**Problem Statement**

**OBJECTIVE**: Protect H.264/AVC video with fingerprinting against illegal distribution.

**PROPOSED SOLUTION**: Embed Tardos fingerprinting code in H.264/AVC using spread spectrum (SS) embedding while taking into account the reconstruction loop.

**OUR APPROACH**: The proposed solution consists of the following steps:

- Tardos fingerprinting code is generated offline for each user.
- Fingerprinting code is embedded into DC coefficients of all $4 \times 4$ blocks.
- Spread spectrum watermarking is used for embedding fingerprinting codes.
- Reconstruction loop is taken into account during the embedding process.

**H.264/AVC**

H.264/AVC [1] is the state of the art video codec and performs better than previous standards owing to many new tools including:

- 4X4 integer transform.
- Better entropy coding techniques:
  - CABAC
  - CAZLC
- Quarte pixel motion estimation.
- Multiple block size motion estimation
- Multiple reference frames.

**Tardos fingerprinting code generation**

A fingerprinting code is analyzed based on its code-length $m$, maximum number of colluders $c_0$, false-positive ($c_1$) and false-negative ($c_2$) values. For binary asymmetric Tardos Code, the length of code is given as $m = 100 c_0 \ln \left( \frac{1}{c_1} \right)$ and $c_1$ and $c_2$ is given as $c_2 = \epsilon^2$, as proposed by Tardos [2].

For code generation, we have the following three steps:

- For a code of length $m$, we generate random and independent probabilities ($p(i)$) under $m$, with the distribution $f(p) = \frac{1}{\sqrt{m+1}}$, for $p \in [0, 1]$. Practically $p$ is between $t$ and $1-t$ with $t = 10^{-3}$ and has high frequency on the edges:

\[
\text{f}(p) = \frac{1}{\sqrt{m+1}}
\]

- The next step is to generate Tardos code. For the case of binary Tardos code, each line of $S$ is filled with 0 or 1 with $\text{Prob}(S(i,j) = 1) = p(i)$. Each column is a fingerprinting code for separate user. For $n$ users with $m$ code-length, it is a matrix of size $m \times n$:

\[
\begin{array}{cccccc}
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 \\
\end{array}
\]

- For accusation process, a sequence $Z$ is extracted from the pirated copy and an accusation score $A_j$ is associated with user $j$ given as:

\[
A_j = \sum_{i=1}^{m} U(Z(i), S(i,j), p(i)),
\]

where

\[
U(1,1,p) = \sqrt{(1-p)/p} \quad U(1,0,p) = \sqrt{p/(1-p)},
\]
\[
U(0,0,p) = \sqrt{p/(1-p)} \quad U(0,1,p) = \sqrt{(1-p)/p}.
\]

$A_j$ for accused users may be modeled with a Gaussian centered at $\mu = \frac{2c_0}{c_0}$, while $A_j$ for innocent users may be modeled with a Gaussian centered at 0. Accused users (the traitors) have a score above $\mu - \sqrt{m} (\approx \frac{c_0}{2} - \sqrt{m})$, where $\sqrt{m}$ is the standard deviation of the Gaussian.

**Spread Spectrum Embedding in H.264/AVC**

- Reconstruction loop is taken into account for Tardos code embedding:

\[
x(i,j) = x(i,j) + \epsilon(i,j)
\]

- Integer transform

\[
f(v, u) = y(u, v)
\]

- Quantization

\[
x(i,j) = x(i,j) + \epsilon(i,j)
\]

- Inverse integer transform

\[
y_u(v, u) = y_u(v, u) + \epsilon_u(v, u)
\]

- Inverse quantization

\[
y_s(v, u) = y_s(v, u) + \epsilon_s(v, u)
\]

- Watermarking

\[
y_s(v, u) = y_s(v, u) + \epsilon_s(v, u)
\]

- Entropy coding

\[
x_u(v, u) = x_u(v, u) + \epsilon_u(v, u)
\]

- Entropy decoding

- Watermarked Bitstream

**Experimental Results**

- H.264/AVC JSVM 10.2 in AVC mode intra 4 x 4 MB mode along with CABAC.
- Parameters $n = 100$, $c_1 = 10^{-3}$, $c_0 = 20$, $m = 92104$. Payload is 10 bits/frame. Hence 9211 CIF frames (369 seconds of video at 25 fps) are used to embed the code.

**Visual Analysis**

For collider set $S$, the $j$th pixel of colluded video $x(z)$ is:

\[
x_w(u) = x_u(u), x_o(u) = x_o(u)
\]

with $S(i,j)$ is a message bit to be inserted with value 0 or 1.

**Conclusion**

- Colluders can be traced in a colluded video having acceptable visual quality.
- Tardos code with SS technique is an efficient video fingerprinting framework.
- The presented results can be improved by using informed watermarking and fine-tuning the parameters.

**References**