Secure Donation-Based Video Submission System Using SHA-256 and Digital Signature for Media Share Platforms

Muhammad Rafly Fauzan - 18223132[1, 2, 3]
Department of Information System and Technology
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]18223132@std.stei.itb.ac.id, [2]mraflyfauzan12@gmail.com, [3]clazzi.hzkz@gmail.com,

***Abstract*—This paper proposes a secure donation-based video submission system for Media Share platforms using SHA-256 and digital signature. In donation-based livestreaming, viewers may submit video links along with donations, but the submitted data can be vulnerable to tampering, fake requests, and replay attacks. The proposed system generates a SHA-256 hash from donation and video request metadata, then signs the hash using a private key to provide integrity and authenticity. A prototype system is implemented to simulate video submission, queue management, signing, and verification processes. Experimental evaluation is conducted by modifying video URLs, donation amounts, timestamps, and queue positions to test whether the system can detect unauthorized changes. The results are expected to show that the proposed mechanism can effectively identify tampered requests while maintaining acceptable processing performance.**

Keywords—SHA-256, digital signature, media share, data integrity, authentication, replay attack.

I. Introduction

The growth of livestreaming platforms has created new forms of interaction between streamers and viewers. One popular interaction model is donation-based video submission, where viewers can donate money and submit a video link to be played during a livestream. This feature is commonly found in media share systems, donation platforms, and livestreaming environments. It allows viewers to participate more actively in the stream while also helping streamers receive financial support.

However, this system also introduces several security risks. A video submission usually contains important data, such as the donor's name, donation amount, video URL, submission time, payment status, and queue position. If this data is not properly protected, it can be modified or misused by unauthorized parties. For example, an attacker may change the submitted video URL, manipulate the donation amount, create a fake video request, modify the queue order, or resend an old request to make the same video appear again. These problems can harm the streamer, the donors, and the platform itself. The main security issue in a donation-based video submission system is how to ensure that every video request is authentic, has not been modified, and is linked to a valid donation transaction. To solve this problem, a cryptographic mechanism is needed to protect the integrity and authenticity of the submitted data.

SHA-256 is a cryptographic hash function that can generate a unique hash value from input data. In this system, SHA-256 is used to create a digital fingerprint of the video request data, including the donation ID, donor name, donation amount, video URL, timestamp, nonce, and queue position. If any part of the data is changed, the generated hash value will also change. Therefore, SHA-256 can help the system detect data tampering.

Digital signature is used to verify the authenticity of the video request. After the system generates a SHA-256 hash, the hash is signed using a private key. Later, the signature can be verified using the corresponding public key. If the verification is successful, the system can confirm that the video request is valid and has not been modified. If the verification fails, the request can be rejected because it may have been altered or forged.

This paper proposes a secure donation-based video submission system using SHA-256 and digital signature for media share platforms. The proposed system is not limited to theoretical discussion, but also includes system design, prototype implementation, and experimental testing. The experiments are conducted by simulating several data manipulation scenarios, such as changing the video URL, donation amount, timestamp, and queue position. The purpose of these experiments is to evaluate whether the proposed system can detect unauthorized changes in video request data. The main contribution of this paper is the design and evaluation of a cryptographic verification mechanism for donation-based video submission. By using SHA-256 and digital signature, the proposed system can improve data integrity, authenticity, and trust in media share platforms. This research is expected to provide a practical security approach for protecting video requests in livestreaming donation systems.

II. Theoretical Background

A. Donation-Based Video Submission System

A donation-based video submission system or known as *Media Share* is a feature that allows viewers to submit video links to be played during a livestream after sending a donation or tip. In media share platforms, viewers can request videos or songs, and the submitted media can be added to a queue before being displayed on the stream [1].

This mechanism increases audience participation because viewers are not only watching the stream but also contributing content to the livestream session. In a typical media share process, a video request may contain several important data fields, such as donation ID, donor name, donation amount, video URL, timestamp, payment status, and queue position. Some platforms also allow the submitted media to be moderated before being played, so that the streamer or moderator can control which content appears on the stream [1].

