Implementasi Digital Signature pada Pengecekan Originalitas dari Film/Video

Muhammad Darrel Alwafin 18219109
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 18219109@std.stei.itb.ac.id

Implementasi tanda tangan digital dalam pengecekan originalitas suatu video merupakan sebuah metode yang digunakan untuk memverifikasi keaslian dan integritas konten video secara digital. Dalam konteks ini, tanda tangan digital adalah sebuah nilai yang dihasilkan menggunakan algoritma kriptografi pada data video yang bersifat unik dan tidak dapat dipalsukan. Pada implementasi ini, terdapat dua tahap utama. Tahap pertama adalah pembuatan pasangan kunci, yaitu kunci privat dan kunci publik. Kunci privat hanya diketahui oleh pemilik video, sedangkan kunci publik dapat didistribusikan ke pihak yang ingin memverifikasi video. Kunci publik digunakan untuk memverifikasi tanda tangan, sementara kunci privat digunakan untuk membuat tanda tangan. Tahap kedua adalah pembuatan tanda tangan digital. Proses ini melibatkan penghitungan hash dari konten video menggunakan algoritma hash yang kuat, seperti SHA-256. Hash ini kemudian dienkripsi menggunakan kunci privat menggunakan algoritma padding yang sesuai. Hasil enkripsi tersebut menjadi tanda tangan digital yang terkait dengan video tersebut.Pada tahap verifikasi, pihak yang ingin memverifikasi video akan menggunakan kunci publik untuk memverifikasi tanda tangan digital. Video yang ingin diverifikasi dihitung hash-nya menggunakan algoritma yang sama dengan yang digunakan pada tahap pembuatan tanda tangan. Selanjutnya, tanda tangan digital dan hash video tersebut dibandingkan. Jika tanda tangan cocok dengan hash video yang dihasilkan, maka video dianggap asli

.Keywords: tanda tangan digital, video, verifikasi, kunci privat, kunci publik, hash, SHA-256, originalitas.

I. INTRODUCTION (*HEADING 1*)

Video dan film telah menjadi bagian integral dari budaya populer dan hiburan modern. Namun, penyebaran video atau film yang tidak original atau bajakan telah menjadi permasalahan yang serius. Penyebaran video atau film bajakan melibatkan penggunaan dan distribusi karya tanpa izin dari pemilik hak cipta. Ini melanggar hak eksklusif yang dimiliki oleh pembuat karya tersebut dan melanggar undang-undang hak cipta yang ada. Penyebaran video atau film bajakan memiliki dampak ekonomi yang merugikan industri hiburan. Bajakan mengurangi pendapatan yang seharusnya diterima oleh pembuat film, rumah produksi, dan pihak terkait lainnya. Dalam beberapa kasus, penjualan dan pemasukan dari salinan bajakan dapat menghambat investasi di industri ini dan menyebabkan kerugian yang signifikan. Penyebaran video atau film bajakan berdampak negatif pada ekosistem industri hiburan secara keseluruhan. Kerugian finansial yang ditimbulkan dapat menghambat produksi karya baru, mengurangi insentif bagi para pembuat film, dan berpotensi menyebabkan kehilangan lapangan kerja di sektor hiburan. Ini mengancam keberlanjutan industri hiburan dan perkembangan kreativitas dalam pembuatan konten yang original. Mendapatkan video atau film bajakan dari sumber yang tidak terpercaya berpotensi menimbulkan risiko keamanan. Bajakan seringkali disertai dengan malware, virus, atau program berbahaya lainnya yang dapat merusak perangkat pengguna. Risiko ini meningkatkan kekhawatiran akan privasi dan keamanan data pribadi.

Dengan ini, kita dapat menggunakan tanda tangan digital (digital signature) untuk memverifikasi keaslian suatu video. Tanda tangan digital adalah teknik kriptografi yang menyediakan integritas dan otentikasi untuk data digital, termasuk video.

II. LANDASAN TEORI

A. Proses Digital Signature

Digital signature pada kriptografi memungkinkan verifikasi integritas dan otentikasi data digital melalui penggunaan kunci pribadi dan kunci publik. Prosesnya dimulai dengan pembuatan sepasang kunci kriptografi, di mana kunci pribadi hanya diketahui oleh pemiliknya dan kunci publik dapat diberikan kepada pihak lain.

