Video Streaming Security: Reliable Hash Chain Mechanism Using Redundancy Codes

Emad Mohamed Abd Elrahman Abousabea, Mohammed Boutabia, Hossam Afifi

Video Streaming Security: Reliable Hash Chain Mechanism Using Redundancy Codes

Emad Abd-Elrahman, Mohammed Boutabia and Hossam Afifi
TELECOM SudParis, CNRS SAMOVAR UMR 5157
9, Rue Charles Fourier - 91011 Evry Cedex, France
{emad.abd_elrahman, mohamed.boutabia, hossam.afifi} @it-sudparis.eu

ABSTRACT
The prospection of video streaming security has been changed considerably during the last years. With the new generation of hand healed devices and the delivery rate up to 2 Mb/s, the new prospection searched for fast security measures that have no great effects on the streaming fluidity. The hash chain has been largely used for such applications. The benefits from this deployment are the fast and the light calculations. But, the hash chain still suffers from some drawbacks related to chain link and robustness. This work studies different methods for achieving resynchronisation state for hash chain link. It also proposes a hybrid algorithm based on redundancy codes and windows flow which called *Redundancy Code Synchronization Recovery State (RC-SRS*). This technique merges the pros of all methods and avoids the cons of them. In the end, analytical and simulation results for the hybrid algorithm have been made. The results indicate that, that proposal has a good overall performance in terms of complexity and calculation time.

General Terms
Security, Reliability, Performance.

Keywords
Streaming, Hash Chain, Resynchronisation, Redundancy Code.

1. INTRODUCTION
Hash chain concept was firstly proposed in [1] for securing password authentication. After that many applications used this technique as a light and a fast security way. Hash chain has achieved a remarkable successful for securing popular applications like authentication of multicast traffics [6, 12], the routing of sensor networks and sensors applications [7], the privacy of RFID authentication [8], data streaming [9], micropayment systems [10], one time password [1] and a lot of data origin authentication applications. The main advantage of hash chains is the light calculations compared to other cryptographic algorithms like encryption algorithms. It also provides a fast and secure way for the real time applications which are very sensitive to any delay caused by the security measures.

The specific natures of video streaming (time sensitive and packet loss tolerance application) forced researchers to find a suitable security measure for it. Therefore, the elasticity and light calculations of hash chain attract the security research towards using it in heavy loaded applications like video streaming. So, before any explanations, we must know the nature of video streaming server and client as follows:

Video streaming: Streaming means delivers media from a video server over any type of network (like Internet) to any client in real time mode. In this case, no video file is ever transferred or downloaded to the client's machine. But, the media is played by the client's software as it is delivered by the specified rate. This streaming includes the IPTV, VOD and Broadcasts of live personal events in real time. So, the system has two components:

The streaming server: is responsible for streaming the original video or audio file (creates streams) in standard real time format for transporting over the network and also responsible for handling the clients' requests to access the hosted videos and audios files. The server software is responsible for sending the media over the Internet to the client machines.

The client's software: is responsible for reading the received real time standard packets and decoding it to the specific codes for valid viewer application.

Online or live videos: The live events, such as concerts, speeches, and lectures, are commonly streamed over the Internet as they occurred with the assistance of special broadcasting software. The broadcasting software encodes a live source, such as video from a camera, in real time and delivers the resulting stream to the server. The server then sends the live stream to the clients.

Regardless of when different clients connect to the stream, each one could see the same point in the stream at the same time of his joining.

Offline or On-demand video: For an On-demand video delivery, the files are archived and pre-stored on the server. Each client initiates the stream from the beginning, so no one ever comes in "late" to the stream. In this case, no need for broadcasting software from the server side. YouTube provides a good example for such case which contain a huge database of short videos and represent a successful model for file allocation that attract many users daily [17].

This work studies the reliability problem for using hash chain with video streaming. The study focuses on the effects of packet loss delay and replay attacks on broking the hash link. The proposed algorithm can overcome on those attacks by its reliable window mechanism with the help of secure redundancy codes.

The rest of this work is organized as follows: part 2 gives overview on the related work. Section 3 discuses the hash chain

MOMM2010, November 8-10, 2010, Paris, France.

synchronisation problems and proposed solutions comparison. Part 4 illustrates our algorithm architecture and assumption for achieving redundancy of hash link. Section 5 highlights the window mechanism challenges against some popular attacks. Section 6 shows the results. Section 7 concludes our work.

2. RELATED WORK

Hash chain is a successive application of a cryptographic hash function *h(.)* to any string. The link of chain means that; the initialisation value currently input to *h(.)* will be the output of the previous hash calculated from the previous part of data.

Hash chains for video streaming and multicast applications have been considered extensively in the literature. However, handling of the resynchronisation problem for broken hash links still needs a lot of work. In [2], the principle of synchronisation point for chaining was proposed in cooperative with watermarking chain to secure data stream.

