Performance Analysis of Elliptic Curves for Real-time Video Encryption

Nilanjan Sen
Dept.of Computer Science and Engineering
University of North Texas
Denton, USA
nilanjansen@my.unt.edu

Ram Dantu
Dept. of Computer Science and Engineering
University of North Texas
Denton, USA
ram.dantu@unt.edu

Jagannadh Vempati
Dept. of Computer Science and Engineering
University of North Texas
Denton, USA
jagannadhvempati@my.unt.edu

Mark Thompson
Dept. of Computer Science and Engineering
University of North Texas
Denton, USA
mark.thompson2@unt.edu

***Abstract*—The use of real-time video streaming is increasing day-by-day, and its security has become a serious issue now. Video encryption is a challenging task because of its large frame size. Video encryption can be done with symmetric key as well as asymmetric key encryption. Among different asymmetric key encryption technique, ECC performs better than other algorithms like RSA in terms of smaller key size and faster encryption and decryption operation. In this work, we have analyzed the performance of 18 different ECC curves and suggested some suitable curves for real-time video encryption.**

Keywords—ECC, Real-time video streaming, Encryption, Security

I. Introduction

Rapid advances and developments in our Internet infrastructure and the plethora of applications that drive these technologies have made real-time media streaming a household commodity. In this context, we define real-time media streaming to be the continuous flow of a media stream, such as audio or video, over the Internet by a provider where the content is most likely presented to the end user before it has been entirely downloaded. With growth in the areas of video conferencing, web-based television services, e-learning, and telemedicine, popular Internet-driven businesses such as YouTube and Netflix provide live media streaming to both its corporate and individual users. For end-users, streaming video means that they can enjoy the convenience of watching the video almost as soon as it begins to download. As a result, Internet traffic sharing of live streaming video is increasing at a relatively constant rate and is expected to reach 82% of all consumer Internet traffic by 2021, according to Cisco's June 2017 Visual Networking Index report [1].

So, while consumers cannot get enough of multimedia streaming, security and privacy has become a major concern for both providers and end-users who are feeling the effects of copyright infringement from illegal sharing of multimedia data, public revelation of sensitive data, and the like. In domains such as telemedicine, real-time streaming video can be very problematic as authentication and encryption issues present impersonation and privacy challenges, especially when valuable multimedia assets are being distributed and transmitted over a network. The Real-time Transport Protocol (RTP) is used extensively for real-time media transmission over IP networks, but unfortunately data encryption is not built-in to RTP. Instead, the Secure Real-time Transport Protocol (SRTP) was developed after-the-fact to provide encryption and other security protections in both unicast and multicast applications [17]. Symmetric key cryptography techniques, such as AES, are commonly used to provide multimedia encryption on the data stream itself, but the key exchange procedure with this kind of scheme has been problematic due to ongoing implementation issues in early SRTP solutions that resulted in many solutions using non-standard negotiation and key management techniques [14, 23].

In this paper, we propose that asymmetric key cryptography may be a more viable option. We have thus implemented an asymmetric key cryptographic transform in RTP/RTSP payload based on Elliptic Curve Cryptography (ECC) in real-time video streaming applications. In our scenario, the server uses the receiver's public key to encrypt the video streams to send those streams to the client, who then decrypts the encrypted media using its private key. Careful consideration has to be made in identifying an optimal encryption method that would protect the multimedia data while still ensuring real-time streaming of videos, as end-users would reject any solution that degrades the quality of the content or adds significant delay to the process through high computational costs and overheads. In our implementation, we have tested and analyzed the performance of 18 ECC curves and compared these results with the performance of non-encrypted as well as AES-encrypted streams. Based on our results, we suggest some suitable ECC curves for the encryption of real-time video streaming.

II. Related Work

The use of ECC to encrypt the real-time video is not new. Spanos and Maples in 1995 proposed an approach to video encryption. Since most compression algorithms are lossy in nature, they selected the most sensitive portion of the compressed video stream to reduce the amount of encryption [2]. Bhargava et al. proposed four MPEG video encryption algorithms based on secret key which could be used to secure different video applications like video-on-demand, video conferencing etc. The encryption was done by randomly changing the sign bits of Discrete Cosine Transform coefficients as well as by changing the motion vector's sign bits. They applied the inverse Discrete Cosine Transform method for encryption/decryption at the time of MPEG video compression/decompression [3]. Tawalbeh et al. have proposed two ECC-based encryption algorithms. One is selective