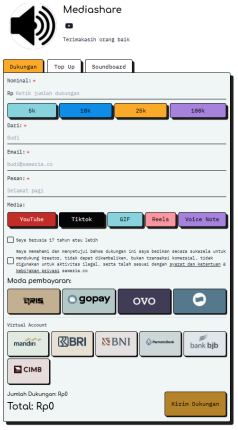

Fig 1. Input of An Example of Media Share *Saweria*

Fig. 2 Output of An Example of Media Share *Saweria*

B. SHA–256 Hash Function

SHA-256 is part of the Secure Hash Standard defined by NIST. A cryptographic hash function generates a fixed-size message digest from input data, and the digest can be used to detect whether the original data has changed [3]. In this research, SHA-256 is used to generate a hash value from donation-based video request data.

The data that can be hashed includes donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position. The resulting hash value acts as a digital fingerprint of the request. If any part of the request is modified, such as changing the video URL or donation amount, the generated hash value will also change. Therefore, SHA-256 can help the system detect data tampering.

C. Digital Signature

A digital signature is a cryptographic mechanism used to detect unauthorized modification of data and to authenticate the identity of the signer [4]. In this research, a digital signature is used to verify that a video request was created or approved by the legitimate system. The signing process begins by generating a SHA-256 hash from the video request data. Then, the hash is signed using the system's private key. During verification, the system uses the corresponding public key to check whether the signature is valid. If the signature is valid, the request is considered authentic and unchanged. If the signature is invalid, the request may have been modified or forged.

Digital signatures are suitable for this system because they support two important security goals: integrity and authenticity. They can also support non-repudiation, because a valid signature can provide evidence that the signed data was generated using the claimed private key [4].

D. Replay Attack Prevention Using Timestamp and Nonce

A replay attack occurs when a valid data transmission is maliciously or fraudulently repeated by an attacker [6]. In a donation-based video submission system, this can happen when an old valid video request is resent to the system so that the same video appears again without a new donation. To reduce this risk, the proposed system includes a timestamp and nonce in each request.

A nonce is a value that is used only once or has a very small chance of repeating, such as a random value, timestamp, sequence number, or a combination of these values [7]. OWASP also recommends using unique nonces and validating timestamps to prevent message replay vulnerabilities [8]. In this system, the timestamp is used to check whether the request is still within an acceptable time window. The nonce is used to ensure that the same request cannot be reused. If the nonce has already been used or the timestamp is expired, the request should be rejected.

III. System Design

A. System Overview

The system is designed to secure donation-based video submissions in Media Share platforms by applying SHA-256 and digital signature mechanisms. The main objective of the system is to ensure that each video request is valid, authentic, and has not been modified before it is added to the playback queue. In this system, a donor submits a donation along with a video URL. After the donation is verified, the system collects important request data, such as donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position.

The request data is then converted into a consistent format and processed using SHA-256 to generate a hash value. This hash value acts as a digital fingerprint of the request. After that, the hash is signed using the system's private key to

produce a digital signature. The original request data, hash value, and digital signature are stored in the database.

Before a video request is added to the queue or played during the livestream, the system performs a verification process. The verification process includes recalculating the hash value, comparing it with the stored hash, verifying the digital signature using the public key, checking timestamp validity, and ensuring that the nonce has not been used before. If all verification steps are successful, the request is considered valid and the video is added to the playback queue. After the video is played, the request status is marked as completed. However, if the verification fails, the system rejects the request, records the failure reason in the log, marks the request as invalid, and returns to the request monitoring process to wait for the next submission. The general workflow of the proposed system is shown in following Fig. 3.

