音频隐写术：解决怎么嵌入的问题

原创

唠嗑！ 已于 2022-04-11 14:00:37 修改 40 收藏 3
分类专栏： 多媒体编码安全 文章标签： 网络安全 语音识别 音视频 视频编解码 系统安全
于 2022-04-07 15:39:25 首次发布

前言

在之前的系列文章中，已经讨论了**编码参数域**的隐写。本文章将从另一个**嵌入域**方向叙述隐写：**熵编码域**。对于编码参数域来讲，修改编码参数将引起很多关联的**频域系数**发生改变，此举对音频的内容修改很大，也不能精确控制，最终导致对统计特性的改变也很大。

相反地，量化系数修改的方式更加接近原子操作，能够定量精确控制**音频内容质量修改**和**噪声扩散**。当然，随意修改熵码字也会导致码流膨胀，进一步导致编码器崩溃。从可行性角度来看，隐写算法要兼容编码标准、有约束地修改量化系数，保证修改后的哈夫曼码流的长度保持不变。

一. 三种基本嵌入方式

以下三种嵌入方式都是基于MP3码流结构。

1.1 等长熵码字替换嵌入

在编码标准中提供了很多张哈夫曼码表，每张码表都会存在很多码字长度相等的哈夫曼码字。所以可以使用码长相同的哈夫曼码字替换来嵌入隐藏信息，采用等长码字替换方式是保持码流长度的前提之一。如下图替换例子：

此时替换是基于一个安全性假设：根据哈夫曼码字构造原理，码长相同的码字统计概率相近，所以导致等长哈夫曼码字替换对码字分布的影响很小。

1.2 系数符号位反转嵌入

根据MP3编码标准，非零系数使用一个比特来表示其符号信息，其中比特“0”代表正数，比特“1”代表负数。所以，通过**反转编码符号位**来嵌入信息也不会改变原始码流的结构和长度。当然，为了不影响音频的听觉质量，对修改**符号位的位置**以及**系数值**有一定的限制。

此时的替换是基于一个安全性假设：由于量化系数服从0值对称的广义拉普拉斯分布，所以系数符号位翻转能够对系数直方图攻击免疫。

1.3 linbits位LSB嵌入

当系数值超过15时，就需要使用linbits位来编码系数溢出部分，每张哈弗曼码表分配的linbits位长度是固定的，这样的linbits位的编码规则等价于定长编码。所以采用linbits位LSB嵌入方式也能够保持修改后码流的结构与原始码流结构的一致性。

此时的替换基于一个安全性假设：linbits为LSB嵌入的安全性与通常的LSB算法是类似的

二. 码字映射表的构造

2.1 解决码字替换规则问题

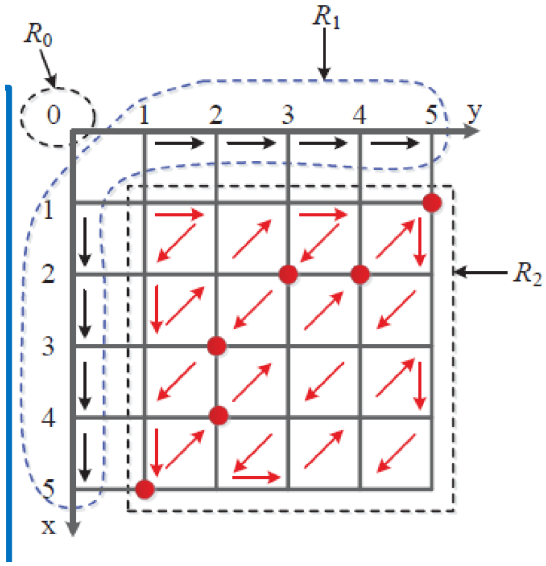
令$h_i^{(k)}$表示第k张哈夫曼码表中的第i个码字，$< x_i^{(k)}, y_i^{(k)} >$表示与码字$h_i^{(k)}$对应的频域系数对。因此，对$\forall i \neq j$，可相互替换的哈夫曼码字对$h_i^{(k)}$和$h_j^{(k)}$需要满足如下三个条件：

> （1）**哈夫曼码字长度。**哈夫曼码字$h_i^{(k)}$和$h_j^{(k)}$码字的长度相等，如下数学表达形式：
>
> $$||h_i^{(k)}|| = ||h_j^{(k)}||$$
>
> （2）**系数符号位个数。**频域系数对$\langle x_i^{(k)}, y_i^{(k)} \rangle$和$\langle x_j^{(k)}, y_j^{(k)} \rangle$具有相同的符号位个数，如下数学形式：
>
> $$\delta(x_i^{(k)} = 0) + \delta(y_i^{(k)} = 0) = \delta(x_j^{(k)} = 0) + \delta(y_j^{(k)} = 0)$$
>
> 其中δ函数的定义如下：
>
> $$\delta(x = m) = \begin{cases} 1, & x = m \\ 0, & x \neq m \end{cases}$$
>
> （3）**linbits位标志。**频域系数$x_i^{(k)}$linbits位的存在性与$x_j^{(k)}$一致。当然，$y_i^{(k)}$和$y_j^{(k)}$也同样满足此条件。