978-1-5386-7252-5/18/$31.00 ©2018 IEEE
DOI 10.1109/NCS.2018.00015

encryption of the quantized DCT coefficients and the second one is perceptual encryption based on selective bit-plane encryption. They have used $E_{53330939}(2, 7)$ curve for encrypting the multimedia stream. They have done selective encryption of a portion of the multimedia data using ECC [4]. Feily et al. have done a comparative study of performance of both ECC based encryption and symmetric key encryption based on Blowfish cipher. They have tested the performance of different encryption techniques on a commercial application named MCS. In their work, they have encrypted the entire compressed video stream using both ECC as well as symmetric key encryption [5]. Bhandari and Wadhe have used ECC based encryption in a different way to encrypt real-time video. They have proposed a real-time video encryption algorithm using ECC and RC5 algorithm, but ECC is not used for encrypting the video stream directly. They have generated RC5 128-bit key for actual video encryption. ECC public/private key pair is generated to encrypt the RC5 key only. During the key exchange phase, the sender encrypts the RC5 key using the receiver's public key and sends that encrypted key to the receiver. The receiver decrypts the RC5 key using its ECC private key [6]. A similar kind of approach can be found in [7, 8]. Apart from these works, many works are done to encrypt images using ECC [9-12]. Shah and Saxena analyzed the performance of various encryption algorithms based on some parameters. They categorized the algorithms into fully layered encryption, permutation based encryption, selective encryption and perceptual encryption. The parameters are visual degradation, encryption ratio, speed, compression friendliness, format compliance and cryptographic security. They found that fully layered or naïve algorithm cannot be used for real-time media encryption because it is slow. Though permutation based algorithms are faster, they lack sufficient level of security. Selective encryption algorithms' speed varies depending on the number of parameter used. Finally, perceptual encryption algorithms are suitable for low quality video. According to them, these four categories of algorithms are not suitable for high security. No single algorithm can satisfy all parameters [18].

Most of the related works partially encrypted the video frames, but we have encrypted the entire video frames.

III. Motivation

Continuous technological advancements are making multimedia streaming a part of our daily lives and until recently, most video streaming was done in the clear, completely unencrypted. Now, with a nearly constant stream of news detailing more and more sophisticated network attacks, businesses such as Netflix are looking to encrypt customer streams [19-21]. There are also a lot of other players in the online movie on-demand streaming services, such as Hulu, Amazon Instant Video, and Vudu. The motivation for this work is to make real-time video streaming more secure. During online video transmission, an eavesdropper can intercept the video packets for personal use or distribution in the black market. Either way, this kind of illegal packet interception can mean huge financial losses for these companies as well as a threat to digital rights protection.

The Real Time Message Protocol Encrypted (RTMPE) and Real Time Message Protocol over TLS/SSL connection (RTMPS) proprietary protocols used in streaming videos were originally developed by Macromedia and are now owned by Adobe. RTMPE generates RC4 keys for video encryption, which can be broken [13]. Like VoIP media encryption, the symmetric key encryption technique AES can be used in this case, but to protect the AES key during the key exchange phase requires that the key itself be encrypted; otherwise the key can be intercepted easily, rendering the cipher insecure. The Diffie-Hellman key exchange technique can be used in this context, but this can also be compromised [14]. There are many asymmetric key encryption techniques, but we have used ECC based encryption technique here, because it is more efficient in terms of key size and computation time compared to other approaches like RSA [25].

IV. Problem Statement

Our present work is based on i) analyzing the performance of various ECC curves for the real-time video encryption, and ii) suggesting a suitable ECC curve for optimal results. Given the importance of having a seamless implementation with efficient video transmission in a real-time environment where consumers expect minimal if no latency at all, we also compare our ECC-based results with non-encrypted as well as AES-encrypted video streaming to ensure that significant complexity and overhead is not added.

V. Architecture

The experiment is done using client-server architecture. The client and server are connected through our institutional network. We have used Ubuntu 16.04 (64 bits) operating system in both server and the client. The code is written in Java, and Bouncy castle (an open-source lightweight cryptography API for Java) is used to implement the encryption and decryption processes. MJPEG video codec is used for our experiment. The clip duration is 20 seconds.

Server configuration:

i. Intel 3.40 GHz Core i7-3770 processor
ii. 16 GB Main memory

Client configuration:

i. Intel 2.50 GHz Core i5-2450M processor
ii. 6 GB Main memory

VI. Network Topology

Our experimental setup consists of one server and one client who are connected to each other through our institutional network. The server is connected to the network through ethernet connection, and the client is connected to the network through the institutional Wi-Fi.

Fig. 1. Experimental setup diagram

TABLE I. TYPES OF ECC CURVES USED

Sl. No.	Curve	Key size (in bits)
X9.62 prime curves:		
1	prime256v1	256
X9.62 binary curves:		
2	c2pnb272w1	272
3	c2pnb368w1	368
4	c2tnb431r1	431
NIST prime curves:		
5	secp256k1	256
6	secp256r1	256
7	secp384r1	384
8	secp521r1	521
NIST binary curves:		
9	sect283k1	283
10	sect283r1	283
11	sect409k1	409
12	sect409r1	409
13	sect571k1	571
14	sect571r1	571
Teletrust Prime curves:		
15	brainpoolp256r1	256
16	brainpoolp320r1	320
17	brainpoolp384r1	384
18	brainpoolp512t1	512

VII. METHODOLOGY

We based our work on the Java based real-time video streaming application from Github [15], and then incorporated out ECC-based encryption code into this platform. Our application consists of a server and a client where the client makes the initial request for a video. The server then encrypts the video file frame-by-frame and sends it to the client. The Real Time Streaming Protocol (RTSP) is used to play the streaming media at the client end after decryption. The encrypted video stream is sent to the client as a Real-time Transport Protocol (RTP) payload. UDP is used as transport layer protocol. Figure 1 depicts the block diagram of our experimental setup.