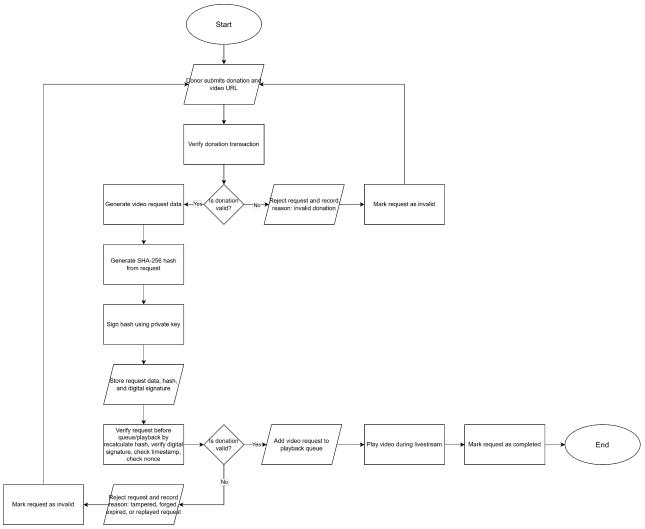

Fig. 3 Flowchart of the secure donation-based video submission system.

This flowchart shows that the system does not only accept or reject a video request, but also performs request status management. A valid request will be added to the playback queue, played during the livestream, and marked as completed. Meanwhile, an invalid request will be rejected, logged, marked as invalid, and the system will return to the request monitoring process. This loop allows the system to continuously handle new donation-based video submissions while maintaining the integrity and authenticity of each request.

B. Video Request Data Structure

The proposed system uses structured request data to ensure that all important information related to a donation-based video submission can be protected. Each video request contains several fields, including donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position. These fields are selected because they directly affect the validity and processing of the video request. For example, the donation amount may influence the priority of the request, while the video URL determines the content that will be played during the livestream. The timestamp and nonce are included to reduce the risk of replay attacks.

The timestamp records the time when the request is created, while the nonce acts as a unique value that should only be used once. If an attacker attempts to reuse an old request, the system can detect it by checking whether the nonce has already been used or whether the timestamp has expired. Therefore, the request data structure is not only used to store submission information, but also to support the security verification process. An example of the video request data structure is shown below.

```
{
   "donation_id": "MS12345",
   "donor_name": "RezaA",
   "amount": 10000,
   "Video_url": "https://youtube.com/watch?v=example",
   "timestamp": "2026-06-19T20:00:00",
   "nonce": "N123456",
   "payment_status": "verified",
   "queue_position": 5
 }
```

Before the data is processed using SHA-256, it must be converted into a consistent format. This step is important because different data formats or field orders may produce different hash values even if the meaning of the data is the same. Therefore, the system uses a fixed field order when generating the hash.

C. Hashing and Signing Process

The hashing and signing process begins after the donation transaction has been verified by the system. At this stage, the system collects the important data from the video request, including the donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position. These data fields are arranged into a consistent format so that the same request will always produce the same hash value. This formatting process is important because even a different field order or small formatting difference can produce a different SHA-256 hash.

After the request data is prepared, the system applies the SHA-256 hash function to generate a fixed-size hash value. This hash value acts as a digital fingerprint of the video request. Any change to the original request data, such as modifying the video URL, changing the donation amount, or altering the queue position, will produce a different hash value. Therefore, the generated hash can be used to detect whether the request data has been modified after it was created.

Once the hash value has been generated, the system signs the hash using the private key. The result of this process is a digital signature that is linked to the original request data. The private key is only owned by the trusted system, so only the legitimate system should be able to generate a valid signature. After the signing process is complete, the original request data, hash value, and digital signature are stored in the database. This stored information will later be used in the verification process before the video request is accepted into the playback queue.

D. Hashing and Signing Process

The verification process is performed before a video request is added to the playback queue or played during the livestream. The purpose of this process is to ensure that the request is still valid, has not been modified, and was generated by the legitimate system. To perform verification, the system retrieves the stored video request data, hash value, and digital signature from the database.

The system then recalculates the SHA-256 hash from the current request data using the same consistent format used during the signing process. The newly generated hash is compared with the stored hash value. If the two hash values are different, it means that the request data has been changed after it was signed. In this case, the system rejects the request because it may have been tampered with.