A good starting point for how to sign a digital streaming video was introduced in [13]. Authors proposed two cases; offline and online streams. They chain blocks based on the packets inside the block. Each block carries the hash of the next one (online case). For the offline case, they calculate the hash based on the whole video and the receiver must have some buffer so as to start the verification after a specific length of the video. Their algorithm does not handle the redundancy of chain links.

In [14], authors introduce the Butterfly Graph. They divided the packets into groups and each group has one signature calculated based hashing. The redundancy is achieved by sending the signed packets several times. Their overall concern is to keep a good performance as the amount of redundancy is increased.

In [15], the work is based on signing a small number of special packets in data stream; each packet is linked to a signed packet via multiple hash chain. The links depend on six hashes per packet. Hence six packets carry the same hash value and this represents a large overhead. Two solutions for securing the video stream are compared. The first solution is called TESLA (Timed Efficient Stream Loss-tolerant Authentication). The second scheme is called EMSS (Efficient Multichained Stream Signature).

Another work [16], handled the video stream authentication by assuming a combination of one-way hashes and digital signatures to authenticate packets. Their idea can be explained as follows; if a collision resistant hash of packet P_i is appended to packet P_{i+1} before signing P_{i+1} then, the signature on P_{i+1} guarantees the authenticity of P_i and P_{i+1} at the same time. The drawback of that proposal is the large overhead as it increased linearly with the growth numbers of packets.

3. HASHING METHODS

In general, the hash is an irreversible cryptographic process (one way function). This means that, it is too hard to apply the inverse transformation, and you can therefore only compare hashes results. The advantage is that the short processing time which has no effects on the media quality.

The one-way functions used are based on popular cryptographic hash functions, such as MD5 [4] or SHA-1 [5]. Numerous works have been proposed to drive light and efficient one way function chains based on MD5 or SHA-1 algorithms as a hashing based [3]. The objectives of that work were to construct fast generation and verification of hash chains.

When the hash chains are applied to the video streams like VoD or IPTV, it faces several problems for keeping the hash link continuity in case of packet drops, packet delay or some packets replay.

In the next sections, we propose solutions for those problems and discuss advantages and disadvantages of each technique. We classified the resynchronization algorithms into 4 types, each type has its own advantages and disadvantages. Those 4 types are:

- **SHHC**: Self-Healing Hash Chains
- **TSP** :Time-Synchronisation Point
- **MLHC**: Multi Layer Hash Chain
- **TSS**: TimeStamp Synchronisation

And at the end, we propose a hybrid mechanism from all the methods mentioned above which called **RC-SRS**: Redundancy Code Synchronisation Recovery State. The performance of this algorithm against the others will be discussed at the end of this section.

3.1 Self-Healing Hash Chains (SHHC)

Videostreaming can be based on TCP or UDP transport protocols. TCP is mainly used to overcome Network Address Translation (NAT) filters but the most appropriate is UDP. Usually, during online streaming, we have some packet drops that can be measured with respect to some packet loss tolerance. The acceptable tolerance may not affect the streaming quality. However, the loss could affect synchronisation of the hash chains of the stream.

The SHHC can overcome this problem by re-synchronizing the chains despite the loss of some packets from the stream. The SHHC is a robustness system able to resolve the synchronisation problem of chains. As in Figure 1; the stream is divided into specific time blocks (Δt) and at each time a hash must be calculated for this period of time. The parity will depend on some redundancy of the concatenated hashes. To guarantee the synchronisation, three hashes may be concatenated together.

The advantages of this scheme are the low overhead for memory and calculations. This procedure will add some redundancy for tracking the synchronisation points of the stream.

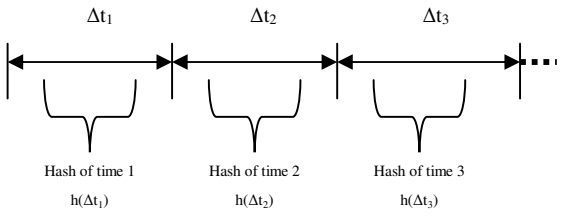

Figure 1: The time hash-chain of sending party

3.2 Time-Synchronisation Point (TSP)

This mechanism is used to assure synchronisation of hash chains in case of packet loss. It depends on adding additional information bits to the steam. The stream must be divided into a pre-defined specific time blocks. After each block, a synchronisation point must be inserted in the sender side as shown in Figure 2. Those

inserted points can help the receiver tracking the synchronisation of the stream.

In this case, the receiver must keep in tracking those time synchronisation points (TSP) to not lose the stream synchronisation or the hash link breaks. The drawback of this one is the large bits overhead added for synchronisation.

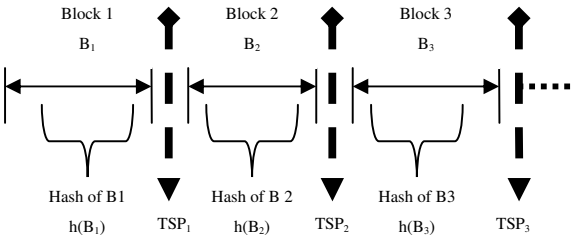

Figure 2: The time synchronisation point of sending party

3.3 Multi Layer Hash Chain (MLHC)

This technique was used for some applications and gave good results for the problem of security assurance for the E-lottery winners and their serial numbers generation tickets [7]. The multi layer means here to have calculations for hash chains where each calculation is representing one layer according to the base of calculation. Although the objectives are different (e-lottery and video streaming), this technique could be very effective especially in offline video streaming security mode. When we use this technique in video streaming the layers conception will completely be different so as to match the specific nature real time applications. The hash levels calculations for e-lottery algorithm are shown in Figure 3.