VIII. RESULT ANALYSIS

In this work, we have tested the video encryption using 18 different ECC curves. The used curves are given in table 1. We have used 3 types of curves namely X9.62 curves (Public key cryptography standard used for the financial services industries), NIST recommended curves, and the Brainpool curves developed by Teletrust. For the first two types of curves, we have used both prime as well as binary curves. All the curves have key size more than 224 bits, because NIST recommends the key of size more than or equal to 224 bits for better security [16].

We have measured the performance of the curves based on the end-to-end delay in milliseconds, the jitter in milliseconds, the packet loss, encryption and decryption time, and data rate. In our experiment, we didn't find any packet loss problem in any ECC curves. In this section, we will discuss the performance

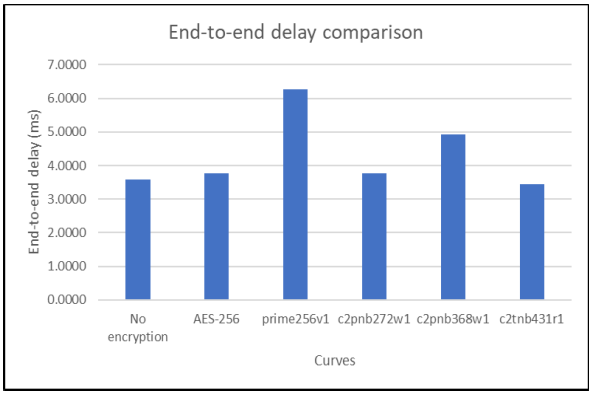

Fig. 2. End-to-end delay comparison between different X9.62 curves. Three binary curves' (272-bit, 368-bit and 431-bit) end-to-end delays are less than the 256-bit prime curve, and 272-bit curve has lowest delay among them.

of the ECC curves based on the end-to-end delay, and jitter compare to the non-encrypted video stream and the encrypted video stream with 256-bit AES encryption algorithm. We will also analyze their performance based on encryption/ decryption time and data rate. In our experiment, we have considered the network delay as the end-to-end delay.

In Fig. 2, we can see the end-to-end delay comparison between four X9.62 ECC curves with AES encryption and non-encrypted video stream. The delay of 256-bit curve is longest, but the delay of 272-bit curve is shortest which is nearly equal to AES-encrypted and non-encrypted stream. So, X9.62 272-bit curve can be suitable for video encryption among other X9.62 curves.

Fig. 3 depicts the comparison of jitter of four X9.62 curves and AES-encrypted and non-encrypted stream. The jitter of 256 and 272-bit curves are high, but compare to non-encrypted and AES-encrypted video stream, these are very negligible, i.e. nearly 0.06 ms. So, we can ignore this small difference. The percentage difference between the smallest (AES) and the highest (256 & 272-bit) jitter is only 0.14 which is very small.

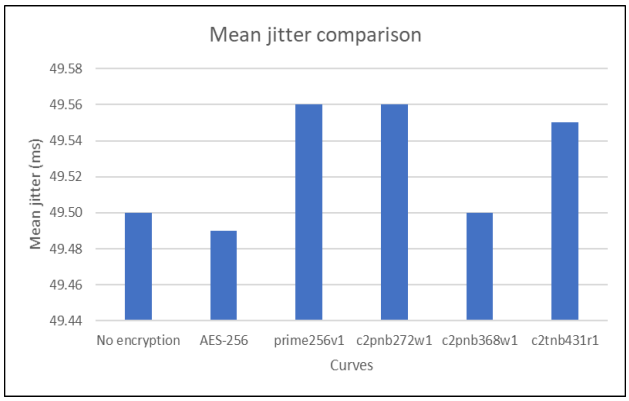

Fig. 3. Jitter comparison between different X9.62 curves. 368-bit curve has smallest jitter, and 256-bit and 272-bit curves have highest jitter than other curves. But overall jitter difference is nearly 0.06 ms which is negligible.

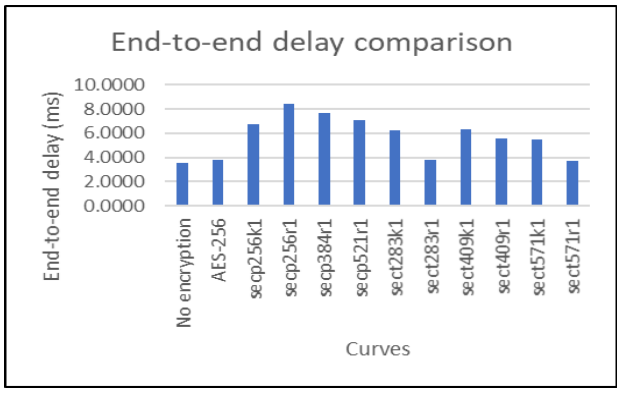

Fig. 4. End-to-end delay comparison between different NIST curves. 283-bit and 571-bit random curves have the shortest delay compare to other curves. 256-bit random curve has longest delay.

