AES [AES]. The key size is fixed at 128 bits, which equates to a **KSIZE** of 16 bytes. Each Broadcast Key encryption is the result of a single AES 128-bit block cipher operation. (This is effectively ECB mode.) The output of each encryption of the Broadcast Key consumes 16 bytes.

4.3 XML BKB Encoding

The XML schema for the XML BKB encoding defined in this section is depicted in Appendix H. This encoding offers greater flexibility than the binary encoding in describing the security properties of a signed BKB. Specifically, the XML encoding can express more information regarding the signing algorithm, the key used to sign the revocation information, and the authority that issued the public key certificate. This flexibility is a consequence of using [xmldsig] to represent a digital signature.

The following sections describe the XML encoding semantics and the relationships between the XML elements and the fields of a BKB.

4.3.1 <sf:BroadcastKeyBlock> Element

The <sf:BroadcastKeyBlock> element encapsulates the Broadcast Key Block information.

4.3.1.1 keyTreeName Attribute

The < sf:BroadcastKeyBlock> MUST identify the key tree name by specifying a URI for this attribute. The assignment of this identifier is out of scope for this specification, as it is an operational issue. However, it is RECOMMENDED that the identifier for the name of the HBES Key Tree use the following syntax:

urn:marlin:starfish:keytree:[Key Tree Number]

where the [Key Tree Number] is a monotonically increasing positive integer.

4.3.1.2 <sf:RevocationInformation> Element

The < sf:RevocationInformation> element bears a Base64 encoded instance of the binary representation of the Revocation Information Fields collection defined in Table 4-1. The < sf:RevocationInformation> element also defines attributes that aid in processing the encapsulated Revocation Information Fields and that support a mechanism to identify the location where an updated BKB may be acquired.

4.3.1.2.1 structureVersion Attribute

The structureVersion attribute represents the same information as the Structure Version field defined in §4.2.2. The same information is Included in the Base64 encoded instance of the Revocation Information Fields collection. However, its presence in the structureVersion attribute is intended to give the processor a hint so that the processor can determine in advance whether the encapsulated revocation information is new.

4.3.1.2.2 revocationVersion Attribute

The revocationVersion attribute represents the same information as the Revocation Version field defined in §4.2.2. The same information is Included in the Base64 encoded instance of the Revocation Information Fields collection. However, its presence in the revocationVersion attribute is intended to give the processor a hint so that the processor can determine in advance whether the encapsulated revocation information is new.

4.3.1.2.3 distributionURIs Attribute

The distributionURIs attribute provides a list of URIs that can be resolved to obtain the newest BKB. That is, each URI is a pointer to the current BKB. All implementations MUST be prepared to resolve the URI using the HTTP GET method.

4.3.1.2.4 issuedOn Attribute

The issuedOn attribute provides the time at which the BKB was issued. Note that a new BKB may be issued independent of the revocation version. For example, this may occur so as to minimize the lifetime of a given broadcast key. Thus this attribute can be used to determine the currency of a BKB relative to another BKB.

4.3.1.2.5 nextUpdate Attribute

The nextUpdate attribute indicates the time at which an updated BKB will be published. This value MUST be later relative to the issuedOn attribute. A party which relies upon the BKB to exclude access to a given broadcast key can rely upon this attribute as an indication of when to resolve the distributionURI.

4.3.1.3 <sf:EncryptedBroadcastKeys> Element

The < sf:EncryptedBroadcastKeys> element bears a Base64 encoded instance of the binary representation of the Encrypted Broadcast Keys collection defined in Table 4-1.

4.3.1.4 <ds:Signature> Element

The signature MUST be detached and the <ds:Signature> element SHALL be present in the <sf:BroadcastKeyBlock> element that contains the XML representation of the signed <sf:RevocationInformation> element.

The <ds:Signature> block MUST contain:

- A <ds:SignedInfo> element
- A <ds:SignatureValue> element
- A <ds:KeyInfo> element

4.3.1.4.1 <ds:SignedInfo>

The <ds:SignedInfo> MUST embed the following elements:

4.3.1.4.1.1 <ds:CanonicalizationMethod>

The <ds:CanonicalizationMethod> element is empty and its ds:Algorithm attribute MUST have the following value:

http://www.w3.org/2001/10/xml-exc-c14n#

4.3.1.4.1.2 <ds:SignatureMethod>

The <ds:SignatureMethod> element is empty and its ds:Algorithm attribute SHALL have one of the following values:

http://www.w3.org/2000/09/xmldsig#rsa-sha1
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

as specified in [xmldsig] and [RFC4051], respectively.

4.3.1.4.1.3 <ds:Reference>

There MUST only be one <ds:Reference> element inside the <ds:SignedInfo> block.
The value of the ds:URI attribute of the <ds:Reference> element MUST be the ID attribute of the < sf:RevocationInformation> element.
The <ds:DigestMethod> element is empty and its ds:Algorithm attribute MUST have one of the following values:

http://www.w3.org/2000/09/xmldsig#sha1
http://www.w3.org/2001/04/xmlenc#sha256

as specified in [xmldsig] and [xmlenc], respectively.

The <ds:DigestValue> element MUST contain the Base64 encoded value of the digest.

4.3.1.4.2 <ds:SignatureValue>

The signature value MUST be the Base64 encoded value of the signature of the canonicalized ([xml-exc-c14n]) <ds:SignedInfo> element with the key described in the <ds:KeyInfo> element.

4.3.1.4.3 <ds:KeyInfo>

The public key used to verify the signature MUST be carried in an X.509 v3 certificate, and MUST be accompanied by other certificates necessary to complete the certificate path to a trust anchor.

These certificates MUST be carried, encoded in Base64, in <ds:X509Certificate> elements. These <ds:X509Certificate> elements are embedded in a <ds:X509Data> element which is a child of the <ds:KeyInfo> element, and MUST appear in sequential order, starting from the signing key’s certificate. The certificate of the trust anchor is omitted (since it cannot necessarily be determined to be trusted).

Appendix A. Hash algorithm: HBES SHA-1

HBES SHA-1 is a hash algorithm that utilizes SHA-1. As specified in [SHA-1], SHA-1 produces a 160-bit digest (hash) from a message. The HBES SHA-1 algorithm is the following: The SHA-1 algorithm is given a 128-bit value as input and outputs a 160-bit value. The 128 most significant bits are extracted from the 160-bit output. Thus, the HBES SHA-1 algorithm produces a 128-bit digest from a 128-bit input message.

The hash produced as a result of executing the HBES SHA-1 algorithm using seed S_i as input is denoted T_0. The HBES SHA-1 algorithm can then be executed again, this time taking T_0 as input. The result is designated T_1. This can be input to HBES SHA-1, producing T_2, and so on. Thus, T_{j-1} designates the result of performing the hash j times. Figure 1 shows HBES SHA-1 being executed multiple times, starting with seed S_i. Table 4-2 shows the SHA-1(160-bit) and HBES SHA-1 (128-bit) hash outputs for performing the HBES SHA-1 algorithm three times, starting with the specified random seed S_i.