If the hash values match, the system continues by verifying the digital signature using the corresponding public key. A valid signature indicates that the request was signed using the trusted private key and that the request has not been forged. However, if the signature verification fails, the system rejects the request because it may have been created by an unauthorized party or modified after signing.

In addition to hash and signature verification, the system also checks the timestamp and nonce. The timestamp is used to ensure that the request is still within the allowed time limit, while the nonce is used to make sure that the same request has not been used before. If the timestamp is expired or the nonce has already been recorded in the system, the request is rejected as a possible replay attack. When all verification checks are successful, the request is considered valid and can be added to the playback queue. After the video is played, the request status is marked as completed.

IV. Result and Analysis

A. Program Overview

The prototype was implemented to evaluate the proposed secure donation-based video submission system. The program simulates the main security workflow, starting from receiving video request data, validating the request structure, converting the data into a canonical format, generating a SHA-256 hash, signing the hash using a digital signature, and verifying the request before it is accepted into the playback queue. The implemented request structure contains several fields, including donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position. These fields represent the important metadata in a donation-based video submission system. The program allows each field to be modified for testing purposes, so the effect of data manipulation can be observed clearly during the verification process.

Fig 4. Program execution for the signing and verification process

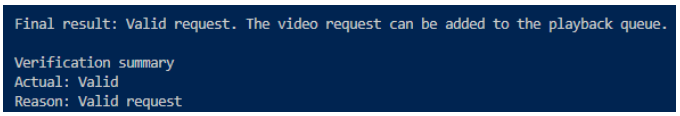

Fig. 5 Example of a valid request that successfully passes verification

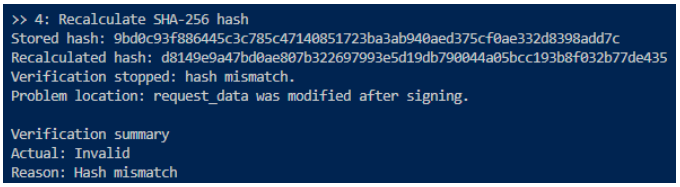

Fig. 6 Example of an invalid request where the system detects a hash mismatch after the request data is modified.

The command-line output shows the signing and verification process step by step. During the signing process, the program displays the original request data, validates the request structure, generates the SHA-256 hash, and signs the hash using ECDSA P-256. During the verification process, the program recalculates the hash, compares it with the stored hash, verifies the digital signature, checks the timestamp, and validates nonce uniqueness. If a problem is detected, the program displays the specific reason for the failure, such as hash mismatch, invalid digital signature, expired timestamp, replay detected, or unverified payment status.

B. Tampering Detection

The tampering detection test was conducted to evaluate whether the proposed system can detect unauthorized changes in donation-based video request data. The test was performed by creating a valid signed request and then modifying specific fields to simulate possible attacks or invalid conditions. The tested scenarios include changes to the video URL, donation amount, queue position, timestamp, digital signature, stored hash, expired timestamp, reused nonce, and payment status.

TABLE I. Tampering Detection Result

Test Case	Scenario	Expected Result	Actual Result	Verification Reason
TC-01	No modification	Valid	Valid	Valid request
TC-02	Video URL changed	Invalid	Invalid	Hash mismatch
TC-03	Donation amount changed	Invalid	Invalid	Hash mismatch
TC-04	Queue position changed	Invalid	Invalid	Hash mismatch
TC-05	Timestamp changed	Invalid	Invalid	Hash mismatch
TC-06	Invalid digital signature	Invalid	Invalid	Invalid digital signature
TC-07	Request and stored hash changed, signature unchanged	Invalid	Invalid	Invalid digital signature
TC-08	Expired timestamp	Invalid	Invalid	Timestamp expired
TC-09	Reused nonce / replay request	Invalid	Invalid	Replay detected
TC-10	Payment status is pending	Invalid	Invalid	Payment status is not verified

As shown in Table I, the original request without modification was successfully verified as valid. This indicates that the signing and verification process works correctly when the request data is unchanged. In contrast, all modified or invalid requests were rejected by the system. The modification of the video URL, donation amount, queue position, and timestamp resulted in a hash mismatch. This happens because SHA-256 produces a different hash value when any part of the input data changes. Therefore, even a small modification in the request data can be detected during verification. This result shows that SHA-256 is effective for maintaining the integrity of video request metadata.