Fig. 4 depicts the comparison of NIST recommended ECC curves. Though NIST generally recommends prime curves over binary curves, in our experiment we have analyzed the performance of both type of curves for real time video encryption. We can see that the end-to-end delay is shortest in 283-bit and 571-bit random curves which are binary curves. Among prime curves, 256-bit koblitz curve has the shortest delay. So, NIST recommended 283-bit curve can be considered in this context among other NIST curves, because 571-bit curve has big key size which usually takes more time to encrypt and decrypt the media.

Fig. 5 depicts the jitter comparison of NIST curves. Like X9.62 curves, here also the jitter difference among the curves are negligible, only 0.07 ms. So, we may not consider this difference.

Now we will analyze the performance of the third category of ECC curves, i.e. Brainpool curves. Fig. 6 depicts the end-to-end delay comparison of these curves. The 512-bit curve has

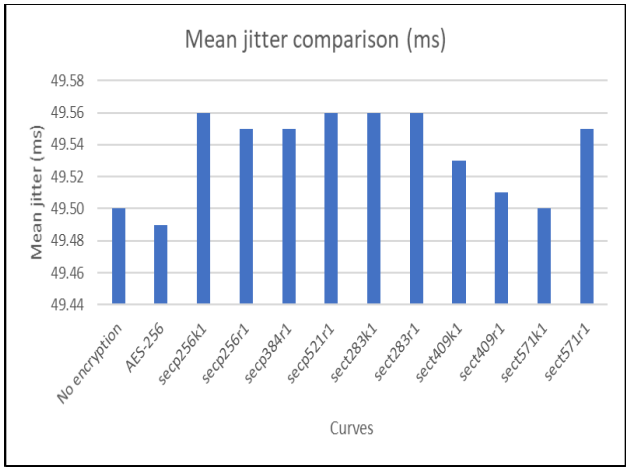

Fig. 5. Jitter comparison between different NIST curves. 409-bit random and 571-bit koblitz curves have smaller jitter, and 256-bit, 283-bit and 521-bit curves have higher jitter than other curves. Overall jitter difference is 0.07 ms which is very small.

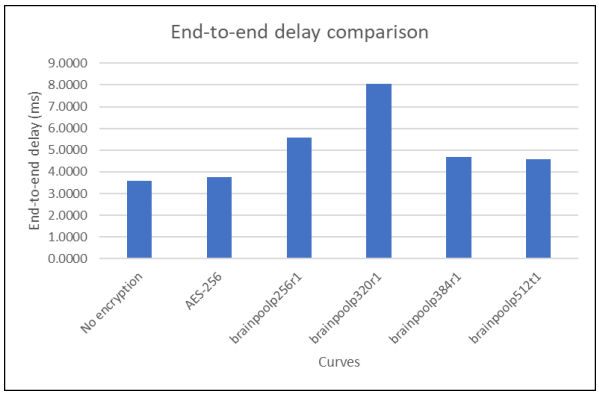

Fig. 6. End-to-end delay comparison between different Brainpool curve. 384-bit and 512-bit curves have shortest end-to-end delay. 320-bit curve has longest delay among other curves.

the shortest end-to-end delay, but 256-bit and 384-bit curves also have short delays, and the difference is nearly 1 ms. So, we can consider 256 or 384-bit curves among other brainpool curves for the real-time video encryption as far as delay is concerned, but not 512-bit curve because of its large key size.

Fig. 7 depicts the jitter comparison of Brainpool curves. Like X9.62 and NIST curves, the jitter differences are negligible in these curves. The maximum difference here is 0.08 ms. So, like previous two categories, we can ignore this difference as well.

From our experiment, we have seen that there are not much differences in jitter among the 18 curves as well as the AES-encrypted and non-encrypted video stream. So, we can conclude that this metric does not affect much in the quality of real-time video if it is encrypted with any of the above mentioned ECC curves.

So far, we have seen the category-wise performance of different ECC curves in terms of end-to-end delay and jitter. Now we will see the comparison of the performance of 18 curves all-together in terms of end-to-end delay. Fig. 12 depicts the end-to-end delay of all curves. X9.62 272-bit and 431-bit curves and NIST recommended 283-bit and 571-bit random curves have nearly same and shortest delay among other

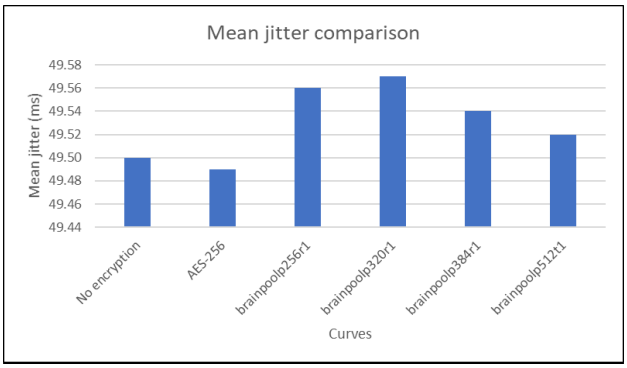

Fig. 7. Jitter comparison between different Brainpool curves. 512-bit curve has lowest jitter, and 256-bit and 320-bit curves have high jitter compare to other curves. But here also the overall jitter difference is 0.08 ms which is negligible.

curves. X9.62 256-bit curve, NIST recommended 256-bit, 283-bit, 409-bit and 572-bit koblitz curves, and Brainpool 256-bit, 384-bit and 512-bit curves have little longer delay compare to the previous set, but the difference is nearly 2.5 ms. The ECC random curves have long delay in this context. So, after analyzing all curves, we can conclude that X9.62 272-bit and 431-bit curves and NIST recommended 283-bit and 571-bit random curves performs well in the context of end-to-end delay.