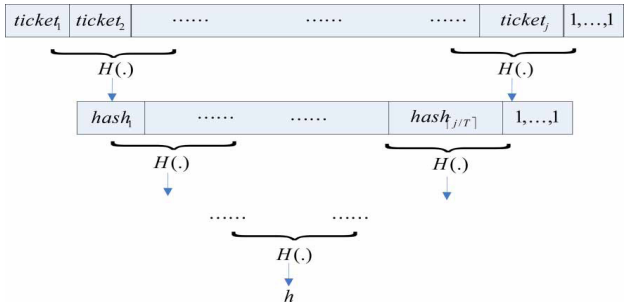

Figure 3: The multi level hash chains as proposed in [7]

3.4 TimeStamp Synchronisation (TSS)

The sequence of packets could be used as a good measurement for achieving video synchronisation and also keeping the link of hash chain. This can be efficient with less calculation cost and time overhead because the timestamp is a mandatory field in RTP packet as described in [18] for RTP packets of MPEG-4 streams. Figure 4 illustrates the stack for sequence number and timestamp parts of RTP standard packet. In this case, The RTP packets are responsible for sequence numbers and timestamp synchronisation (TSS) between the source and destination. The benefit of this technique is the reuse of parameters from RTP standard packets as shown in Figure 4.

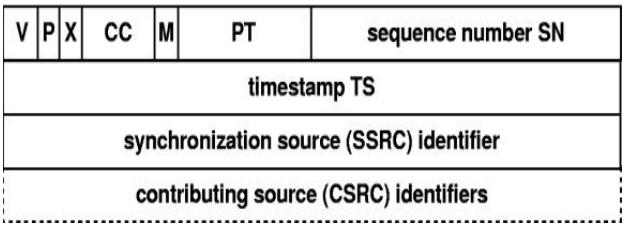

Figure 4: Part of RTP packet header from MPEG stream.

To conclude those four techniques, we can make performance comparison between them as shown in Table 1. This comparison is the base for building the proposed hybrid technique.

We suggest a hybrid technique (**RC-SRS)** capable of inheriting the advantages of the previous solutions and that overcomes most of their drawbacks. This technique could be useful for slow processor endpoints so as to minimise the calculation needs during every session. Also, it could be used to rapidly re-establish the link synchronization in case of packet loss session problems or delay time. Another criterion is packet and hash information caching that have more CPU intensive compared with using time synchronization that may be important for low-memory mobile platforms.

Our proposal is inspired from redundancy code techniques. **Redundancy Code (RC)** is a generic concept introducing some redundancy in the system to overcome hash chain break in case of packet loss and to increase the reliability of the security system. We also adopt window mechanism for reliable and secure hashing links. Moreover, the indexing mechanism that we used besides window helps us a lot overcoming the problem of packet reply.

Table 1: comparison between different techniques for resynchronisation in term of performance

	Robustness	**Recovery Time**	**Memory Overhead**	**Bits Overhead**	**Calculation Overhead**	**Video Streaming Mode**
Self-healing	High	Low	Very Low	Very Small	Small	Online & Offline
TSP	Moderate	Very Low	Small	Large	Low	Online & Offline
Multi-Layer	Low	Large	Large	Small	High	Offline
TimeStamp	Moderate	Very Low	Low	Small	Very Low	Online & Offline

4. THE (RC-SRS) HASH CHAIN METHOD

Before the description of the proposed architecture, we must have a look on the packetization sequence of the video stream. Figure 5 illustrates the simple sequence in standard manner based on real time transport protocol as in [18]. The video is considered as a group of chunks output from the coder such as MPEG-TS [19]. This gives better clarification on which the stream word represents for us and what is the packet structure for our proposal. The hashing calculations will be done after the highlighted row in Figure 5 (for transmitter) and before in the case of the receiver.