Ketika seseorang ingin menandatangani pesan atau data, pesan tersebut diolah menggunakan fungsi hash untuk menghasilkan nilai hash yang unik. Nilai hash ini berfungsi sebagai "sidik jari" pesan, yang akan memastikan integritasnya. Selanjutnya, nilai hash tersebut dienkripsi menggunakan kunci pribadi pemilik. Proses enkripsi ini menghasilkan tanda tangan digital yang unik untuk pesan atau data tersebut.

Tanda tangan digital ini kemudian dapat disertakan bersama dengan pesan atau data saat dikirimkan kepada penerima. Untuk memverifikasi tanda tangan, penerima mengolah pesan atau data yang diterima menggunakan fungsi hash yang sama seperti yang digunakan oleh pengirim. Hasil dari pengolahan ini harus menghasilkan nilai hash yang sama dengan nilai hash yang diperoleh dari tanda tangan digital yang diterima. Jika kedua nilai hash tersebut cocok, maka tanda tangan dianggap valid.

Penting untuk dicatat bahwa verifikasi tanda tangan digital ini dilakukan menggunakan kunci publik yang terkait dengan kunci pribadi pengirim. Kunci publik dapat digunakan oleh siapa pun untuk memverifikasi tanda tangan, tetapi tidak dapat digunakan untuk membuat tanda tangan baru. Hal ini menjaga keamanan dan keotentikan tanda tangan digital.

Dengan menggunakan proses digital signature ini, penerima pesan atau data dapat memastikan bahwa pesan atau data tersebut tidak diubah sejak tanda tangan dibuat dan juga dapat mengidentifikasi pihak yang bertanggung jawab atas pesan atau data tersebut. Secara keseluruhan, digital signature pada kriptografi memberikan mekanisme yang aman dan terpercaya untuk memverifikasi integritas dan otentikasi data digital.

B. Digital Signature pada Video

Digital signature pada kriptografi dapat digunakan untuk memverifikasi originalitas suatu video dengan memastikan integritas dan otentikasi data video tersebut. Prosesnya dimulai dengan menghasilkan tanda tangan digital untuk video menggunakan kunci pribadi yang hanya diketahui oleh pihak yang berwenang, seperti pembuat atau pemilik video. Tanda tangan digital ini dibuat dengan mengolah video menggunakan fungsi hash yang menghasilkan nilai hash yang unik untuk video tersebut.

Kemudian, tanda tangan digital tersebut dienkripsi menggunakan kunci pribadi untuk menghasilkan tanda tangan digital yang unik. Tanda tangan digital ini terkait erat dengan video asli dan tidak dapat dengan mudah dipalsukan atau diubah tanpa pengetahuan kunci pribadi yang sesuai.

Untuk memverifikasi originalitas video, penerima video menggunakan kunci publik yang terkait dengan kunci pribadi pengirim. Penerima mengolah video yang diterima menggunakan fungsi hash yang sama dengan yang digunakan oleh pengirim. Hasil dari pengolahan ini harus cocok dengan nilai hash yang dihasilkan dari tanda tangan digital yang diterima.

Jika nilai hash video yang diolah cocok dengan nilai hash yang dihasilkan dari tanda tangan digital, maka tanda tangan dianggap valid dan video dianggap original. Hal ini menunjukkan bahwa video tidak mengalami perubahan sejak tanda tangan dibuat, dan juga memverifikasi bahwa tanda tangan tersebut berasal dari pihak yang berwenang.

Dengan menggunakan digital signature pada kriptografi, verifikasi originalitas video dapat dilakukan secara efektif. Hal ini memberikan keyakinan bahwa video tersebut tidak mengalami perubahan yang tidak sah dan dapat dipercaya sebagai salinan asli. Proses ini penting dalam melindungi hak cipta dan memastikan bahwa video yang ditonton adalah versi yang otentik dan sah.

III. SKEMA RANCANGAN

berikut adalah skema rancangan umum dalam penggunaan tanda tangan digital untuk memverifikasi keaslian suatu video:

A. Proses persiapan:
   a. Generate pasangan kunci: Buat kunci privat (private key) dan kunci publik (public key) menggunakan algoritma kriptografi seperti RSA.
   b. Simpan kunci privat: Simpan kunci privat yang dihasilkan dalam format yang sesuai, seperti PEM atau DER.
   c. Simpan kunci publik: Simpan kunci publik yang dihasilkan dalam format yang sesuai, seperti PEM atau DER.