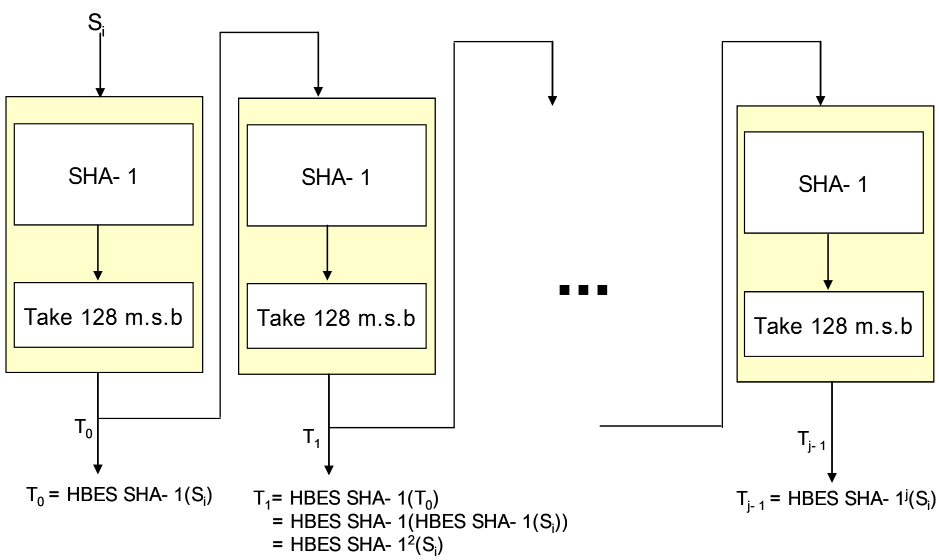

Figure 1. Performing HBES SHA-1 j times, starting with seed S_i

A seed S_i		0x4ecc9377b6b6229549b5b941cde9ef1b
Result of hashing once	SHA-1	0x2f715941d912cd23a2f0492346ca1c671A07759f
	HBES SHA-1	0x2f715941d912cd23a2f0492346ca1c67
Result of hashing twice	SHA-1	0x27682ebad05eeb3e8578f2eb379519b46e9fe685
	HBES SHA-1	0x27682ebad05eeb3e8578f2eb379519b4
Result of hashing three times	SHA-1	0x862bca975cbf2c1fa9c78adfd8e133dfcf7b0927
	HBES SHA-1	0x862bca975cbf2c1fa9c78adfd8e133df

Table 4-2 An example of SHA-1 and HBES SHA-1

Appendix B. An Example HBES Key Tree (HKT)

Figure 2 shows an example of a 4-ary tree and the **npid** and **devid** (see §3.2.2) of one of the Leaf Nodes.

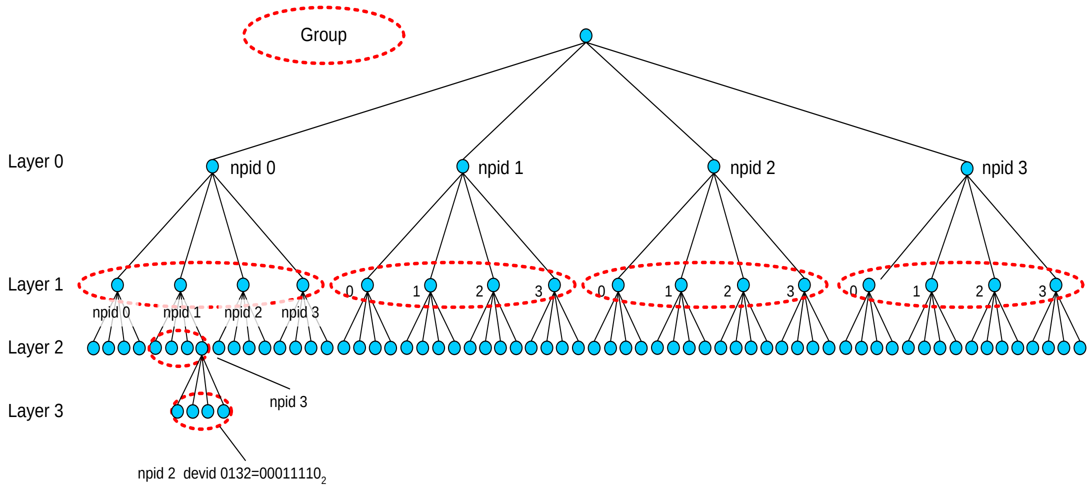

Figure 2. An example of HKT (t = 4)

Appendix C. An Example HBES Node Key Set (HNK)

Figure 3 shows the composition of the HBES Node Key Set (HNK) for each of the Nodes in a Group with eight Nodes. In a t-ary tree, there are t Nodes per Group and each Node has t keys. The Nodes in a Group are numbered from left to right as Node 0, Node 1,... Node t-1. The figure shows the keys for the Nodes in a Group in an 8-ary tree. The 8 keys of each Node (i.e., its HNK) are shown in the column below the Node designation (Node 0, Node 1, etc.).

Each Node i includes a seed value S_i as one of its keys. All seed values are randomly generated and pairwise independent. In addition to the seed value, each Node contains t-1 (7 in this example) keys whose values are the result of applying a hash function, denoted by h, one or more times to each of the seed values of the other Nodes in the Group. The notation h^j (S_i) indicates the result of applying the hash function j times, with S_i as the initial input to the hash function. In Starfish, the hash function is HBES SHA-1, as described in Appendix A. The assignment of keys is described in §3.3.1.

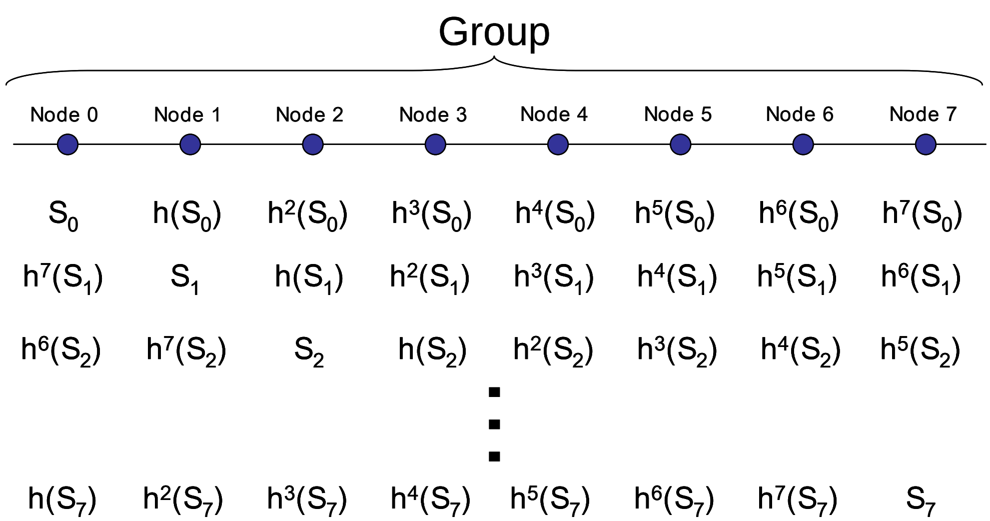

Figure 3. An example of the HNK in a Group of eight Nodes (t = 8)

Appendix D. Example of Exclusion

D.1. *Determining the KEKs in one Group*

Figure 4 shows a Group in an 8-ary tree where Node 1 and Node 6 are excluded. In this example, there are only two Representative Excluded Leaf Nodes (**E**=2), and the Group depicted is the single Group at Layer 0. Node 1 is the Layer 0 Ancestor Node of the leftmost Representative Excluded Leaf Node, and Node 6 is the Layer 0 Ancestor Node of the other Representative Excluded Leaf Node. (When a Leaf Node is excluded, all its Ancestor Nodes are also excluded.)