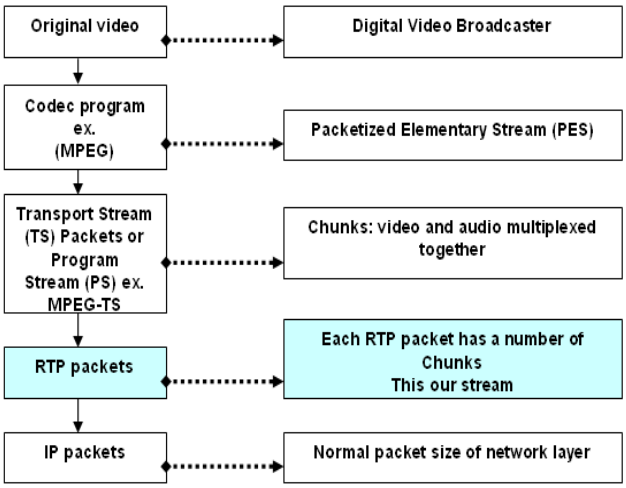

Figure 5: The sequence diagram for video packetizing in the transmitter side

4.1 The Architecture Design

The proposed architecture in Figure 6 has many parameters that need to be initialized:

- **Stream of Chunks:** are the output blocks after MPEG-TS (like RTP packets).
- **Blocks Bi:** are the blocks of packets that have a relation with their numbered Chunks. For example; each Block=10 Packets and each Packet=7 Chunks in case of RTP Packet.
- **IV0:** is the initial vector to start the hash chain (we will consider the same proposed values by the standard of MD5 or SHA-1).
- **h(.):** is the hash functions used for calculate the output hash like SHA or MD5 series.
- **hi:** is the output hash value or in some cases, the output digest.
- **Combination Code:** is the coding process that will be used to calculate a redundancy code for generating the hash value in case of a missing hash value of a block (it is a combination of logical functions like XOR function).
- **RC:** is the output Redundancy Code that is responsible for recalculate the missed hash value so as to keep the hash link not broken. It is based on Forward Error Correction (FEC) codes.

The added Redundancy Code will allow the receiver to detect and correct the errors in the digest values (under some restrictions). This code is the key factor for solving the resynchronization problem of hash link. This could happen without the need to ask the sender for additional data retransmitions because the sender is memory-less in case of video streaming. The advantages of RCs are that; buffering is not required and the retransmission of hash values can often be avoided (which reduces the bandwidth requirements and time plus the memory). RC is therefore applied in this situation where the retransmissions are difficult to achieve in real time applications and memory-less devices. The main objective from RCs is the hash link synchronization and finding the recovery point of synchronization by obtaining the hash value of this time. It represents the initial vector for next hash calculation in our chain.

4.2 Assumptions

Although Digital Signature (DS) is used frequently in video streaming authentication and verification, it had some drawbacks. It could assure that the receiving data is not altered or changed during its routing from source to destination. But, the problem is the large overhead produced by it [20]. So, our algorithm proposes a security measure for video streaming based on pure hashing.

Our proposed solution is built to fit the new generation of handheld devices that have some limitations in all processing capabilities compared to the normal PCs. So, the treatment of any video will be considered as an online one (from the receiver side) although if in some cases, the sender knows all videos lengths accessed by the others. This assumption will eliminate the need of data buffering at the receiver side before starting the video playing.

We assumed that, the redundancy in this case is mandatory for synchronization matter. But, when we calculate the RC for some parts of data, this calculation will mainly depend on the degree of redundancy and the accepted overhead.

For example, if RC calculated based on 3 hash values (as shown in Figure 6) like *RC1= combination (h1, h2, h3)* and *RC2= combination (h3, h4, h5)* then we have redundancy 3/4. we take 4 hash values the redundancy will be 4/5, and so on. So, the degree of redundancy will be the key factor that controls the calculations of the RCs codes.

The block diagram in Figure 6 explains the different steps of hash chain process done by the sender. It has some tasks to prepare video streaming packets before hashing. Then, the chain starts with initial value IV0 (standard value according to the type of hash). After that, the RC will be calculated according to the redundancy factor and predefined window size. The final step is to put the RC inside the packets and the index towards them in the other packets.

For the receiver verifications, it will apply the rules in section 5 to assure the reliability and integrity of the streams. The sequence will be reversed and starting from which the sender finished his process of hashing. After receiving the window it will calculate the hash and compare it with the received ones in the RC. In this case, the receiver will not find the hash directly but it will drive it from the RC. That is why the RC-SRS algorithm also has the self-healing features. Finally, after the verification process, the receiver passes the block of packets to the decoder for playing the stream.