After analyzing the performance of 18 curves in terms of end-to-end delay and jitter, now we will analyze the performance of the curves in terms of their encryption and decryption time. Fig 13 and 14 depict the encryption and decryption time comparison between 18 curves as well as that of AES encryption. We know that AES is symmetric key encryption algorithm, which always performs faster than any asymmetric key encryption as far as encryption and decryption are concerned. But the encryption time of X9.62 and NIST 256-bit curves is nearly 3 ms which is quite reasonable.

In our experiment, the size of the video frames is between 6500 bytes and 14000 bytes. From Fig. 13, we can see that X9.62 256-bit curve and NIST recommended 256-bit prime curves (both koblitz and random) have smaller encryption time compare to other ECC curves. Though AES has smallest encryption time, the difference between AES and these curves is nearly 3 ms, which is negligible. There are 5 more curves which have lesser encryption time between 6 ms and 8 ms. X9.62 272-bit curve and NIST recommended 384-bit prime curve have encryption time less than 6.5 ms which can be considered as less. As the key size increases, the encryption time increases as well. NIST recommended prime curves perform better than binary curves in this context. Most of the Brainpool curves take more time for encryption. The reason behind that, Brainpool curves use pseudo random primes [22]. On the other hand, NIST recommended curves use quasi-Mersenne primes [23]. So, Brainpool curves are slower compare to other curves.

In case of decryption time (Fig. 14), all the curves behave similarly like encryption time plot. Here also, the differences between AES and X9.62 256-bit curve and NIST recommended 256-bit curves are negligible, only less than 1.15 ms. Five curves have decryption time between 2 ms and 3 ms. Brainpool 256-bit curve falls under this category, but its encryption time is nearly 10 ms. So, we should not consider this curve. X9.62 272-bit curve and NIST recommended 384-bit curve have lesser decryption time also.

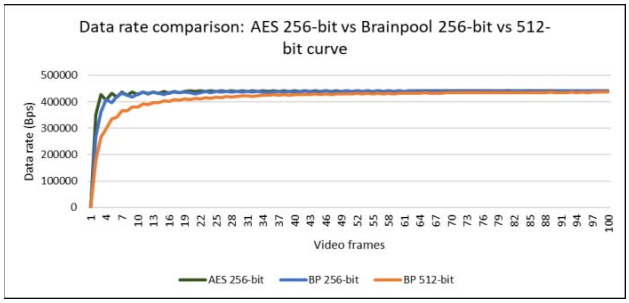

Fig. 8. Data rate comparison between AES and two Brainpool curves. Data rate of encrypted video stream with smaller key size is more than that of bigger key size for first 40-45 frames. So, video quality of first few frames will be better where smaller keys are used for encryption.

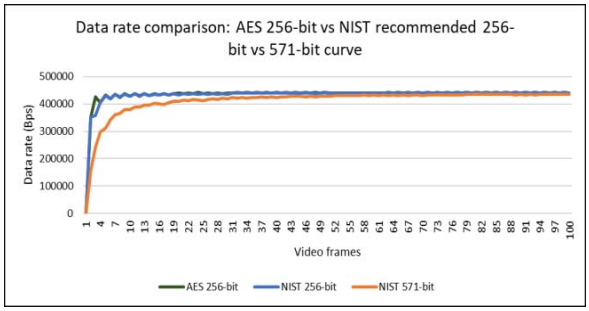

Fig. 9. Data rate comparison between AES and two NIST recommended curves. Data rate of encrypted video stream with 256-bit curve is more than that of 571-bit key for first 50-55 frames. So, video quality of first few frames will be better where smaller keys are used for encryption.

Finally, we will analyze the performance of the curves in terms of video data rate. The video data rate is the number of bits that are processed per unit time. It is related to the quality of the video. If all other factors (like codec) remain same, then high data rate means better quality of video. In our application, it is measured in bytes per second. Fig. 8-10 depict the graph of three categories of ECC curves. We have taken the curves of the smallest key size and the largest key size in each category into consideration. We have compared the performance of each set of ECC curves with AES encryption algorithm also. We have also considered the first 100 frames, because we have observed that the data rates of the curves in each category become nearly the same after first few encrypted frame transmissions.