B. Proses penandatanganan:
   a. Baca video: Baca video yang akan ditandatangani sebagai file biner.
   b. Hitung hash: Hitung nilai hash (hash value) dari konten video menggunakan fungsi hash, seperti SHA-256. Ini akan menghasilkan nilai hash yang merupakan representasi unik dari konten video.
   c. Muat kunci privat: Muat kunci privat yang telah disimpan sebelumnya.
   d. Tandatangani hash: Tandatangani nilai hash menggunakan kunci privat yang dimuat. Ini akan menghasilkan tanda tangan digital untuk video.
   e. Simpan tanda tangan: Simpan tanda tangan digital yang dihasilkan sebagai bagian dari video atau dalam file terpisah.

C. Proses verifikasi:
   a. Baca video: Baca video yang akan diverifikasi sebagai file biner.
   b. Hitung hash: Hitung nilai hash dari konten video menggunakan fungsi hash yang sama yang digunakan saat penandatanganan.
   c. Muat kunci publik: Muat kunci publik yang telah disimpan sebelumnya.
   d. Muat tanda tangan: Muat tanda tangan digital yang terkait dengan video yang akan diverifikasi.
   e. Verifikasi tanda tangan: Gunakan kunci publik untuk memverifikasi tanda tangan terhadap nilai hash yang dihitung. Jika tanda tangan valid, video dianggap autentik dan tidak mengalami perubahan.

D. Hasil verifikasi:
   a. Jika tanda tangan valid, video dianggap autentik dan tidak mengalami perubahan.
   b. Jika tanda tangan tidak valid, video dianggap tidak autentik atau telah mengalami perubahan.

Implementasi untuk membangkitkan pasangan kunci, menghasilkan tanda tangan digital untuk video menggunakan kunci privat, dan memverifikasi tanda tangan menggunakan kunci publik yang sesuai. Berikut adalah penjelasan langkah demi langkah untuk kode tersebut:

1. Fungsi generate_key_pair() dan generate_key_pair2() digunakan untuk menghasilkan pasangan kunci (kunci privat dan kunci publik) untuk setiap video.

Kunci privat digunakan untuk menghasilkan tanda tangan digital, sementara kunci publik digunakan untuk memverifikasi tanda tangan.

2. Fungsi generate_signature(video_file, private_key_file) digunakan untuk menghasilkan tanda tangan digital untuk video tertentu. Pada langkah-langkahnya:
    a. Video dibaca dan kontennya diambil.
    b. Dilakukan penghitungan hash SHA-256 dari konten video.
    c. Kunci privat dibaca dari file yang ditentukan.
    d. Tanda tangan digital dihasilkan dengan menggunakan kunci privat, padding PKCS1v15, dan fungsi hash SHA-256.
    e. Tanda tangan digital dikembalikan sebagai output.

3. Fungsi verify_signature(video_file, signature, public_key_file) digunakan untuk memverifikasi tanda tangan digital dari video tertentu. Pada langkah-langkahnya:
    a. Video dibaca dan kontennya diambil.
    b. Dilakukan penghitungan hash SHA-256 dari konten video.
    c. Kunci publik dibaca dari file yang ditentukan.
    d. Tanda tangan digital diverifikasi menggunakan kunci publik, padding PKCS1v15, dan fungsi hash SHA-256.
    e. Jika verifikasi berhasil, fungsi mengembalikan nilai True, jika gagal, fungsi mengembalikan nilai False.

4. Pada bagian utama kode:
    a. Fungsi generate_key_pair() dipanggil untuk menghasilkan pasangan kunci untuk video pertama, dan generate_key_pair2() dipanggil untuk menghasilkan pasangan kunci untuk video kedua.
    b. Kemudian, tanda tangan digital dihasilkan untuk masing-masing video menggunakan kunci privat yang sesuai.
    c. Terakhir, dilakukan verifikasi tanda tangan untuk video pertama menggunakan kunci publik video kedua, dan verifikasi tanda tangan untuk video kedua menggunakan kunci publik video pertama.
    d. Jika kedua verifikasi berhasil, maka dianggap bahwa video memiliki konten yang sama. Jika salah satu verifikasi gagal, maka dianggap bahwa video memiliki konten yang berbeda atau telah dimanipulasi.