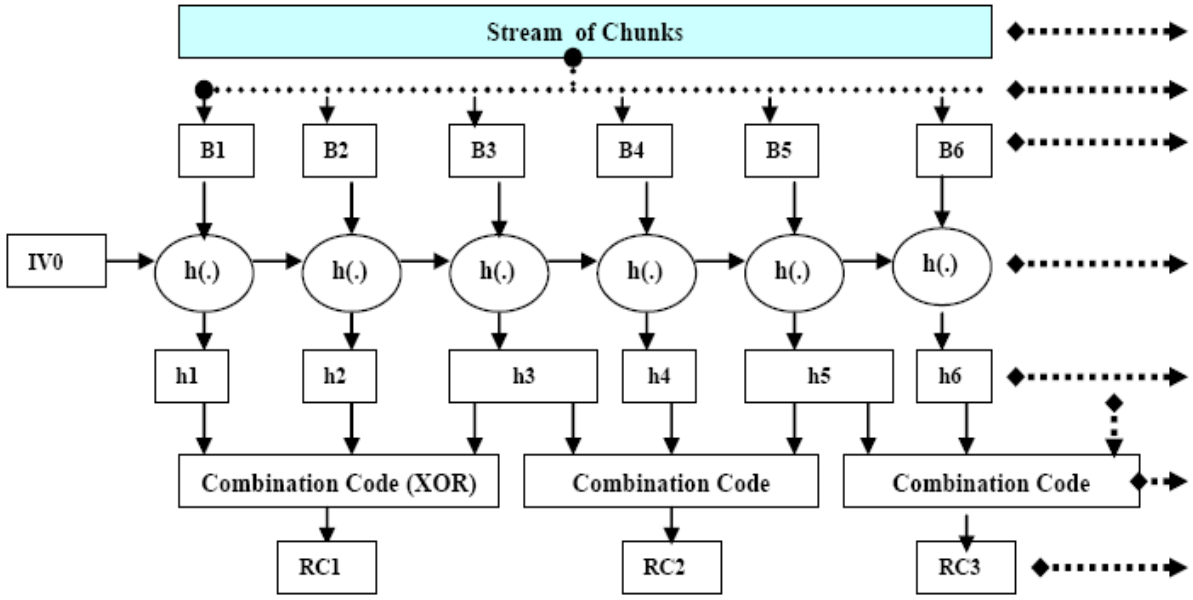

Figure 6: Block diagram for the hash chain architecture based redundancy for video streaming. The original stream is divided into chunks then assembled them to specific Blocks Bi after that the hash chain applied to the Blocks; finally, the RCs calculated.

5. WINDOW MECHANISM

This mechanism proposes a high degree of reliability for hash chains. The algorithm depends on an agreement held between the server and clients about the processing capabilities for a number of packets or bytes per window. This window size also has direct relation with the video streaming rate. The RC code is calculated based on this size and each window has sequence number for reliability considerations. Figure 7 illustrates this window mechanism. So, the *window* is a dynamic buffer which its size defined by video playing rate and receiver processing memory.

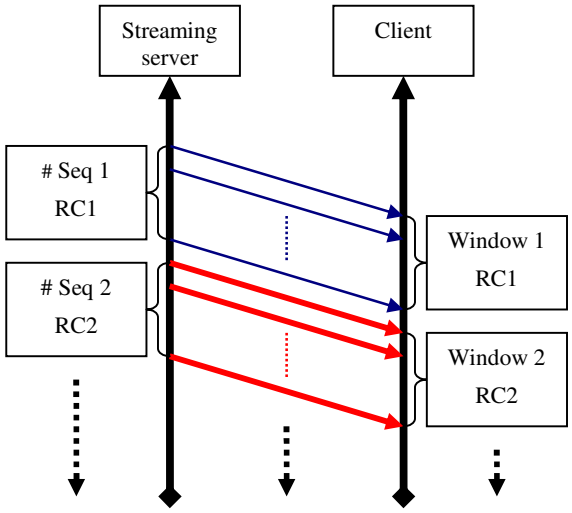

Figure 7: The window redundancy mechanism

For example, if the RC combines 3 hash values of three Blocks then the window size will be 3x Block size. So, the window has a direct relation with the redundancy calculation. Also, the buffer has a relation with the delivery rate of playing stream and processing capabilities of the clients. The following parts discuss the challenging of synchronisation and their treatments.

5.1 Online versus Offline Synchronization

Online: It is not loosely time synchronization because the joining time must be the initial point of synchronization between the hashing generator in the sender and the verification process in the receiver.

Offline: It is loosely time synchronization case because any receiver can freely join the stream at any time it likes. At any initial time value (*Ti*), the verification side can negotiate the starting point with the generator side.

We have two scenarios of joining clients:

- ***Clients start running from initial value of stream:*** in this case the initialization vector will be the standard one proposed by the type of hash method. For example SHA-1 has a specific 160 Bit IV and MD5 has also standard 128 Bit IV.
- ***Clients start running after some time passed from the stream:*** in this case, the client joins the online stream after some time of the broadcasting start. So, the problem is the initialization vector for starting the process of chain. Each window has RC value that will help the receiver in finding the IV for its joining time.

5.2 The reliability: anti-reply and anti-delay

The reliability and robustness could be achieved by the sequence number of the redundancy code. Each time the sender prepares the window size packets defined the sequence number for those bundle of packets. The receiver does not check the bundle that did not arrive in its right order so as to prevent the reply attack. This sequence number is embedded by the sender side as shown in figure 7 for each window. For the drop or loss packet attack, we have in this case; each packet has an index to the location of RC code that will help the receiver verification for this packet. Each window has specific number (N) and each packet has an index number (n) where each ($n \leq N$). The packet that will come within out of its order ($n \succ N$) it will be directly dropped by the receiver verification procedure.

If we have any file with (n) packets it will be divided to N windows or Blocks where each (N=n/m) where (m) is the number of packets per window. The index number of packets (P_1 till P_m) for each window has a direct relation with the generated sequence number of any window (N).