The invalid digital signature test also produced the expected result. When the signature was changed, the system rejected the request because the signature could not be verified using the public key. In another scenario, the request data and stored hash were both changed, but the signature was not regenerated using the private key. The system still rejected the request because the stored signature no longer matched the modified hash. This shows that the digital signature provides an additional layer of protection. Even if an attacker can recalculate the hash, the attacker cannot create a valid signature without access to the private key.

The replay attack scenario was tested by reusing the same nonce. The system rejected the second use of the request because the nonce had already been recorded. This indicates that nonce validation can help prevent old valid requests from being reused. The expired timestamp scenario was also rejected because the request was outside the allowed time window. In addition, a request with pending payment status was rejected because only verified payment status is allowed to enter the video queue. Overall, the tampering detection results show that the proposed mechanism can detect several types of invalid requests. SHA-256 is useful for detecting changes in request data, digital signature is useful for verifying authenticity, and timestamp with nonce validation is useful for reducing replay attack risks.

C. Performance Analysis

The performance test was conducted to measure the processing time of SHA-256 hashing and digital signing with the following result.

TABLE II. HASHING VS. SIGNING PERFORMANCE

Number of Requests	Hashing Total Time (ms)	Hashing Avg. Time (ms/request)	Signing Total Time (ms)	Signing Avg. Time (ms/request)
100	0.818	0.00818	23.611	0.2361
500	1.886	0.00377	11.429	0.0228
1000	3.760	0.00376	22.720	0.0227
5000	18.607	0.00372	112.459	0.0225

Based on Table II, SHA-256 hashing has a lower processing time than digital signing. For 5000 requests, the total hashing time was 18.607 ms, with an average of 0.00372 ms per request. Meanwhile, the total signing time was 112.459 ms, with an average of 0.0225 ms per request. This indicates that SHA-256 hashing is very lightweight for processing video request metadata.

Digital signing requires more processing time than hashing because it uses asymmetric cryptographic operations. Unlike hashing, which only produces a fixed-size digest from input data, digital signing involves private key operations. Therefore, it is expected that signing will be slower than hashing. However, the average signing time remains small and can still be considered acceptable for a metadata-based video submission system. The result also shows that the average hashing time becomes relatively stable when the number of requests increases. For 500, 1000, and 5000 requests, the average hashing time stays around 0.0037 ms per request. The signing time also becomes more stable for larger request sizes, with an average time around 0.0225 ms per request for 5000 requests. The higher average signing time in the 100-request test may be caused by measurement overhead, initialization cost, or small sample size.

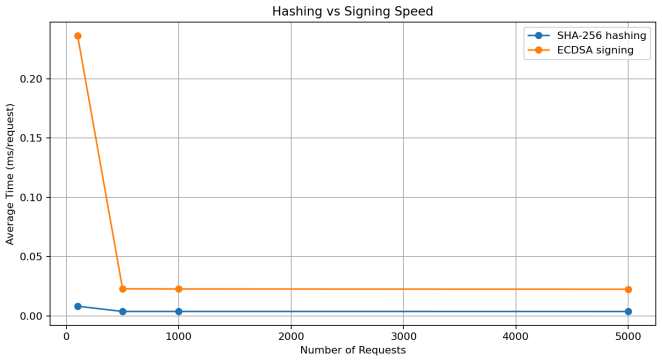

Fig. 7 Hashing vs. Signing Performance in ms

Fig. 7 shows that the signing line is higher than the hashing line, meaning that signing is slower than hashing. However, both operations still show low average processing time per request. This result supports the practicality of the proposed system because the cryptographic operations are applied only to request metadata, not to the video file itself. As a result, the system can provide integrity and authenticity protection without introducing excessive computational cost.