Harap diperhatikan bahwa pada kode tersebut, Anda menggunakan lokasi file video yang sama untuk video_file1 dan video_file2. Dalam kasus praktis, Anda perlu memastikan bahwa Anda memilih video yang berbeda untuk membandingkan keaslian dan tidak menggunakan video yang sama untuk kedua verifikasi.

IV. Analisa dan Pembahasan

Untuk melakukan analisa dan pembahasan, perlu ditampilkan gambar kode dan penjelasannya secara satu persatu. Kemudian, kode secara keseluruhannya akan dijelaskan lebih lanjut output dan hasil yang didapatkan dari kode tersebut.

```
import hashlib
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
```

Gambar IV.1 Import Library

Import terlebih dahulu semua library yang dibutuhkan. “import hashlib” dan “from cryptography.hazmat.primitives import hashes” adalah untuk mengimpor modul hashlib dan hashes dari library cryptography. Modul ini digunakan untuk menghasilkan hash dari konten video menggunakan algoritma hash SHA-256.

“from cryptography.hazmat.primitives.asymmetric import rsa, padding” adalah untuk mengimpor modul rsa dan padding dari library cryptography. Modul ini digunakan untuk menghasilkan pasangan kunci RSA, menghasilkan tanda tangan digital, dan melakukan verifikasi tanda tangan.

“from cryptography.hazmat.primitives import serialization” adalah untuk mengimpor modul serialization dari library cryptography. Modul ini digunakan untuk membaca dan menulis kunci publik dan privat dalam format yang tepat.

```
def generate_key_pair():
    private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
    public_key = private_key.public_key()

    private_key_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_key_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    with open('private_key.pem', 'wb') as private_key_file:
        private_key_file.write(private_key_pem)
    with open('public_key.pem', 'wb') as public_key_file:
        public_key_file.write(public_key_pem)

def generate_key_pair2():
    private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
    public_key = private_key.public_key()

    private_key_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_key_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
```

Gambar IV.2 Fungsi generate_key_pair()

Bagian kode yang ini adalah fungsi generate_key_pair() yang digunakan untuk menghasilkan sepasang kunci RSA dan menyimpannya dalam format PEM.

“private_key=rsa.generate_private_key(public_exponent=65537, key_size=2048)”: Baris ini menghasilkan sebuah kunci privat RSA dengan panjang 2048 bit menggunakan eksponen publik yang diberikan (65537).

“public_key = private_key.public_key()”: Baris ini mengambil kunci publik yang sesuai dengan kunci privat yang dihasilkan sebelumnya.

“private_key_pem=private_key.private_bytes(encoding=serialization.Encoding.PEM,format=serialization.PrivateFormat.PKCS8,encryption_algorithm=serialization.NoEncryption())”: Baris ini mengonversi kunci privat ke format PEM dengan menggunakan encoding PEM, format PKCS8, dan tanpa enkripsi (NoEncryption).

“public_key_pem=public_key.public_bytes(encoding=serialization.Encoding.PEM,format=serialization.PublicFormat.SubjectPublicKeyInfo)”: Baris ini mengonversi kunci publik ke format PEM dengan menggunakan encoding PEM dan format SubjectPublicKeyInfo.

“with open('private_key.pem', 'wb') as private_key_file: private_key_file.write(private_key_pem)”: Baris ini membuka file dengan nama 'private_key.pem' dalam mode write binary ('wb') dan menulis kunci privat dalam format PEM ke file tersebut.

“with open('public_key.pem', 'wb') as public_key_file: public_key_file.write(public_key_pem)”: Baris ini membuka file dengan nama 'public_key.pem' dalam mode write binary ('wb') dan menulis kunci publik dalam format PEM ke file tersebut.