Late breaking news: One day before the submission, we have found some interesting results that we want to share. From our result, we have found that the video frames encrypted with ECC curves of smaller key size have a higher data rate at the beginning. This has been observed in all three categories of ECC curves for the first few encrypted video frames. We have seen that the initial data rates of the video frames encrypted with smaller and larger key elliptic curves, are different in first 40-45 encrypted frames using Brainpool curves, in first 50-55 encrypted frames using NIST curves, and in 18-20 encrypted frames using X9.62 curves respectively. The initial data rate decreases with the increase in key size. In case of X9.62 curves (Fig. 10), the initial data rate difference is not so large compare to other two types of curves, because here the key difference is small (256-bit and 431-bit, compared

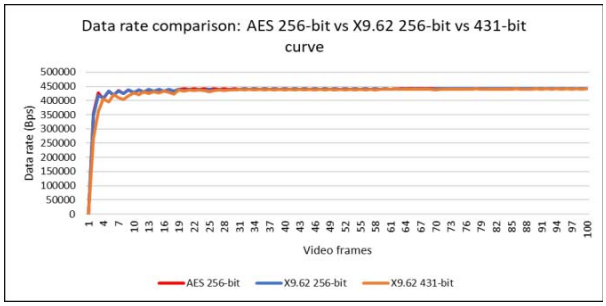

Fig. 10. Data rate comparison between AES and two X9.62 curves. Data rate of encrypted video stream with 256-bit curve is more than that of 431-bit curve for first 18-20 frames. The data rate difference is small compared to previous two, because the key size difference is less than the previous two cases.

to 256-bit and 571-bit in Brainpool and NIST curves). We have also observed that there is little or almost no difference in data rate between AES and 256-bit ECC curves. So, we can say that ECC curves with smaller key size perform better in context of initial video data rate at the beginning of the video clip. Since this is a very interesting finding, we need to do further research on this issue.

From the various measurements, we have analyzed that X9.62 272-bit and 431-bit curves, and NIST 283-bit and 571-bit random curves have lowest end-to-end delay. Apart from that, X9.62 256-bit curve, NIST 256-bit, 283-bit, 409-bit and 572-bit koblitz curves, and Brainpool 256-bit, 384-bit and 512-bit curves have little more delay compare to the previous set. But in terms of encryption and decryption time, X9.62 256-bit and 272-bit curves and NIST 256-bit prime curves (both random and koblitz), and 384-bit curve perform better. We have also observed that the initial data rate for first few encrypted frames is higher if we encrypt them with smaller keys. So, we conclude that X9.62 272-bit curve and NIST 256-bit prime curves (both random and koblitz) can be considered as suitable for real-time video encryption.

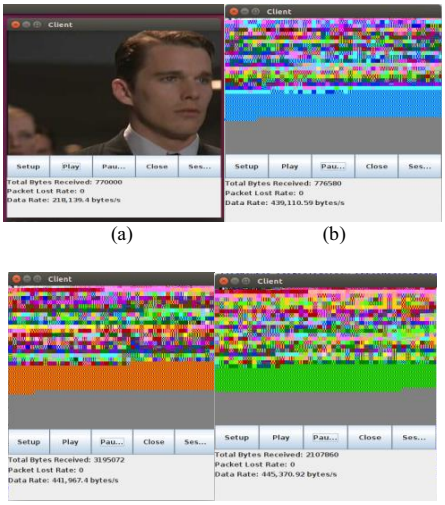

Fig. 11. (a) Video is played at client side after decryption, (b) Video encrypted with Brainpool curve, (c) Video encrypted with X9.62 curve, (d) Video encrypted with NIST recommended curve

IX. Conclusion

In this work, we have analyzed the performance of encrypted real-time video stream using 18 ECC curves. We have considered two metrics viz. end-to-end delay and jitter since no packet loss is found in the entire experiment. We have considered the encryption and decryption time of all curves. We have also observed that initial video data rate is high if we encrypt the video stream using ECC curves with smaller key size. After considering all aspects, we have concluded that X9.62 272-bit binary curve and NIST recommended 256-bit prime curves (both random and koblitz) can be considered as suitable for real-time video encryption. The jitters of all curves are nearly equal and their differences are negligible compare to AES-encrypted and non-encrypted video stream. Unlike other related works that encrypted the video frames partially, we have encrypted the entire video frame using ECC.

In this work, we have encrypted MJPEG video file for our experiment. In future, we will test the same using different types of video file formats that are supported by RTP. We have run this experiment in the institutional network. In the future, we will use a setup where the client and server will be connected to each other through Internet. Last but not the least, we will work further on the video data rate issue to find the relation between the key size and the data rate.

Acknowledgment

We would like to express our profound and deep sense of gratitude to Mr. Logan Widick and other students of Network Security Lab for their invaluable help without which our work would not have been successful. This research was partially supported by NSF grants CNS1229700, CNS1637291, and IIS1545599.

Appendix

The screenshots of the client playing the streaming video after decryption, and encrypted streaming video before decryption are given above. There is no significance of the blue, orange and green colors below the screen of encrypted video. These colors have appeared periodically in encrypted video screen irrespective of the ECC curves.