5.3 Collision Avoidance

We assume the hash function that will be used is H, also we have two different messages x, y (where $x \neq y$). If $H(x) = H(y)$ this means a collision occurs.

As we said before, the simple idea of hash chains is that each packet will carry the hash of the previous packet, that's why we called it chains.

We propose a simple hash algorithm that will decrease the computations layer into two layers only. The first one will hash the original message and the output will be the Ho (hash original) and the second one will be the Hn (hash new) to the previous Ho.

For example, if we have message X=(x1,x2,……..,xn) the hashing output will be H(X)=h(x1),h(x2),……..,h(xn), which represent the normal hash (the first phase in figure 8).

In the second phase we will apply hash chains Hc(X)=hc1(h(x1)), hc2(h(x1)),……, hcn(h(xn)). That is why RC-SRS algorithm has the layering features. The final message X that will be sent is:

$$X=\{x1 \parallel hc1(h(x1)),\ x2 \parallel hc2(h(x2)),\ldots\ldots,\ xn \parallel hcn(h(xn))\}$$

If any one detect this message and try to calculate the hash of X, he will find H(X) not Hc(X) because Hc(X)=H(H(X)) on the base of Hc is hash chains. The hash chains used IV that is generated locally on the client and no one can detect it easily. So, the degree of security for the second phase is one of our objectives to enhance the data integrity sent or received. The summary of the layers process of hashing are shown in Figure 8.

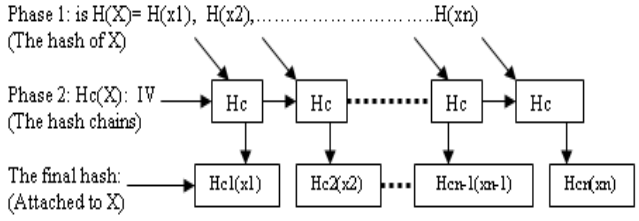

Figure 8: The layering process for collision avoidance

For any man-in-the-middle (MITM) captures the data X and calculates its hash H(X), it will not recover Hc(H(X)). So, the data integrity will then be assured.

Till now, we have H(X) the output of first hashing and H(IV,X) the output of second hashing (pure hashing). But, we can add more security if we use H(Ks,IV,X) which Ks is security key applying on the hash function and this leads us to Digital Signature (DG) which is beyond to this work . This Ks value can be generated by private key generator PKG and distributed by Diffie-Hellman (DH) key management algorithm [21] or Elliptic Curve Cryptography (ECC) technique [22].

5.4 Indexing Mechanism

For MPEG-TS, the output chunks as in figure 5 are equal 188 Bytes/chunk. The RTP packet size as we captured from the packet analyser during the simulation was 1370 Bytes which equal 7(chunks) x188 Bytes + 54 (total headers rest). So, on the base of 1500 Bytes standard packets which called the Maximum Transmission Unit (MTU), we still have 1500-(1370+2 bytes for index to the place of RC) = 128 Bytes. We will depend on using two bytes in our indexing mechanism. The RC value will be put in any random packet inside the window so as to overcome the packets tracking attack. Each packet inside the window has hierarchal indexing towards the RC location starting with window number (as shown in figure 7) followed by packet order inside this window that contains this redundancy code.

For the sender, it prepares the packets and piggybacks the calculated hash values and redundancy code in the packets. Also, each packet will have an index to that place in the packet which carries this code.

For the receiver, as each packet contains an index to RC and window number. It drops the packets that not belonged to the receiving window sequence and have not the right indexing.

6. RESULTS

We construct our results based on the ITU-T recommendations for real time applications delay; (Packet transfer delay must be < 150ms, acceptable < 400ms, not acceptable > 400ms and Jitter around 50 ms). Our analytical and simulation results were built based on the previous recommendations for end-to-end delay. The previous recommendations are important for interactive media like audio and video conferencing. More ever, if we increase this end-to-end delay to be accepted < 2s this will not affect the streaming as it is one direction and not interactive.

The end-to-end delay = delay of (digitization + packetization + transmission + propagation + queuing + jittering + hashing). We assume that; the delays of digitization, transmission, propagation and queuing are negligible and jittering time is fixed. So, E2E (end-to-end) delay can be calculated as:

$$E2E = D_p + D_h$$

Where D_p is the packetization delay and D_h is the hashing delay, which composes of two times as:

$$D_h = 2.T_{hcalc} + T_{hverif}$$

Where T_{hcalc} represents the time consumed for hash calculations in sender or receiver (we assumed it is equal) and T_{hverif} represents the hash verification time consumed in the receiver.

Figure 9 illustrates E2E delay for different types of video rates starting from 64Kb/s till 2Mb/s. We notice that, the D_p is inversely proportional with the increasing of video rate. The hash based algorithm used for all is MD5 standard.