Dengan menjalankan fungsi “generate_key_pair()”, akan dihasilkan sepasang kunci privat dan publik RSA, dan keduanya akan disimpan sebagai file 'private_key.pem' dan 'public_key.pem'.

```python
def generate_signature(video_file, private_key_file):
    with open(video_file, 'rb') as file:
        video_content = file.read()

    video_hash = hashlib.sha256(video_content).digest()

    with open(private_key_file, 'rb') as key_file:
        private_key = serialization.load_pem_private_key(key_file.read(), password=None)

    signature = private_key.sign(video_hash, padding.PKCS1v15(), hashes.SHA256())

    return signature
```

Gambar IV.3 Fungsi generate_signature()

Kode tersebut merupakan implementasi dari fungsi “generate_signature(video_file, private_key_file)” yang digunakan untuk menghasilkan tanda tangan digital dari konten video menggunakan kunci privat RSA yang diberikan.

Membuka file video menggunakan “open(video_file, 'rb')” dengan mode baca binary ('rb') dan menyimpan isinya ke dalam variabel “video_content”.

Menghitung hash SHA-256 dari konten video menggunakan “hashlib.sha256(video_content).digest()” dan menyimpan hasilnya ke dalam variabel “video_hash”. Hash digunakan untuk merepresentasikan "sidik jari" unik dari konten video.

Membuka file kunci privat menggunakan “open(private_key_file, 'rb')” dengan mode baca binary ('rb') dan membaca isi file tersebut.

Menggunakan modul “serialization” dari library Cryptography untuk memuat kunci privat RSA dari pem-format yang dibaca sebelumnya. Fungsi load_pem_private_key() digunakan untuk memuat kunci privat RSA dari file dalam format PEM. Argumen key_file.read() adalah isi file kunci privat dalam bentuk string.

Menggunakan kunci privat yang telah dimuat, fungsi “sign()” digunakan untuk menghasilkan tanda tangan digital dari hash konten video. Fungsi ini menerima beberapa argumen, yaitu “video_hash” sebagai data yang akan ditandatangani, “padding.PKCS1v15()” untuk jenis padding yang digunakan dalam penandatanganan, dan “hashes.SHA256()” untuk jenis fungsi hash yang digunakan.

Tanda tangan digital yang dihasilkan disimpan dalam variabel “signature”. Fungsi mengembalikan nilai “signature”, yaitu tanda tangan digital dari konten video menggunakan kunci privat RSA yang diberikan.

```python
def verify_signature(video_file, signature, public_key_file):
    with open(video_file, 'rb') as file:
        video_content = file.read()

    video_hash = hashlib.sha256(video_content).digest()

    with open(public_key_file, 'rb') as key_file:
        public_key = serialization.load_pem_public_key(key_file.read())

    try:
        public_key.verify(signature, video_hash, padding.PKCS1v15(), hashes.SHA256())
        return True  # Signature verification successful
    except (ValueError, TypeError):
        return False  # Signature verification failed
```

Gambar IV.4 Fungsi verify_signature()

Kode tersebut merupakan implementasi dari fungsi “verify_signature(video_file, signature, public_key_file)” yang digunakan untuk memverifikasi tanda tangan digital pada konten video menggunakan kunci publik RSA yang diberikan.

Membuka file video menggunakan “open(video_file, ‘rb’)” dengan mode baca binary (‘rb’) dan menyimpan isinya ke dalam public t “video_content”.

Menghitung hash SHA-256 dari konten video menggunakan hashlib.sha256(video_content).digest() dan menyimpan hasilnya ke dalam video_hash. Ini dilakukan untuk menghasilkan hash yang akan digunakan dalam verifikasi.

Membuka file kunci public menggunakan “open(public_key_file, ‘rb’)” dengan mode baca binary (‘rb’) dan membaca isi file tersebut.

Menggunakan modul “serialization” dari library Cryptography untuk memuat kunci public RSA dari pem-format yang dibaca sebelumnya. Fungsi “load_pem_public_key()” digunakan untuk memuat kunci public RSA dari file dalam format PEM. Argumen “key_file.read()” adalah isi file kunci public dalam bentuk string.

Setelah kunci public berhasil dimuat, fungsi “verify()” digunakan untuk memverifikasi tanda tangan digital. Fungsi ini menerima beberapa argumen, yaitu “signature” yang merupakan tanda tangan digital yang akan diverifikasi, “video_hash” yang merupakan hash dari konten video yang akan dibandingkan, “padding.PKCS1v15()” untuk jenis padding yang digunakan dalam verifikasi, dan “hashes.SHA256()” untuk jenis fungsi hash yang digunakan.