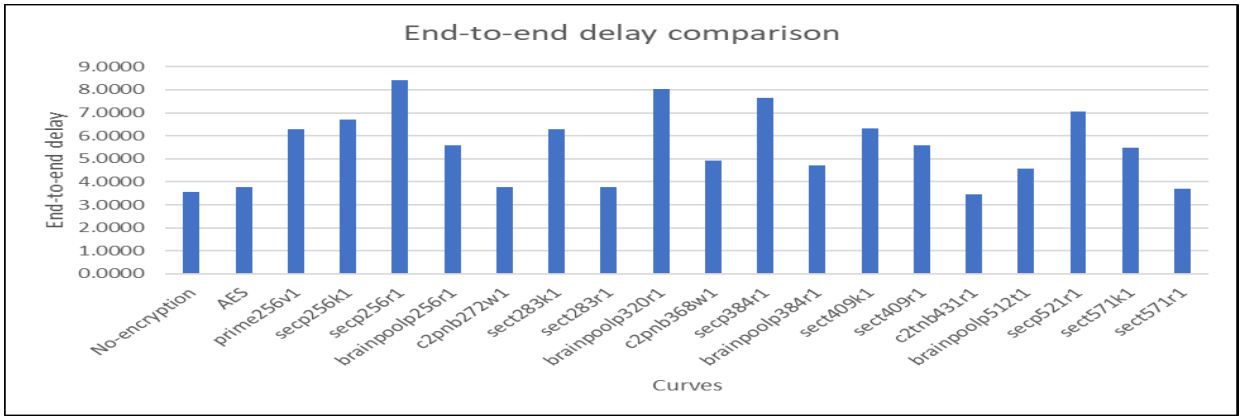

Fig. 12. End-to-end delay comparison between 18 ECC curves. X9.62 272-bit and 431-bit, and NIST recommended 283-bit and 571-bit curves have lower end-to-end delay.

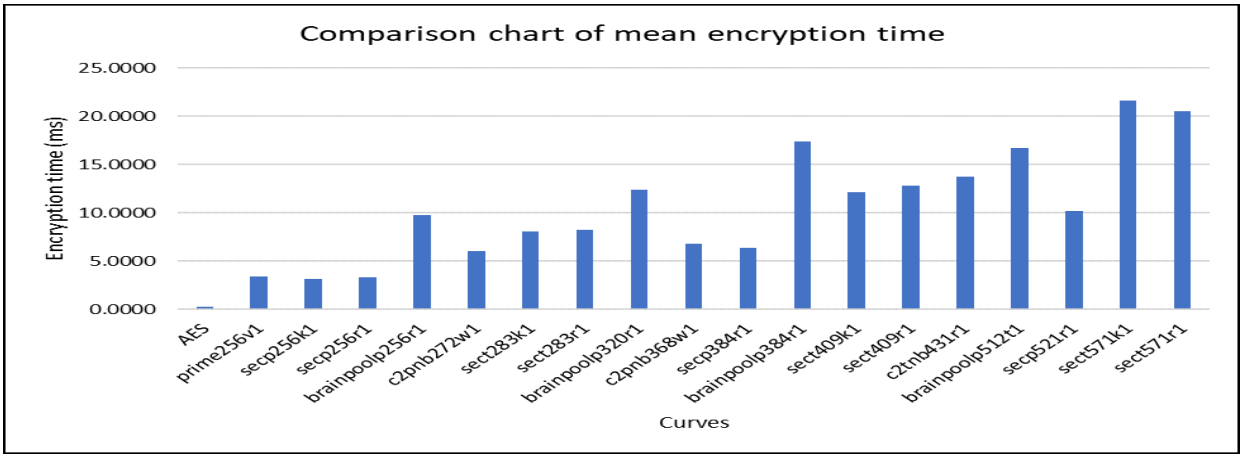

Fig. 13. Encryption time comparison between 18 ECC curves. X9.62 256-bit curve and NIST recommended 256-bit curves have small encryption time. X9.62 272-bit curve and NIST 283-bit and 384-bit curves also have small encryption time. Brainpool curves have large encryption time compare to their counterparts in terms of key length.

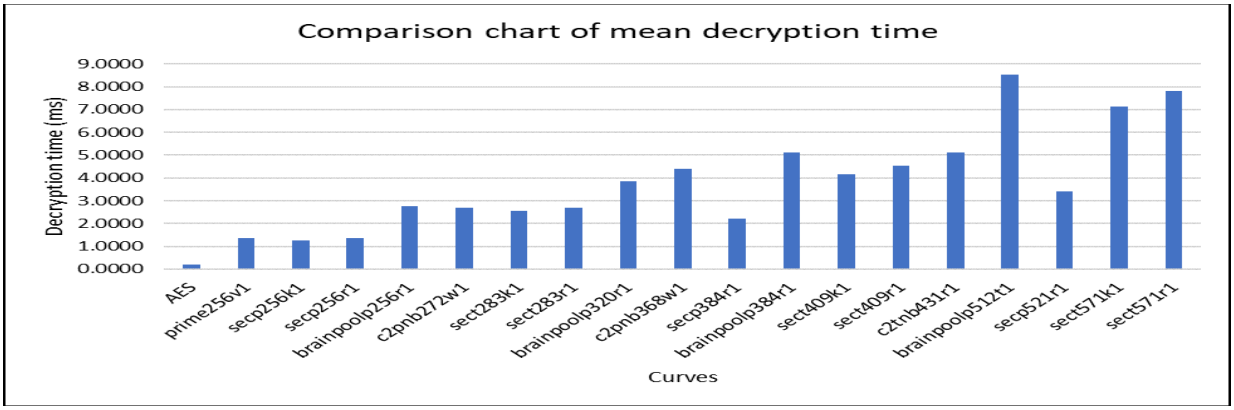

Fig. 14. Decryption time comparison between 18 ECC curves. X9.62 256-bit curve and NIST recommended 256-bit curves have small decryption time. X9.62 272-bit curve and NIST 283-bit and 384-bit curves also have small decryption time. Like previous plot, Brainpool curves have large decryption time compare to their counterparts in terms of key length.