As described in §3.4, an Interval is a set of cyclically consecutive Non-Excluded Nodes in a Group. In this example, the Group contains two Intervals. Interval 1 consists of four Nodes (Node 2 through Node 5), and Interval 2 consists of two Nodes (Node 7 and Node 0). The start Node of an Interval is the Node whose immediate left neighbor is an Excluded Node, so the start Node is Node 2 for Interval 1 and Node 7 for Interval 2.

A Key Encryption Key (KEK) is determined for each Interval in a Group. A KEK is used to encrypt a Broadcast Key (bK) such that only the Non-Excluded Nodes in the Interval have or can calculate the KEK and thereby decrypt the bK. The KEK for an Interval is the result of hashing the start Node's seed value one time less than the number of Nodes in the Interval. Thus, the seed value for Interval 1 is S_2, since Node 2 is the start of Interval 1, and the KEK is $h^3(S_2)$, the result of hashing S_2 three times. If you look at the keys for each of the Nodes in the Group (see Appendix C) , you can see that only the Nodes in Interval 1 can calculate this KEK. That is, Node 5 directly contains this key, Node 2 has the seed S_2 and can calculate the key by hashing S_2 three times, Node 3 has $h(S_2)$ and can calculate the KEK by hashing that twice, and Node 4 has $h^2(S_2)$ and can calculate the KEK by hashing that once. Similarly, the KEK for Interval 2 is $h(S_7)$, the result of hashing S_7 one time (one time less than the Interval length of 2).

The BKB is as follows. Since only the Layer 0 Group is shown in the figure, the Excluded Node IDs shown are only those for Layer 0.

<BKB len., S. ver., R. ver., KCD, Res., # of E. Nodes = 2>

<000001_2, 000110_2, …, Sig., E($h^3(S_2)$, bK), E($h(S_7)$, bK)>

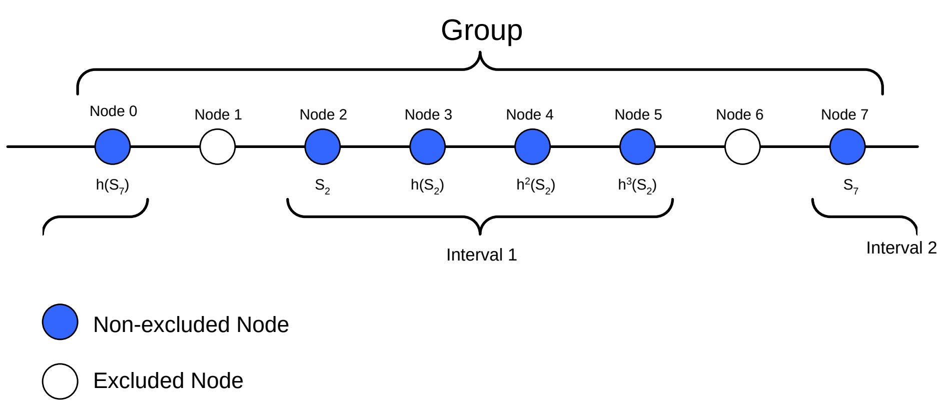

Figure 4. An example of determining KEKs (t = 8)

D.2. *The List of Excluded Node IDs*

Figure 5 shows an example of a 4-ary tree in which 7 Leaf Nodes are excluded: all Nodes in the 10th Group in Layer 3, the 1st Node in the 22nd Group in Layer 3, and the 2nd and 3rd Nodes in the 55th Group in Layer 3. Since the Ancestor Nodes of each excluded Leaf Node are also always excluded, such Nodes are also depicted as excluded in the figure.

When constructing the BKB, the value **E** is the number of Representative Excluded Leaf Nodes, as defined in §3.4. For the 10th Group in Layer 3, only the leftmost Node in the Group is a Representative Excluded Leaf Node, so in this example, **E** is 4. In the BKB , the list of Excluded Node IDs appears following **E**. The list contains **E** entries per Layer, so in this example there are 16 entries (4 Layers * 4 entries per Layer). The Excluded Node IDs are constructed as described in §3.4. Note that for a 4-ary tree, each **gid** and **npid** are 2bits.

<BKB len., S. ver., R. ver., KCD, Res., # of Representative Excluded Leaf Nodes E = 4>

<0000_2, 0001_2, 0011_2, 0011_2, 0010_2, 0101_2, 0001_2, 0001_2>

<0001_2, 0101_2, 0010_2, 0010_2, 1100_2, 0000_2, 0101_2, 0110_2>

<Sig.> <E(S_2 of the first group in Layer 0), bK)>

<E($h^2(S_3)$ of the 1st group in Layer 1), bK)> <E($h^2(S_2)$ of the 2nd group in Layer 1), bK)>

<E($h^2(S_2)$ of the 4th group in Layer 1), bK)> <E($h^2(S_2)$ of the 3rd group in Layer 2), bK)>

<E($h^2(S_2)$ of the 6th group in Layer 2), bK)> <E($h^2(S_3)$ of the 14th group in Layer 2), bK)>

<E($h^2(S_1)$ of the 22nd group in Layer 3), bK)> <E($h(S_3)$ of the 55th group in Layer 3), bK)>

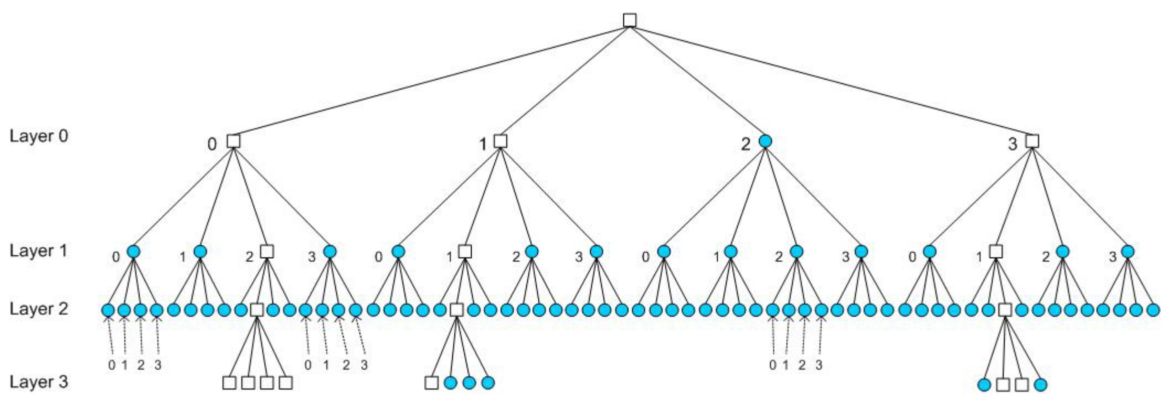

Figure 5. An example having 7 Excluded Leaf Nodes (t = 4)