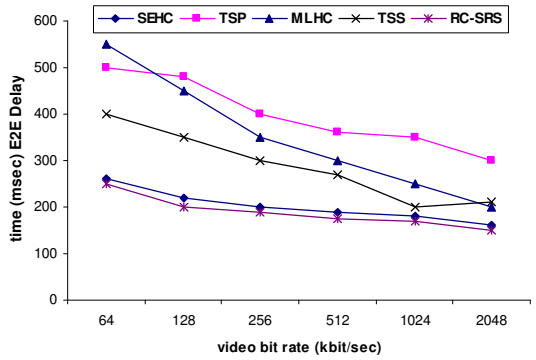

Figure 9: Performance in terms of total computation time (E2E) depends on MD5 hashing.

Figure 10 shows same calculations for E2E delay comparing the 5 algorithms for hash chain resynchronisation based on SHA-1 standard. The difference in hashing calculations between MD5 and SHA1 is almost about 30 ms as the output link in SHA1 (20-Bytes) and in MD5 (16-Bytes). But, this big difference in delay between them as shown in figure 9 and figure 10 is depending on the number of redundancy codes that will be taken into account for reliability as it illustrates in Figure 12.

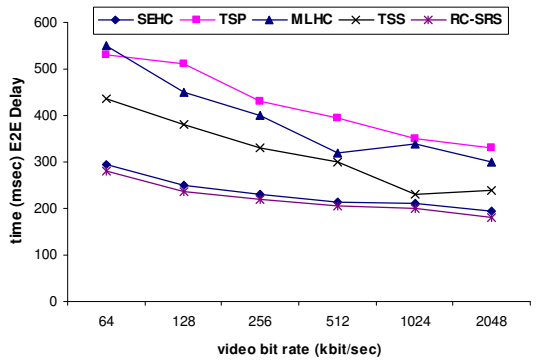

Figure 10: Performance in terms of total computation time (E2E) depends on SHA-1 hashing.

We also tested the performance evaluation for the hybrid algorithm RC-SRS against the Packet Error Rate (PER). We assumed different percentages of PER from 0 to 0.5 as shown in Figure 11. The results indicate that; more than 25% of packet loss could lead to hash broken. This means that, the RC algorithm can guarantee up to 0.25 losses of video stream packets. This percentage seems to be very high and not acceptable for the fluidity of the stream but the more affected negatively is the audio part. The quality of performance based on MD5 or SHA-1 is far away as shown in the Figure 11. This figure gives just an example of video rate 1 Mbps (1024 Kbps) which indicates accepted loss till 0.1% from the packets streaming to have recovery state near 100%. According to standard, the accepted losses could not exceed 0.01%.

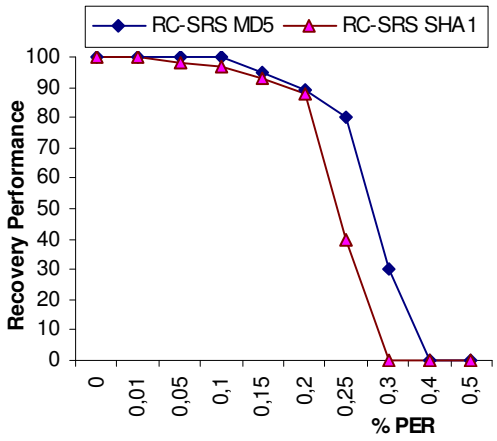

Figure 11: The percentage of recovery state against the packet error rate (PER) for 1024 Kbps video stream.

In term of the redundancy overhead and the best allowable added bytes by the RC to the packets of stream, we conducted analytical calculation for RC based MD5 or SHA-1.

Figure 12 compares the total overhead (the added bytes to stream as a redundancy code) in case of using MD5 and SHA-1 hash algorithms. As shown, if we assume the number of packets per block equal 10, so the full redundancy means sending the RC 10 times (means with each packet). But, this will lead to very high overhead.

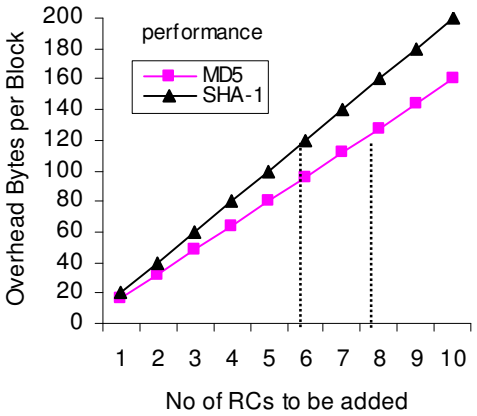

Figure 12: The overhead bytes in terms of number of redundancy trials for RC-SRS.

It is well known that, the MPEG-TS output chunks are equal to 188 Bytes/chunk. The RTP packet size as we captured from the packet analyser during the simulation was 1370 Bytes which equal 7(chunks) x188 Bytes + 54 (total headers rest). So, on the

base of 1500 Bytes standard packets MTU, we still have 1500-(1370+2 bytes to index the place of RC) = 128 Bytes. Those 128 Bytes give the probability of sending the RC 8 times in case of MD5 and 6 times in case of SHA-1 as shown in Figure 12. Those results obtained under our assumptions of packet and block sizes.