2.2 构造映射表

映射表在构建时，需要使得码字替换对频域系数变化造成的影响尽可能地降低，同时需要控制**搜索算法的时间复杂度**。具体有三个步骤，如下：

步骤一：先对码字搜索空间采取**Zig-zag**方式进行遍历。以7号码表的Zig-zag搜索为例，如下：



步骤二：令$\Pi^{(k)}$表示包含第k张码表中所有哈夫曼码字的集合，由此可将$\Pi^{(k)}$分成两个子集：$\Pi_u^{(k)}$和$\Pi_v^{(k)}$，如下：

> $\Pi_v^{(k)}$集合：可用码字，也就是可隐写码字，可被用来嵌入信息
>
> $\Pi_u^{(k)}$集合：都是不可用于隐写的哈夫曼码字

$\Pi_v^{(k)}$和$\Pi_u^{(k)}$可以通过迭代的方式生成：首先将$\Pi_v^{(k)}$初始化为ϕ，如果码字对(h_i, h_j)满足可替换码字的三个限定条件，就将h_i和h_j放入$\Pi_v^{(k)}$，否则将它们放入$\Pi_u^{(k)}$；重复上述操作，直到$\Pi_u^{(k)}$为ϕ。

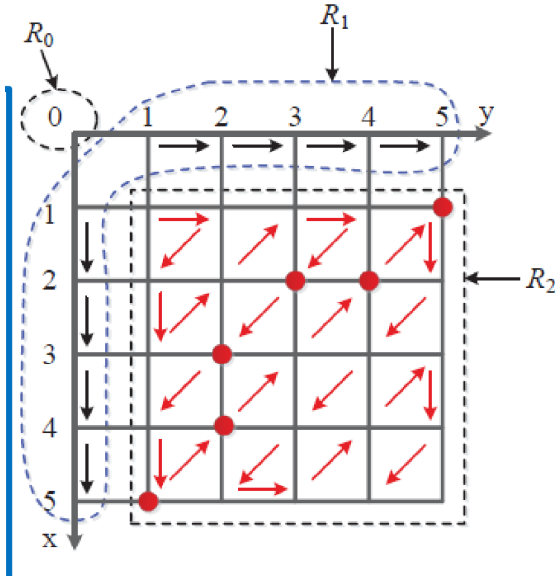
步骤三：对于可用码字空间$\Pi_v^{(k)}$中的每个哈夫曼码字h_i，按照它被放入$\Pi_v^{(k)}$的顺序进行编码。为了编码嵌入信息比特“0”和“1”，进一步将$\Pi_v^{(k)}$划分为两个子空间：$\Pi_0^{(k)}$和$\Pi_1^{(k)}$。

> $\Pi_1^{(k)}$：放入哈夫曼码字h_i序号为奇数，哈夫曼码字表示嵌入比特“1”
>
> $\Pi_0^{(k)}$：放入哈夫曼码字h_i序号为偶数，哈夫曼码字表示嵌入比特“0”

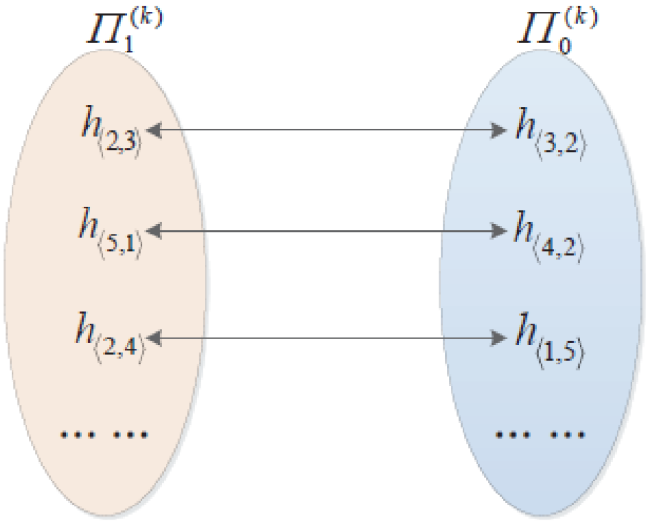
到此为止，就完成了码字映射表的构建，依据映射表就可以实现比特信息的嵌入，且每个$\Pi_v^{(k)}$中可隐写1比特信息码字。