Jika verifikasi tanda tangan berhasil, yaitu tanda tangan digital cocok dengan konten video dan kunci public yang digunakan, fungsi mengembalikan nilai “True” yang menunjukkan bahwa verifikasi berhasil.

Jika terjadi kesalahan atau verifikasi tanda tangan gagal, seperti tanda tangan tidak cocok atau terdapat kesalahan dalam argumen, fungsi mengembalikan nilai “False” yang menunjukkan bahwa verifikasi gagal.

Dengan menggunakan fungsi verify_signature(), program dapat memverifikasi keaslian konten video dengan membandingkan tanda tangan digital yang dihasilkan oleh fungsi generate_signature() dengan kunci public yang digunakan dalam verifikasi.

```
# Generate key pairs for each video
generate_key_pair()

# Video, Private Key, and Public Key allocation
video_file1 = 'C:\\ITB\Semester 8\Kriptografi\CS.mp4'
video_file2 = 'C:\\ITB\Semester 8\Kriptografi\CS.mp4'
private_key_file1 = 'private_key.pem'
private_key_file2 = 'private_key2.pem'
public_key_file1 = 'public_key.pem'
public_key_file2 = 'public_key2.pem'
```

Gambar IV.5 Alokasi Video, Private Key, dan Public Key

Kode bagian ini terdiri dari beberapa bagian terpisah yang menjelaskan bagaimana menghasilkan pasangan kunci (private-public key) untuk setiap video dan menunjukkan contoh penggunaan kunci-kunci tersebut.

Fungsi “generate_key_pair()” digunakan untuk menghasilkan pasangan kunci (private-public key). Di dalamnya, kunci privat (private key) dan kunci public (public key) dihasilkan menggunakan metode “rsa.generate_private_key()” dengan panjang kunci sebesar 2048 bit dan public exponent sebesar 65537. Kunci public kemudian diperoleh dari kunci privat dengan memanggil metode “private_key.public_key()”.

Setelah kunci-kunci dihasilkan, mereka disimpan dalam format PEM yang merupakan format standar yang digunakan untuk menyimpan kunci-kunci dalam bentuk teks. Kunci privat disimpan dalam file “private_key.pem” dan kunci public disimpan dalam file “public_key.pem” menggunakan metode “private_key.private_bytes()” dan “public_key.public_bytes()”.

Di sini juga terdapat dua video file yang sama yaitu “CS.mp4”. Kedua video ini akan digunakan untuk proses verifikasi.

```
# Generate and save the signature for video 1
signature1 = generate_signature(video_file1, private_key_file1)

# Generate and save the signature for video 2
signature2 = generate_signature(video_file2, private_key_file2)
```

Gambar IV.6 *Assign Signature*

Pada kode bagian ini, terdiri dari pemanggilan fungsi “generate_signature()”. Parameter dari fungsi ini terdiri dari “video_file(1 & 2)” dan “private_key_file(1 & 2)”. Fungsi ini digunakan untuk menghasilkan digital signature dari video tersebut.

```
# Verify the signatures of the videos
is_same = verify_signature(video_file1, signature2, public_key_file2) and verify_signature(video_file2, signature1, public_key_file1)

if is_same:
    print("Videos have the same content.")
else:
    print("Videos have different content or have been tampered with.")
```

Gambar IV.7 Verifikasi Kedua Video

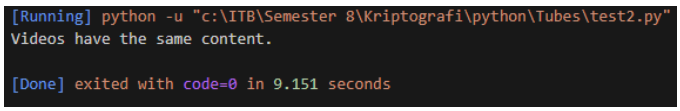
Kode bagian ini bertujuan untuk memverifikasi tanda tangan digital dari dua video.

Fungsi verify_signature dipanggil dua kali untuk melakukan verifikasi tanda tangan digital dari kedua video. Pada verifikasi pertama, fungsi “verify_signature(video_file1, signature2, public_key_file2)” dipanggil dengan parameter “video_file1” yang berisi path file video pertama, “signature2” yang berisi tanda tangan digital dari video kedua, dan “public_key_file2” yang berisi path file kunci publik yang akan digunakan untuk verifikasi. Fungsi “verify_signature()” akan mengembalikan nilai “True” jika verifikasi tanda tangan sukses, dan “False” jika verifikasi gagal.