Appendix E. BKB Example

This section provides an example of how HBES Node exclusion works, using the full structure of HBES for 2^{64} users and the complete 16-ary tree, as shown in Figure 6. In this example, there is one Representative Excluded Leaf Node and the HNKs are those specified in Appendix F. Assume that the Broadcast Key is the following

bK = 0x0d93e99d7f10aef880d82cf95bed6e41

and parameters for RSA are as follows:

- RSA composite : pq = 0xa4149336adb7e2997d01f023754abbce28193bb121d64157970826bc37d654162c3846c303cf4ecfdc2a357447a134f03d89cb0ddb332e83313d1c11bcf342509f461aa52ba0163cf25072ba1d6955bf66edd0274ec9e22981e096030590abdd210f1cb5039b52372affedb69ca47d5ec118cadd4c1161d790415c59821bbeb5
- p = 0xcd44f67cb7ece79b27ee3c941e4e12bbd2da1a065cc5f9308d00d26fba73b6fb5ac0bb93389753d0976b55503e867a82e047715f34ff25e816a24e03111afd91
- q = 0xcca1a7a4fb9a153660d0c179bfa41968ecc3ac5216b046f0db495c240678c797605e20027c3aea6d0686a904944b5ada3a799526d0c7e99a3402d35112422ce5
- Public key : e = 0x010001
- Secret key : d = 0x1e78bfba38dadc975ab2e071055862b66f95f812f650bf03d045b043e62ec4a0f5876204d7914976aad19fe9bf5fbde01bdd9a3b318938cb1e7ad5daa97797c9eb1a879a36d49a76eb3d1b28de047852835e06e446b12a774ca1350661fe077738a53cc2deb3d324d6404f439ee2a356e3188b9b15afb9c8a24bd85e2ff13ec1

In this example, Node 4 of the 2^{nd} group in Layer 15 is excluded, so its Ancestor Nodes are also excluded. As with Figure 5, the Excluded Nodes are depicted using squares in Figure 6. None of the keys in the HBES Node Key Set (HNK) of Node 4 of the 2^{nd} group in Layer 15, nor any keys that can be calculated from the keys in its HNK, should be used as the KEK for Layer 15, since this would allow Node 4 to access the Broadcast Key. Similarly, none of the keys in the HNKs for that Excluded Node's Ancestor Nodes, nor keys that can be calculated from the keys in those HNKs, should be used as KEKs for their Layers. Since it is desirable to limit the size of the BKB, it is best to use the smallest possible subset of the remaining keys that cover all Non-Excluded Nodes. The algorithms for specifying Excluded Node IDs and for determining and utilizing KEKs (see §3.4) ensure that is the case.

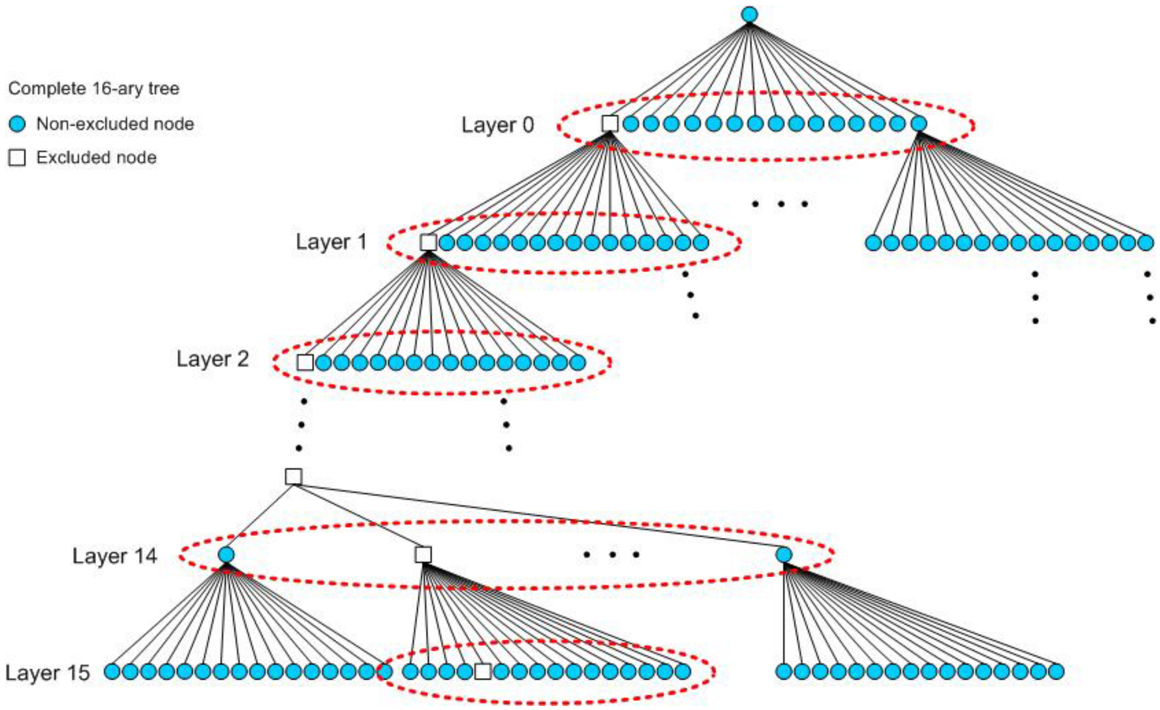

Figure 6. An example of one Excluded Node

In this example , the BKB is

⟨440, 0, 1, 0x826f7c67388c79e6bff174e35cf5307c, 0, 1⟩
⟨00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2⟩
⟨00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000000_2, 00000001_2, 00000100_2⟩

⟨Sig.⟩

⟨E(h^{14}(S_1 of the first Group in Layer 0), bK) = 0xd8e774066f0c4f76cd418caeddef34b8>
⟨E(h^{14}(S_1 of the first Group in Layer 1), bK) = 0x10522a5af3137596b028361877ed396e⟩
⟨E(h^{14}(S_1 of the first Group in Layer 2), bK) = 0x6ae668050a20decf917e2fa79c9264eb⟩

…

⟨E(h^{14}(S_2 of the first Group in Layer 14), bK) = 0x6800283da05f3729111b44a4fbd9f523⟩
⟨E(h^{14}(S_5 of the second Group in Layer 15), bK) = 0x034e8f0c075d53b854149c0e70e351de⟩.

The following table precisely describes the BKB for Figure 6.

Contents	Value
BKB length (4 bytes)	440
Structure version (4 bytes)	0
Revocation version (4 bytes)	1
Key check data (16 bytes)	0x826f7c67388c79e6bff174e35cf5307c
Reserved (8 bytes)	0

The number of Representative Excluded Leaf Nodes (4 bytes)	1
Excluded Node ID for Layer 0 (8 bits) || Excluded Node ID for Layer 1 (8 bits) || … Excluded Node ID for Layer 13 (8 bits) || Excluded Node ID for Layer 14 (8 bits) || Excluded Node ID for Layer 15 (8 bits)	00000000_2 || 00000000_2 || … 00000000_2 || 00000001_2 || 00000100_2
Signature (128 bytes)	0x0d571aefb4c3e54852b84435e87b9483 c2134212605556ad46190807a1fecc89 3658934227f51483b09e04a811dbde90 4fabed338d9dd12c88e3d57a68a4d4f8 e9e01d139ea336b87fa6c22d3b401163 b5a14065f612104ebc1296b6fd85f6ee 24c492dddff6b39813c842850265b442 d4b7696c9f0c96a8bc84a119781bb26b
Enc. bK using KEK generated from the Interval in Layer 0 || Enc. bK using KEK generated from the Interval in Layer 1 || Enc. bK using KEK generated from the Interval in Layer 2 || … Enc. bK using KEK generated from the Interval in Layer 14 || Enc. bK using KEK generated from the Interval in Layer 15	0xd8e774066f0c4f76cd418caeddef34b8 0x10522a5af3137596b028361877ed396e 0x6ae668050a20decf917e2fa79c9264eb … 0x6800283da05f3729111b44a4fbd9f523 0x034e8f0c075d53b854149c0e70e351de

BKB for Figure 6

Appendix F. Example of KEKs and HNKs

The following tables show the KEKs and HNKs of Nodes in Figure 6.