7. CONCLUSION

This work has been proposed a complete study about the hash chain resynchronisation problem and solutions that could be used for its recovery with any kind of data streaming. After that, the comparison between methods leads to a hybrid algorithm which called RC-SRS. The analytical and simulations results for this algorithm assure the reliability and efficiency of it over the 4 algorithms (SHHS, TSP, MLHC and TSS). In term of complexity, a comparison has been made between those different ways for achieving the resynchronisation of hash chain. The RC-SRS for resynchronisation based on redundancy codes gives acceptable results which indicate that the RCs will not cause harmful computations time for sender or receiver verifications. Also, the overhead added was accepted in term of the standard packet size and MTU 1500 Bytes. The final contribution for RC-SRC is its hybrid features of (SHHS, TSP, MLHC and TSS) as; it is self-healing because it depends on the RC for driving the hash values at the receiver. It used indexing mechanism for achieving reliability like TSP. Also, it builds the technique of MLHC for overcoming the collision avoidance and finally the sequences numbers in packets and windows like TSS for preventing the delay and reply attacks.

As this work is interested mainly in the hashing for achieving the security measure reliability for video streaming, it is still lacking some degree of security. In the future, we will also integrate security signatures and analyse their robustness versus their verification time.

8. REFERENCES

[1] Leslie Lamport; 'Password authentication with insecure communication'; Communications of the ACM, 24(11), pp.770-772, November 1981.

[2] Huiping Guo, Yingjiu Li, Sushil Jajodia;' Chaining watermarks for detecting malicious modifications to streaming data'; Information Sciences 177, pp.281–298, 2007.

[3] Yih-Chun Hu and Markus Jakobsson and Adrian Perrig, 'Efficient Constructions for One-Way Hash Chains', ANCS'05, July 2005.

[4] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, April 1992.

[5] National Institute of Standards and Technology (NIST). Secure hash standard, Federal Information Processing Standards (FIPS) Publication 180-1, May 1993.

[6] Heba K. Aslan; 'A hybrid scheme for multicast authentication over lossy networks'; Computers & Security 23, pp.705-713, 2004.

[7] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar; 'SPINS: Security Protocols for Sensor Networks', ACM Mobile Computing and Networking, Rome, Italy, 2001.

[8] Syamsuddin, I.; Dillon, T.; Chang, E.; Song Han; 'A Survey of RFID Authentication Protocols Based on Hash-Chain Method'; Third International Conference on Convergence and Hybrid Information Technology ICCIT 08; Vol. 2, 11-13, pp.559 – 564, November 2008.

[9] Huiping Guo , Yingjiu Li , Sushil Jajodia; 'Chaining watermarks for detecting malicious modifications to streaming data'; Information Sciences, pp.281–298, 2007.

[10] Min-Shiang Hwang and Pei-Chen Sung ; 'A Study of Micro-payment Based on One-Way Hash Chain'; International Journal of Network Security, Vol.2, No.2, pp.81–90, March 2006.

[11] YINING LIU, LEI HU, and HEGUO LIU;' Using an efficient hash chain and delaying function to improve an e-lottery scheme ';International Journal of Computer Mathematics; Vol. 84, No. 7, pp.967–970, July 2007.

[12] Yacine Challal, Abdelmadjid Bouabdallah and Yoann Hinard; 'RLH: receiver driven layered hash-chaining for multicast data origin authentication'; Computer Communications 28, pp.726–740, 2005.

[13] Rosario Gennaro and Pankaj Rohatgi; 'How to sign digital streams', In Proceedings of the Advances in Cryptology CRYPTO'97, pp. 180-197, 1997.

[14] ZHISHOU, Z., APOSTOLOPOULOS, J. SUN, Q., WEE, S., AND WONG, W; 'Stream authentication based on generalized butterfly graph'; In Proceedings of the IEEE International Conference on Image Processing (ICIP'07), Vol. 6. pp.121–124, 2007.

[15] A. Perrig, R. Canetti, J. Tygar and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in Proc. of IEEE Symposium on Security and Privacy, pp. 56-73, 2000.

[16] P. Golle and N. Modadugu, 'Authentication streamed data in the presence of random packet loss', ISOC Network and Distributed System Security Symposium, pp.13-22, 2001.

[17] Abd-Elrahman Emad, Afifi Hossam;' Optimization of File Allocation for Video Sharing Servers', NTMS 3rd IEEE International Conference, pp.1 – 5, December 2009.

[18] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata ; 'RTP Payload Format for MPEG-4 Audio/Visual Streams', RFC 3016, November 2000.

[19] M.-J. Montpetit, G. Fairhurst, H. Clausen, B. Collini-Nocker, H. Linder; 'A Framework for Transmission of IP Datagrams over MPEG-2 Networks ' ; RFC 4259, November 2005.

[20] Pinkas, et al.; 'Electronic Signature Formats', RFC 3126, September 2001.

[21] E. Rescorla, 'Diffie-Hellman Key Agreement Method', RFC 2631, June 1999.

[22] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, B. Moeller; 'Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)', RFC 4492, May 2006.