REFERENCES

[1] D. Mortensen, "THE LIVE STREAMING VIDEO REPORT: Forecasts, emerging players, and key trends for brands' and publishers' next big opportunity", http://www.businessinsider.com/the-live-streaming-video-report-forecasts-emerging-players-and-key-trends-for-brands-and-publishers-next-big-opportunity-2016-8

[2] G.A. Spanos, and T.B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video", Fourth International Conference on Computer Communications and Networks, IEEE, September 1995, DOI: 10.1109/ICCCN.1995.540095

[3] B. Bhargava, C. Sh, and S. Wang, "MPEG Video Encryption Algorithm", International Journal of Multimedia Tools and Application, Vol. 24, September 2004, pp. 57-79.

[4] L. Tawalbeh, M. Mowafi, and W. Aljoby, "Use of elliptic curve cryptography for multimedia encryption", IET Information Security, Vol. 7, June 2013, pp. 67-74.

[5] M. Feily, S., Noori, and S. Ramadass, "On the performance of symmetrical and asymmetrical encryption for real-time video conferencing system", International Journal of Computer Science and Information Security, 2010, Vol. 8(7), pp. 49–55.

[6] L. Bhandari, and A. Wadhe, "Speeding up Video Encryption using Elliptic Curve Cryptography", International Journal of Emerging Research in Management & Technology, Vol. 2(3), March 2013, pp. 24-29.

[7] K. Gupta, and S. Silakari, "Efficient hybrid image cryptosystem using ECC and chaotic map", International Journal of Computer Application, Vol. 29(3), 2011, pp. 1–13.

[8] Z. Zhao, and X. Zhang, "ECC-based image encryption using code computing", International Conference on Communication, Electronics and Automation Engineering, Advances in Intelligent Systems and Computing, Vol. 181, 2013, pp. 859–865.

[9] G. Zhu, and X. Zhang, "Mixed image element encryption system", Ninth International Conference for Young Computer Scientists, Hunan, November 2008, pp. 1595–1600.

[10] K. Gupta, S. Silakari, R. Gupta, and S.A. Khan, "An ethical way of image encryption using ECC", First International Conference on Computational Intelligence, Communication Systems and Networks, Indore, July 2009, pp. 342–345.

[11] K. Gupta, and S. Silakari, "Efficient image encryption using MRF and ECC", International Journal of Information Technology and Knowledge Management, 2009, Vol. 2(2), pp. 245–248.

[12] V.K. Yadav, A.K. Malviya, D.L. Gupta, S. Singh, and G. Chandra, "Public key cryptosystem technique elliptic curve cryptography with generator g for image encryption", International Journal of Computer Technology and Application, Vol. 3(1), 2012, pp. 298–302.

[13] N. AlFardan, R. Holloway, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C.N. Schuldt, "On the Security of RC4 in TLS", 22nd Usenix Security Symposium, pp. 305-320, August 2013.

[14] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thome, L. Valenta. B. VanderSloot, E. Wustrow, S. Zanella-Beguelin, and P. Zimmermann, "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice", ACM SIGSAC Conference on Computer and Communications Security, pp. 5–17, October 2015.

[15] https://github.com/mutaphore/RTSP-Client-Server

[16] E. Barker, and A. Roginsky, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A.

[17] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC - 3711, https://tools.ietf.org/html/rfc3711

[18] J. Shah, and V. Saxena, "Video Encryption: A Survey", International Journal of Computer Science Issues, Vol. 8(2), March 2011, pp. 525-534.

[19] Netflix Techblog, https://medium.com/netflix-techblog/protecting-netflix-viewing-privacy-at-scale-39c675d88f45

[20] "Netflix will start encrypting its streams to block prying eyes and internet providers", https://qz.com/384184/netflix-will-start-encrypting-its-streams-to-block-prying-eyes-and-internet-providers/

[21] D. Goodin, "It wasn't easy, but Netflix will soon use HTTPS to secure video streams", Ars Technica, April 4, 2015,

https://arstechnica.com/information-technology/2015/04/it-wasnt-easy-but-netflix-will-soon-use-https-to-secure-video-streams/.

[22] M. Lochter, and J. Merklevideo, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC - 5639, https://tools.ietf.org/html/rfc5639

[23] M. Adalier, and A. Teknik, "Efficient and Secure Elliptic Curve Cryptography Implementation of Curve P-256",

https://csrc.nist.gov/csrc/media/events/workshop-on-elliptic-curve-cryptography-standards/documents/papers/session6-adalier-mehmet.pdf

[24] P. Thomas, and J. Kanclirz Jr., "Practical VoIP Security", Syngress Publishing, 2006.

[25] J. Lopez, and R. Dahab, "An Overview of Elliptic Curve Cryptography", Technical Report,

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2771