Layer	KEK
0	0xdac9cce7ff9e4372bd426ab18a016f1d
1	0xcb7c1100ccd4c76fa2656a05a21e34a0
2	0x6c3f2fa0c14a9ec87831e727bc403c4d
…	…
14	0x6faab091e96b675602a0c0cffe2cf583
15	0x46c2fed9f61c4d356c484b71a9e71031

KEKs for Figure 6

Node 0	S_0	0xdbac74b2f6b0af904a53f7c05363a38c	Node 1	S_1	0xe4adf64314b08b1350067113705d103d
	$h^{15}(S_1)$	0x90188cbde20269797c90e1893a28b207		$h^{15}(S_2)$	0xa2d1b524c7356e276fb993a2565cba96
	$h^{14}(S_2)$	0x8169817b6793e46c7c84b3519aea0fc7		$h^{14}(S_3)$	0x57e71046e81b5323028ad6395b30d92d
	$h^{13}(S_3)$	0x3273a4e66d7cd36f308d20d9647d04b7		$h^{13}(S_4)$	0x72bad10cdd85e89e3069261dd628ff13
	$h^{12}(S_4)$	0xa2b645affea34d89fd4f7cba333b35a6		$h^{12}(S_5)$	0x555282ec0fdcbbe2742e4ae7435abc9d
	$h^{11}(S_5)$	0x3eed13dc306e7277eb5f612e7e9dc2bf		$h^{11}(S_6)$	0x29563150af58157dc44efc4fd0872dda
	$h^{10}(S_6)$	0x7468a8104b13af91c23a19c3818e966d		$h^{10}(S_7)$	0x69b35dff8ee59f2e88b84ed4d4ca820e
	$h^{9}(S_7)$	0xbc1684be93e0d276e6b628c25726c235		$h^{9}(S_8)$	0x3e937b0268b712352cff10da434779f1
	$h^{8}(S_8)$	0x20027414391b81fad72ce1c676f1525e		$h^{8}(S_9)$	0x468aeec5bd5788548aed6eea3e5e9c16
	$h^{7}(S_9)$	0x4596f7d9bb17bc72e99935d04012b5b5		$h^{7}(S_{10})$	0x8f31a7d5bf4e8b5277a4840e83c9e13e
	$h^{6}(S_{10})$	0x5a13d2f677208bb14960bf6d9f6848ec		$h^{6}(S_{11})$	0x8224f94c961b0625647c6b662fdf7759
	$h^{5}(S_{11})$	0xe97f6c5f026486650d0a890783b960cb		$h^{5}(S_{12})$	0x287e0b2e0d0a41e2a722103f15ad7c9f
	$h^{4}(S_{12})$	0xfd576dcb80bc556208dbb2f567c87fc1		$h^{4}(S_{13})$	0x850535954d90c69341f42be1cbb71a6c
	$h^{3}(S_{13})$	0x9d3d01ddee9c75985b88575624d6a81a		$h^{3}(S_{14})$	0x8b0e7f75cbcb707d360d0aef062b30c7
	$h^{2}(S_{14})$	0x8165298b423fc08d20a5658873ebc1d7		$h^{2}(S_{15})$	0xb947538ff3e4a99b8dd1a53253d75196
	$h^{1}(S_{15})$	0xffff5be2bb242ae55e36638c5b5ecd21		$h^{1}(S_0)$	0xc8cd9a8a63656c2cfa996fe040c28cb5
Node 2	S_2	0x4650ee7749a913552b8faee1f4f2de2f	Node 3	S_3	0x194271e78523089314abbf02d25612d0
	$h^{15}(S_3)$	0x5b8763fe6e45d70ee2b86123e49282b4		$h^{15}(S_4)$	0x4fde45b2ff9dd525c02f69919d27f878
	$h^{14}(S_4)$	0xf7c897115fc4835829fdeae2a221fb15		$h^{14}(S_5)$	0x46c2fed9f61c4d356c484b71a9e71031
	$h^{13}(S_5)$	0x0e52673d8464ec0c6533af930c49c3f6		$h^{13}(S_6)$	0x9e9263755a9b448c488743e46f9faf05