V. CONCLUSION

This paper proposed a secure donation-based video submission system using SHA-256 and digital signature for Media Share platforms. The system was designed to protect video request metadata, including donation ID, donor name, donation amount, video URL, timestamp, nonce, payment status, and queue position. SHA-256 was used to generate a hash value from the request data, while digital signature was used to verify the authenticity of the request.

The program implementation showed that the proposed system can detect unauthorized modifications in video request data. Changes to the video URL, donation amount, queue position, and timestamp were detected through hash mismatch. Invalid or forged signatures were rejected during digital signature verification. In addition, expired timestamps, reused

nonces, and unverified payment status were also rejected by the system.

The performance test showed that SHA-256 hashing is faster than digital signing. For 5000 requests, hashing required only 18.607 ms in total, while signing required 112.459 ms in total. Although signing is slower than hashing, the average processing time per request remains low. This indicates that the proposed mechanism is practical for securing metadata-based video submissions in Media Share platforms.

Based on the results, the combination of SHA-256, digital signature, timestamp, and nonce can improve the integrity, authenticity, and reliability of donation-based video submission systems. The proposed system can help prevent tampering, fake requests, and replay attacks before a video request is added to the playback queue. However, this research is still limited to a prototype implementation and simulated request data. The system has not been directly integrated with real donation platforms or livestreaming services. Future work can include integration with real Media Share APIs, secure key storage, automatic moderation, malicious URL detection, and larger-scale performance testing in a real streaming environment.

VI. Appendices

The implementation source code of the program can be accessed on a repository in the following:

https://github.com/Clazz1/mediashare_cryptography

Acknowledgment

First, All praise and gratitude to the Almighty Allah SWT whose grace and blessings giving me a chance to finish this paper. Secondly, I would deeply thank to my family who always caring and supported me until now. Then, I would express my gratitude to the chairman lecturer of II4021 Cryptography Dr. Ir. Rinaldi Munir, M.T. . Lastly I would thank my friends for the encouragement especially from the *Genshiken Sekre Belakang* who gived me helpful advice to finish this paper.

References

[1] Streamlabs, "Media Share Widget: Video & Song Requests for Twitch, YouTube, & More," Streamlabs. [Online]. Available: https://streamlabs.com/stream-widgets/media-share. [Accessed: Jun. 19, 2026].

[2] Saweria, "Dukung Mediashare," Saweria. [Online]. Available: https://saweria.co/mediashare. [Accessed: Jun. 19, 2026].

[3] National Institute of Standards and Technology, "FIPS 180-4: Secure Hash Standard (SHS)," NIST, Aug. 2015. doi: 10.6028/NIST.FIPS.180-4. [Online]. Available: https://csrc.nist.gov/pubs/fips/180-4/upd1/final. [Accessed: Jun. 19, 2026].

[4] National Institute of Standards and Technology, "FIPS 186-5: Digital Signature Standard (DSS)," NIST, Feb. 2023. doi: 10.6028/NIST.FIPS.186-5. [Online]. Available: https://csrc.nist.gov/pubs/fips/186-5/final. [Accessed: Jun. 19, 2026].

[5] R. Shirey, "RFC 4949: Internet Security Glossary, Version 2," Internet Engineering Task Force, Aug. 2007. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4949. [Accessed: Jun. 19, 2026].

[6] National Institute of Standards and Technology, "Nonce," NIST Computer Security Resource Center Glossary. [Online]. Available: https://csrc.nist.gov/glossary/term/nonce. [Accessed: Jun. 19, 2026].

[7] OWASP, "SCWE-022: Message Replay Vulnerabilities," OWASP Smart Contract Security Verification Standard. [Online]. Available: https://scs.owasp.org/SCWE/SCSVS-COMM/SCWE-022/. [Accessed: Jun. 19, 2026].

Statement

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Muhammad Rafly Fauzan 18223132