Pada verifikasi kedua, fungsi “verify_signature(video_file2, signature1, public_key_file1)” dipanggil dengan argumen “video_file2” yang berisi path file video kedua, “signature1” yang berisi tanda tangan digital dari video pertama, dan “public_key_file1” yang berisi path file kunci publik yang akan digunakan untuk verifikasi. Fungsi verify_signature juga akan mengembalikan nilai “True” jika verifikasi tanda tangan sukses, dan “False” jika verifikasi gagal.

Variabel is_same akan berisi hasil dari kondisi “verify_signature(video_file1, signature2, public_key_file2)” and “verify_signature(video_file2, signature1, public_key_file1)”, yang mengecek apakah kedua video memiliki tanda tangan digital yang valid. Jika kedua verifikasi sukses, maka nilai is_same akan menjadi “True”, yang berarti kedua video memiliki konten yang sama. Jika salah satu atau kedua verifikasi gagal, nilai is_same akan menjadi “False”, yang berarti kedua video memiliki konten yang berbeda atau telah diubah dengan tidak sah.

Akhirnya, pesan "Videos have the same content." akan dicetak jika nilai is_same adalah True, yang berarti kedua video memiliki konten yang sama. Jika nilai is_same adalah False, pesan "Videos have different content or have been tampered with." akan dicetak, yang berarti kedua video memiliki konten yang berbeda atau telah diubah.



Gambar IV.8 Output dari Kode

V. HASIL DAN KESIMPULAN

Jika suatu video dibajak secara ilegal, konten video dapat berubah atau dimodifikasi tanpa izin dari pemilik aslinya. Namun, dengan menggunakan tanda tangan digital seperti yang diimplementasikan dalam kode yang diberikan, kita dapat mendeteksi perubahan atau modifikasi pada video tersebut.

Tanda tangan digital yang dihasilkan dari video asli akan berbeda jika ada perubahan pada konten video. Jadi, dengan memverifikasi tanda tangan digital yang terkait dengan video, kita dapat menentukan apakah video tersebut telah mengalami perubahan atau tidak utuh.

Namun, perlu diingat bahwa tanda tangan digital hanya memberikan mekanisme untuk mendeteksi perubahan pada video. Upaya untuk mencegah atau mengatasi pembajakan video ilegal melibatkan langkah-langkah hukum yang lebih luas, seperti perlindungan hak cipta, pemantauan dan penegakan hukum, serta tindakan lain yang sesuai dengan kebijakan dan peraturan yang berlaku. Dalam situasi hukum atau perselisihan yang melibatkan video atau film, verifikasi keaslian dapat memberikan bukti yang kuat untuk mendukung klaim atau tanggapan. Hal ini dapat membantu memastikan bahwa bukti tersebut tidak dipalsukan atau dimanipulasi.

Verifikasi keaslian video atau film dapat membantu dalam melindungi hak cipta dan kepemilikan intelektual. Dengan memastikan bahwa konten tersebut tidak diubah secara ilegal, kita dapat memperkuat perlindungan hak-hak pemilik asli dan mencegah pembajakan atau penggunaan yang tidak sah.

Dalam keseluruhan, dengan kemampuan untuk memverifikasi keaslian video atau film, kita dapat memperkuat keamanan, melindungi hak cipta, memberikan bukti yang kuat, dan membangun kepercayaan publik terhadap konten tersebut. Namun, perlu dicatat bahwa verifikasi keaslian hanya satu aspek dari upaya yang lebih luas dalam menjaga integritas dan keamanan konten digital.

LINKS

Video penjelasan singkat:

https://youtu.be/VGzWuLdDfIA

REFERENCES

[1] Sajedi, Ali. An Introduction to Digital Signature Schemes

[2] Sanjaya, I Dewa Gede Adi. Proses Lahirnya Hak Cipta Terhadap Pembuatan Video Klip Berdasarkan Undang-Undang Nomor 19 Tahun 2002.

[3] Barbarosa, Diego Ibrahim. Peran Kemkominfo Terkait Pembajakan Film pada Situs Streaming Film Ilegal.

Bandung, 22 Mei 2023

Muhammad Darrel Alwafin
18219109