	$h^{12}(S_6)$	0x467b1d151508f7688a9deee2506b973b		$h^{12}(S_7)$	0x55dd0167cea89d736b84f7764b63f8bc
	$h^{11}(S_7)$	0x7c7546d092264eb476098a38705b4092		$h^{11}(S_8)$	0xa07cb1f096097eb0e25807ab0076671f
	$h^{10}(S_8)$	0x050bb39141ff78427e9ca7463923cb5b		$h^{10}(S_9)$	0xa2748f1f0ef1c43bcbbab0a22a96e2b1
	$h^{9}(S_9)$	0x29633230d43612eae5100769533848e9		$h^{9}(S_{10})$	0xf8de7bd41f9b1f5c27c5b4117ada7893
	$h^{8}(S_{10})$	0x78c6bafeb9a29226f7b708bfb57aabbf		$h^{8}(S_{11})$	0xd86c7695a6777df9316ceb5101d6b0d9
	$h^{7}(S_{11})$	0x8a7ad75a7c2e3fa89cdfb797a4404a32		$h^{7}(S_{12})$	0x1f9679f9226861587f9180eef024a283
	$h^{6}(S_{12})$	0xc52b0fe85371aebc61c65c997c9e1653		$h^{6}(S_{13})$	0x310b82924a2eef5e598ec61171628d13
	$h^{5}(S_{13})$	0x8ee6d42698c3ac9ef0eae6d9cf6e7237		$h^{5}(S_{14})$	0x8c62decb560b04570b49fa9a2ac1c90d
	$h^{4}(S_{14})$	0xd63f607a61be6486e148bc7cfb5c2143		$h^{4}(S_{15})$	0x12b241b765cb4d32d88f64aa80373fbd
	$h^{3}(S_{15})$	0xd6396fe03880a961c648df48372a9926		$h^{3}(S_0)$	0xee3da985bc818451f00d0aae830fdac8
	$h^{2}(S_0)$	0xca2bb9e33d0fc01bc515e65b6fb35c7b		$h^{2}(S_1)$	0xa335bf0183742b4d5075b62f0cba1662
	$h^{1}(S_1)$	0xa46ca30e4e459238ec4c0cda38d98994		$h^{1}(S_2)$	0x57a5431f5d2bdd9f9b881ad4ea9c0b87
Node 4	S_4	0x41286cc39e6fca793e948d2dd49bcd63	Node 5	S_5	0x763d3f600e0f9627184732ba48d4f246
	$h^{15}(S_5)$	0xd5f847e6cb46a1039c91ff605d2f5cde		$h^{15}(S_6)$	0x72c49f72ea3c311b2ab8c03c948555a5
	$h^{14}(S_6)$	0xa6bbdfc748829294b660c1389d77a565		$h^{14}(S_7)$	0x16ce4c5745cfff966e2127cd06141690
	$h^{13}(S_7)$	0xf6d29d7f030fd2489e60f48a4db67fb6		$h^{13}(S_8)$	0xd3f4d827a3430161ef066f55e4bfb6b6
	$h^{12}(S_8)$	0x19d9333d5fb738fa0d66cf447744de09		$h^{12}(S_9)$	0x84692a6782eec6aa6ab0c07f0d0aa8e3
	$h^{11}(S_9)$	0xf53af68c4d87bc73f527583ac9d8bfb0		$h^{11}(S_{10})$	0x711173a7e5f65330a3e6e635937cab1f
	$h^{10}(S_{10})$	0x08ae3fc80882a7676315b1d7fd79cb75		$h^{10}(S_{11})$	0xd8b44b359715cac912bc227f7ec676e7
	$h^{9}(S_{11})$	0xb50b707c9d9a8335e6a8e4e8e37f6020		$h^{9}(S_{12})$	0x6b40fd0b59642b3b3bb8791f662664b3
	$h^{8}(S_{12})$	0x4b83ceaef9e752b2972c9f049e88dbb2		$h^{8}(S_{13})$	0x9538ef33c3746ff4e8c13fa66bdab443
	$h^{7}(S_{13})$	0x2385fec2893e79129c2a3225b09013cb		$h^{7}(S_{14})$	0x33b3fd6b59613eca90a1dccdea40b26c
	$h^{6}(S_{14})$	0xed4b31d34daabb119f269a4bcaada007		$h^{6}(S_{15})$	0x1bc6ab83c4b855a339f3590ceee9bf07
	$h^{5}(S_{15})$	0x411de574ff0d9288648ac777f1b617f5		$h^{5}(S_0)$	0xef51a4ab3a3e60bdb083a5e06c88aba8
	$h^{4}(S_0)$	0xa9024744e542535292eccb7d06aa9f2d		$h^{4}(S_1)$	0xa27ada71feea8bd3ade55e9bff9f1023
	$h^{3}(S_1)$	0xe12ecbb50ead2b5f6daa143b70e13ea5		$h^{3}(S_2)$	0x170be024769b8dd182eff2b81f0c6943
	$h^{2}(S_2)$	0x7e4f0703ec05eb080de5dc8e7f77df70		$h^{2}(S_3)$	0xac3106aab96d7507faa557404f5ff173
	$h^{1}(S_3)$	0x3bcaad851f6040199957f85d57cb9ce1		$h^{1}(S_4)$	0x0c9b60a6400749cf843d7f687e356549
Node 6	S_6	0x98b252651f4c933a9ec1ac2ac470220e	Node 7	S_7	0xc3e0a3759d73e22ec6f4c17f9f2cccb6
	$h^{15}(S_7)$	0x05cac60d0aa9d3e480b2ea3a4f366e7d		$h^{15}(S_8)$	0x1e51f79c4ad9d9369df059e825c14e76
	$h^{14}(S_8)$	0xf32ee24c63435ce8cfd09489f29c3d75		$h^{14}(S_9)$	0x0e806d7779510d0ac84f1018cfd6c998
	$h^{13}(S_9)$	0xa97f1689730d077d24aee9ecbee41136		$h^{13}(S_{10})$	0xed44938ecf723e922b9464c1845effa3
	$h^{12}(S_{10})$	0xe960fa4b8ddfc1ffcb103051020312c6		$h^{12}(S_{11})$	0xc29e1343ac12fb8c977285a699908086

Refer to Notices on page 2 for important legal information

	$h^{11}(S_{11}$	0x018a1d95396c6efd6d432fe97e2c9dd9		$h^{11}(S_{12}$	0x1217f4c4ec58680bcb1497532b2d651c
	$h^{10}(S_{12}$	0x62e777c4661dce93ef7cc0c880c6fcd8		$h^{10}(S_{13}$	0x84b2fe6310ee8f56351af9dd064a2901
	$h^{9}(S_{13})$	0xcac0aae64623dcbfa98314487c613931		$h^{9}(S_{14})$	0xd5a9de4c00da23022ddaf494381942f7
	$h^{8}(S_{14})$	0x7a4d4fb9120b9ea0779d319aef45af80		$h^{8}(S_{15})$	0x4fd61fbcfe586144c9e680d0cfd58c98
	$h^{7}(S_{15})$	0x3677a81387170d0a06a88213832beec0		$h^{7}(S_{0})$	0x6bd5704bfef8b9d74d8c0d9be0073e18
	$h^{6}(S_{0})$	0x7b4094b00704f3f4ae339c01083b0704		$h^{6}(S_{1})$	0xd9ccad8bab2bd205deb898d515c65008
	$h^{5}(S_{1})$	0x94f390c2bb59ebb0797fa324cd935ed7		$h^{5}(S_{2})$	0xd99f3e85c4b6b38435e6921918927ad2
	$h^{4}(S_{2})$	0x61b077e95b670289e6145a1ab00c8c56		$h^{4}(S_{3})$	0xc0f1a7ad46f36f5c14b50ea82ca5434f
	$h^{3}(S_{3})$	0x65fac4b6135f7c0134bd51d106d54562		$h^{3}(S_{4})$	0x3cfc2a387dbc8638764c6313a0d12865
	$h^{2}(S_{4})$	0xf7c703aaa21c08440819b949789d4fd0		$h^{2}(S_{5})$	0xf8ec1661a6c590b080bdfe764d134298
	$h^{1}(S_{5})$	0x09fa9b7eae0c2a28f0e8b59568b3de38		$h^{1}(S_{6})$	0xf2ba0d7abb0e499f2a0fe3ce3bf61216
Node 8	S_{8}	0x6eea3a5008b1ed3b99ec01ad488910c6	Node 9	S_{9}	0x0d14d39732edabefcd3a1c3ba59b4dcf
	$h^{15}(S_{9})$	0xc83b2eb193d4571a978f59ac10c4897e		$h^{15}(S_{10})$	0x008d9710c5a3e0fa519559fd5b153d9e
	$h^{14}(S_{10})$	0x852b5a092cf9667ea8c0a7dd0d0a6698		$h^{14}(S_{11})$	0xc455a1367b1096f7e3241f4e0c89af96
	$h^{13}(S_{11})$	0x5ccda2894b01d1898541cad7204421f9		$h^{13}(S_{12})$	0xfe00f5cb7ec8924ffbeec54f2f1871ce
	$h^{12}(S_{12})$	0xf8f898c915416d52dad2c8fe59c7c284		$h^{12}(S_{13})$	0x888e5607e99372e52f41a5ac57e96d87
	$h^{11}(S_{13})$	0x9875de9c19bdb40dd221d2b20d0a35a7		$h^{11}(S_{14})$	0x3035a52c783a746544f35fdd0e1737d5
	$h^{10}(S_{14})$	0x42cf1b4cd295f3de7bd838bc0d0aac05		$h^{10}(S_{15})$	0xd6e05de3a9a970fed7801275c4d3a861
	$h^{9}(S_{15})$	0x543927a5fb31f0eae20e95d7f85027b6		$h^{9}(S_{0})$	0x34ad80aebfa63e0f45b8c33e259518f6
	$h^{8}(S_{0})$	0xeeeac49a87d206a634ade9e2c757d4fa		$h^{8}(S_{1})$	0x2738dbad2137e4782df443571af9d63d
	$h^{7}(S_{1})$	0x05ceea53b1de391838dd9be0f29932a6		$h^{7}(S_{2})$	0x975242671136093d514b396a0174c00b
	$h^{6}(S_{2})$	0x1451fb23f09bc2866ea24a4a7e014753		$h^{6}(S_{3})$	0x2f8193289e8bfa2972bcbc07fe87146a
	$h^{5}(S_{3})$	0x18c475931740e719f67ebb1ca328d6ce		$h^{5}(S_{4})$	0x63cbe67bd77635b6be2d863d45f70234
	$h^{4}(S_{4})$	0xdee26daf889159659ba84a93c8625dcb		$h^{4}(S_{5})$	0xa0a2595571c2467309a26cbe26b3621c
	$h^{3}(S_{5})$	0x4eb724dc7f7c751356b547b3d300db97		$h^{3}(S_{6})$	0x87c354e3ef2befe9f9e975221d8379dd
	$h^{2}(S_{6})$	0x296640dc33e0279cdd5f0e55a75e1865		$h^{2}(S_{7})$	0x9facf07289ef874cc8d8c9b57182ed97
	$h^{1}(S_{7})$	0xd0af18eccad22d44a3f4ce8b28e650aa		$h^{1}(S_{8})$	0x466210ba9b30a0270659b5ae82387578
Node 10	S_{10}	0xdf17d7682b6c261c86cce03f758430d5	Node 11	S_{11}	0x10b6a7c540b6b9884938ba2b9733643f
	$h^{15}(S_{11})$	0x351a8b582990c0a716c9949d849c76c9		$h^{15}(S_{12})$	0xfc76d66ba44a52d6a63e42921d89cbeb
	$h^{14}(S_{12})$	0x347ba1ecc6a30973b2d129483c18be09		$h^{14}(S_{13})$	0x38eb321360043410cb356899949f03a3
	$h^{13}(S_{13})$	0xba82da66de5f3c0f3bf9f41150ec7025		$h^{13}(S_{14})$	0xf02a804b8b39f4ac5ec6d103975be197
	$h^{12}(S_{14})$	0xb2a21f19a6faa60d71b6b2952adc0fd7		$h^{12}(S_{15})$	0x40971c1b1275e8cd25d845ef779a16d6
	$h^{11}(S_{15})$	0x1431641d5102ce8e4fc52e559a0d0a64		$h^{11}(S_{0})$	0x1ad4d1feadacbf96fadd3103bac5d3b8