举例 1

搜索码字空间划分为三个区域R_0, R_1和R_2，三个区域都是封闭的，每个区域独立进行搜索过程。下图描述了区域R_2的搜索过程，搜索的顺序按图中的箭头方向，实心点标注的哈夫曼码字分别是$h_{\langle 2,3\rangle}, h_{\langle 3,2\rangle}, h_{\langle 5,1\rangle}, h_{\langle 4,2\rangle}, h_{\langle 2,4\rangle}, h_{\langle 1,5\rangle}$，这些哈夫曼码字的长度都等于8。



按照搜索顺序，可以将这6个哈夫曼字依次组成3个可替换的哈夫曼码字对，记为$h_{\langle 2,3\rangle} \leftrightarrow h_{\langle 3,2\rangle}, h_{\langle 5,1\rangle} \leftrightarrow h_{\langle 4,2\rangle}, h_{\langle 2,4\rangle} \leftrightarrow h_{\langle 1,5\rangle}$。根据之前讨论过的划分规则，$h_{\langle 2,3\rangle}, h_{\langle 5,1\rangle}$和$h_{\langle 2,4\rangle}$被划分入$\Pi_1^{(k)}$；$h_{\langle 3,2\rangle}, h_{\langle 4,2\rangle}$和$h_{\langle 1,5\rangle}$被划分入$\Pi_0^{(k)}$。具体的哈夫曼码字替换关系图可见如下：



2.3 嵌入函数与提取函数

根据2.2中构造的映射表，码字到二进制映射的关系$f_{ctb}:\Pi_v^{(k)}\mapsto\{0,1\}$可解释为如下：

$$f_{ctb(h)}=\begin{cases}0, & h\in\Pi_0^{(k)}\\ 1, & h\in\Pi_1^{(k)}\end{cases}$$

上式子中h表示哈夫曼码字。

类似情况，逆映射二进制到码字的映射关系$f_{btc}:\{0,1\}\times\Pi_v^{(k)}\mapsto\Pi_v^{(k)}$可解释为如下：

$$f_{btc}(s,g)=\begin{cases}g, & s=0,g\in\Pi_0^{(k)}\\ \hat{g}, & s=0,g\in\Pi_1^{(k)}\\ \hat{g}, & s=1,g\in\Pi_0^{(k)}\\ g, & s=1,g\in\Pi_1^{(k)}\end{cases}$$

上式子中$(g,\hat{g})$是一对可以相互替换的哈夫曼码字对，s是对应比特流的当前状态值。而且，每个码表都能形成对应的映射表。

三. 码字符号位修改方法

选择不同的符号位置会影响到隐写算法的**安全性**和**负载率**的平衡，找到最合适的位置，就可以达到最优的安全负载临界点。**大值区**和**小值区**的系数都需要进行哈夫曼编码，并且所有的非零系数的符号位都是一个比特，当然零值区系数是不需要编码的。

在传输时，最大的隐藏负载率是**1比特/非零系数**。在做嵌入操作时，需要修改系数的符号位，也就是需要反转量化系数的值，比如：

$$1\leftrightarrow -1, 12\leftrightarrow -12$$

从以上例子可以看出，绝对值较大的系数，翻转其符号位会造成系数修改幅度很大，有可能会引起较为严重的**感知失真**。所以，为了避免对大值系数的修改，隐写算法可以设置**阈值T**，控制嵌入时可选用的修改系数范围。例如，当系数绝对值大于预设的阈值时，那么当前系数的符号位就不用来进行信息的嵌入。

以下举出四个不同阈值和码率下可用系数占比：

	$T=1$	$T=2$	$T=3$	$T=4$
128kbps	6.4%	14.6%	17.2%	18.4%
320kbps	7.8%	17.5%	20.5%	22.9%

根据待嵌入的消息比特m_j，可以用来判定是否需要反转系数q_j的符号位，设计的**嵌入规则**如下：

$$q'_j = \begin{cases} -|q_j|, & m_j = 1 \\ |q_j|, & m_j = 0 \end{cases}$$

四. Linbits位修改方法

根据哈夫曼码流结构和量化系数的分布特点，**linbits位域**比**系数符号位域**和**码字替换域**的可利用空间要小很多，从而导致linbits位嵌入的隐写负载率较低。为了提高隐写容量，可以在某些位置使用更大强度的嵌入方式，也就是采用多个**最低位嵌入**来代替LSB嵌入方式。

利用x_i代表第i个系数的linbits位嵌入前的数值； y_i代表第i个系数的linbits位嵌入后的数值；m代表待嵌入的信息值；k代表嵌入强度，k可以理解为使用的最低比特位。由此可得嵌入算法和提取算法如下：

4.1 嵌入算法

$$y_i = x_i - x_i \mod 2^k + m$$

4.2 提取算法

$$m = y_i \mod 2^k$$