	$h^{10}(S_0)$	0x9a209abd294cba104eaec2f9db52a57d		$h^{10}(S_1)$	0xfb50cdbc01625c5c65ce7d180d8139e4
	$h^{9}(S_1)$	0x42836bf3a6c76114ba7fd0bacc775aba		$h^{9}(S_2)$	0xb3e6b8aa8f10291d8ac1b4676b0be84e
	$h^{8}(S_2)$	0x37403a25da8ddda557c8331095ff2f55		$h^{8}(S_3)$	0x6e7f4ab49f3cdd72d2cb8af1cfe99001
	$h^{7}(S_3)$	0x174285db25af70092c803844121d7022		$h^{7}(S_4)$	0x034730b93d1d90c33bf1e4f5f45e14a0
	$h^{6}(S_4)$	0xf99dcc18830b215e0430a85d61e8be7f		$h^{6}(S_5)$	0x3660577d8bb6769198976d2ae1a24d85
	$h^{5}(S_5)$	0xd9518cc6153eb1d5765cc0df27dca07b		$h^{5}(S_6)$	0x0f7af0071b7f9badb009020df2256679
	$h^{4}(S_6)$	0x4e3e2873bdfd40ed81ee2dbcf51b98a3		$h^{4}(S_7)$	0xc9f59569b06d3c6da4835707205bfb1e
	$h^{3}(S_7)$	0x59aafcfc9c84c71c07f68cdbf6f9a68b		$h^{3}(S_8)$	0x2f68278e612b0245e3c6cbd848b4a717
	$h^{2}(S_8)$	0xb0737c4ac32081a2a4760b755ea3a577		$h^{2}(S_9)$	0x283d3414c1579a4dccdb6f2eaf3e3fbc
	$h^{1}(S_9)$	0xb808d53746fec97ff9d55c8f4e17546b		$h^{1}(S_{10})$	0xc4ec264457131fdd8649ebd312642aa5
Node 12	S_{12}	0x53a1bb5395c8ba204c7c54bb9588d2b9	Node 13	S_{13}	0xd75be0f958874805f4813f1980f2c853
	$h^{15}(S_{13})$	0x0d2330ff3cb698571ca0c45f6a453d33		$h^{15}(S_{14})$	0x9a09cac050b6b831c281c519571695ba
	$h^{14}(S_{14})$	0x216d9f32a41f462926d1142d3f8511e3		$h^{14}(S_{15})$	0xc3e067f530d76362626f3c2f1ba61538
	$h^{13}(S_{15})$	0x8d44b1cbeb059fbcfb52e32573d4878f		$h^{13}(S_0)$	0xd70ccddca7723bc802dbc68e8d1120aa
	$h^{12}(S_0)$	0x187698d2a1a51e45289fce30b97b4f1f		$h^{12}(S_1)$	0x7f607e8fde22c21bb59357573bd741ab
	$h^{11}(S_1)$	0x1e67307c8eacb60ffa10642261710c12		$h^{11}(S_2)$	0x7d5ca7742ae469d73e3bcb414f259f1f
	$h^{10}(S_2)$	0xbcdc4759b58890a6e581ec606f8bf857		$h^{10}(S_3)$	0xefe666b75bbd6a16e9ac18691a09b523
	$h^{9}(S_3)$	0x22fa6431425c86039ed3a13fb0ed02c4		$h^{9}(S_4)$	0x9438ca590e95066d80229039a388d701
	$h^{8}(S_4)$	0x6526295bd297ff53b82bd120f981a3bf		$h^{8}(S_5)$	0xb77e64f18dfa12af37e540b0b0d5804a
	$h^{7}(S_5)$	0x66ec769561b32dfb0d0a237c11d8d1a6		$h^{7}(S_6)$	0x3b53fe08fead281ccd89e902e6e99e5a
	$h^{6}(S_6)$	0x25669fa6b435b22538eb43ab5b8b4973		$h^{6}(S_7)$	0xb8db60c88e1869c65bc02688835fb75c
	$h^{5}(S_7)$	0x7a147f678dc3a4b3ff81d8cbf7057178		$h^{5}(S_8)$	0xa16fe2cb7c04e2aefdf6cca54dd48813
	$h^{4}(S_8)$	0xed7fc773829584dd897d4740d95df584		$h^{4}(S_9)$	0x102b534e39773dbb3ea07c194ec261d7
	$h^{3}(S_9)$	0x89f31b08ae95df212f93529bead067b9		$h^{3}(S_{10})$	0x525e35cf012ce90e3a661f0d0a300587
	$h^{2}(S_{10})$	0x8d84c4d691b7919c5366f2047ed99925		$h^{2}(S_{11})$	0x41bdfbb44fa6e9301a096f724f19e827
	$h^{1}(S_{11})$	0x3d06177da437d89c0830bc357b0ec3f7		$h^{1}(S_{12})$	0x497cf3175687963a90cc1c9c06c1a61e
Node 14	S_{14}	0xf7fc4f5543043f2dbaa1028931b9ff4f	Node 15	S_{15}	0x2a120f7f674faf037008c33699937ed1
	$h^{15}(S_{15})$	0xb446982e529cbdfe277d31921a5c0c07		$h^{15}(S_0)$	0x4662a8838ed854322db7fd21e9314bde
	$h^{14}(S_0)$	0xc93d26cea5bfee515f0f63b22f9f48ef		$h^{14}(S_1)$	0x7022747f593ddbdcd135c8c9dce7159c
	$h^{13}(S_1)$	0xde957246edd618cdab294d26c5ef78ce		$h^{13}(S_2)$	0x4476bc2b321bffeae59648a88cd8b50e
	$h^{12}(S_2)$	0x54dc84e1909a21419a0559a27ab8d84c		$h^{12}(S_3)$	0x18cd42d33db9f9ce92a98db4762c647b
	$h^{11}(S_3)$	0x3b1469d75238c2a029ac1c8ff858ceef		$h^{11}(S_4)$	0x13e62a8742e8d8956745f221139d763b
	$h^{10}(S_4)$	0x621a29c7a8ddf6022c37ebb4c3f6cac2		$h^{10}(S_5)$	0x378d5aecba882a6d5d949550a4b1de1b

Refer to Notices on page 2 for important legal information

	$h^9(S_5)$	0x6912cfa082d8c5210e6d9829de389f94		$h^9(S_6)$	0x35aa71ce07183502aaa3b23219ac9ca7
	$h^8(S_6)$	0x97e89cc634175f76a6c4e6d46180e3b8		$h^8(S_7)$	0xd45f04aa8d43a9fcf0f7d0c5e5552792
	$h^7(S_7)$	0x69049052c24be208cb8618ff47360601		$h^7(S_8)$	0x4a7c81a8d714c611e9f9dc1322fbbbe1
	$h^6(S_8)$	0x86882346368fc83683a5cdd687224b40		$h^6(S_9)$	0xa885a9de941370fa424a9ee4b9f729a6
	$h^5(S_9)$	0x043301c76b55b0c8c435c560b64fc67c		$h^5(S_{10})$	0x26b4efc68d3c45e0f9ead39313da68f5
	$h^4(S_{10})$	0xd16e14aed7b3d7fe7e78c7e6231a985a		$h^4(S_{11})$	0x48a9b84246019c7d6c48ccde9324c9d3
	$h^3(S_{11})$	0x11e2c396db87113aa7ae0c774f86b525		$h^3(S_{12})$	0xf50321a152619a7dcde88c2b1a0e130b
	$h^2(S_{12})$	0x6ecee1780d3b715606e9d2760d0a1e55		$h^2(S_{13})$	0x50db026d1e70c18549fc4f1c629f3936
	$h^1(S_{13})$	0x09731833959edb527648f98b3961bf9b		$h^1(S_{14})$	0xd3983f880c2543beed0c1b85bc0eb6db

739 **HNKs of Node 0 ~ Node 15 of the second Group in Layer 15**

Appendix G. Pseudocode for bK Extraction

The following pseudocode is an example showing how a Node extracts a Broadcast Key bK from a BKB it receives.The Node refers to the BKB in order to determine which key in its HBES Node Key Set or in one of its ancestor's HBES Node Key Sets it can use in order to calculate one of the KEKs used to encrypt the Broadcast Key. It then uses the KEK to decrypt the encryption that was a result of encrypting the bK using that KEK, and the result of the decryption is the bK.

```
// Start
Read (nr = # of Excluded Nodes )
start = 0
end = nr
nKEK = 0 // Location of KEK for the Node

for ( i = 0 ; i < 16 ; i++ ) {
        npid = the ith left most 4 bits of devid
        prev_tag = -1
        CountKEK ( 0 , start ) // Count Intervals in Groups not containing the Node or its Ancestor Node
        match = 0 // 1 if npid == the right most 4 bits of curr_tag
        for ( j = start ; j < end ; j++ ) {
                curr_tag = the jth left most 8 bits in the list for Layer i
                // Find both start point and end point of the corresponding Group for the Node
                if ( match == 0 AND npid == the rightmost 4 bits of curr_tag ) {
                        match = 1
                        start = j
                        end = start - 1
                }
                if ( match == 1 AND npid == the rightmost 4 bits of curr_tag ) end++
                // Count Intervals in Group containing the Node or its Ancestor Node.
                // And, if the Node is in this Interval, this algorithm is terminated.
                if ( Interval exists between curr_tag and prev_tag ) {
                        nKEK++
                        if ( npid in the Interval ) {
                                Find KEK for the Node using curr_tag and prev_tag
                                Return bK = Decrypt ( nKEKth encrypted bK with the found KEK)
                        }
                }
                prev_tag = curr_tag
        }
        CountKEK ( end , nr ) // Count Intervals in Groups not containing the Node or its Ancestor Node
```

```
}
//End
CountKEK ( S , E )
//Start
for ( j = S ; j < E ; j++ ) {
            curr_tag = the j^th leftmost 8 bits in the list for Layer i
// If the leftmost 4 bits (the gid) of both prev_tag and curr_tag are equal, the two Nodes
// are in the same Group. Then, check there exists a jump (Interval) between the rightmost
// 4 bits (the npid) of both prev_tag and curr_tag. If there exists not only an Interval
// containing Node 0, but also an Interval containing Node 15 in one Group, then those are the
same Interval.
           if ( ( gid of curr_tag != 1111_2 ) AND ( Interval exists between curr_tag and prev_tag ) )
nKEK++
           prev_tag = curr_tag
}
//End
```

Appendix H. Starfish XML Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
           targetNamespace="http://marlin-drm.com/starfish/1.2"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xmlns="http://marlin-drm.com/starfish/1.2">
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
  <xs:element name="BroadcastKeyBlock" type="BroadcastKeyBlockType"/>
  <xs:element name="RevocationInformation" type="RevocationInformationType"/>
  <xs:complexType name="BroadcastKeyBlockType">
    <xs:sequence>
      <xs:element ref="RevocationInformation"/>
      <xs:element name="EncryptedBroadcastKeys" type="xs:base64Binary"/>
      <xs:element ref="ds:Signature"/>
    </xs:sequence>
    <xs:attribute name="keyTreeName" type="xs:anyURI" use="required"/>
  </xs:complexType>
  <xs:complexType name="RevocationInformationType">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="structureVersion" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="revocationVersion" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="distributionURIs" use="required">
          <xs:simpleType>
            <xs:list itemType="xs:anyURI"/>
          </xs:simpleType>
        </xs:attribute>
            <xs:attribute name="issuedOn" type="xs:dateTime" use="required"/>
            <xs:attribute name="nextUpdate" type="xs:dateTime" use="required"/>
        <xs:attribute name="ID" type="xs:ID" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```