Enhancing Security in Real-Time Audio and Video Communication using Modified RSA Diffie-Hellman Key Exchange and Monoalphabetic Substitution Algorithm

1st Mr. Mayav Antani
Department of Computer Engineering
D.J. Sanghvi College of Engineering
Mumbai, India
mayavantani@gmail.com

2nd Mr. Aryan Parekh
Department of Computer Engineering
D.J. Sanghvi College of Engineering
Mumbai, India
aryanparekh2412@gmail.com

3rd Dr. Narendra M. Shekokar
Department of Computer Engineering
D.J. Sanghvi College of Engineering
Mumbai, India
narendra.shekokar@djsce.ac.in

***Abstract*—Real-time audio and video communication over the internet is vulnerable to eavesdropping and unauthorized access, making it crucial to ensure secure communication. Therefore, there is a need for an encryption algorithm that is not only strong but also fast. This paper proposes a secure real-time audio and video communication scheme using a modified RSA Diffie-Hellman key exchange and monoalphabetic substitution algorithm. The proposed method generates a shared secret key between the sender and receiver using a modified RSA Diffie-Hellman key exchange algorithm. The shared secret key is then used as a seed value to create a substitution map. Lastly, communication security is enhanced by encrypting the audio and video streams using the monoalphabetic substitution algorithm. We evaluate the proposed video encryption scheme using performance metrics such as computation time, correlation coefficient, PSNR, and histogram deviation. Similarly, the audio encryption scheme is evaluated using performance metrics such as computation time, Shannon entropy, and statistical parameters. Our experimental results demonstrate that the proposed method provides a high level of security with minimal impact on communication performance. Furthermore, the proposed scheme outperforms existing schemes like AES, RC4, and ChaCha20 regarding security, encryption, and decryption times. The proposed scheme is suitable for real-time audio and video communication applications such as teleconferencing, telemedicine, and online gaming. Overall, our research highlights the efficacy of using a modified RSA Diffie-Hellman key exchange and monoalphabetic substitution algorithm to enhance the security of real-time audio and video communication.**

***Index Terms*—communication time, statistical parameters, correlation coefficient, histogram deviation, Shannon entropy, computation time, teleconferencing, telemedicine, online gaming, AES, RC4, ChaCha20, shared secret key, modified RSA Diffie-Hellman key exchange, monoalphabetic substitution algorithm, real-time, audio and video communication, PSNR, secure communication, communication performance.**

I. Introduction

Recognizing the need to improve the security of real-time video conversations without slowing down communication, a global survey of extant algorithms was conducted. Unfortunately, existing algorithms employ suboptimal encryption methods, resulting in unsatisfactory results. This research seeks to develop and evaluate a novel encryption technique that provides robust security while minimizing its impact on communication efficacy. The method's suitability for real-time audio and video communication applications will also be demonstrated.

An exhaustive survey of existing algorithms addressed the difficulties of implementing highly secure algorithms in real-time video communication. A method merging video and audio encryption using a mono-alphabetic substitution cipher was proposed using knowledge of cryptography. Although this encryption method is traditionally considered less secure for textual data due to its susceptibility to text prediction, it becomes advantageous for image encryption. This algorithm is suitable for rapid and secure processing due to the lack of discernible pixel-value patterns. In addition, a modified Diffie-Hellman key exchange mechanism was implemented to facilitate sharing of the encryption and decryption secret key.

The sender and receiver generate a shared secret key using a modified RSA Diffie-Hellman key exchange algorithm in the proposed method. This shared key is used as a seed to generate a substitution map, which is then used to encrypt audio and video streams utilizing the mono-alphabetic substitution algorithm. This project seeks to surmount the limitations of existing encryption algorithms to secure live audio and video communication, such as slow encryption/decryption processes, insufficient security measures, and high computational requirements.

The endeavor entails developing, implementing, and evaluating the proposed methodology and comparing its efficacy to that of established cryptographic techniques. In addition, this study intends to advance scientific knowledge by introducing this novel technique for securing real-time audio and video communication.

II. Review of Literature

[1]The paper discusses the need for secure information technology systems due to the presence of malicious actors who seek to cause harm by disrupting or stealing data. It proposes the use of encryption methods to protect against threats and risks, specifically using substitution or transposition principles. The program designed allows for enciphering and deciphering of information, while ensuring the integrity of the data. Measures such as access control and risk management are proposed as countermeasures to prevent security breaches. The paper also emphasizes the importance of training to improve risk management practices.

[2]The article discusses network security and using cryptography to safeguard data and communications against cybercrime. Symmetric encryption is defined as a type of encryption using a single key. For example, RSA is a form of symmetric key encryption that employs both public and private keys. Also mentioned is the Diffie-Hellman cryptography, in which both parties exchange secret keys to encrypt messages.

[3]The paper presents an algorithm for computing the secret alpha in the serious Diffie-Hellman problem and related problems, with computational complexity reduced by O($\sqrt{d}$) compared to the discrete logarithm problem, for an abelian group of prime order p. The algorithm requires the inputs g, (g^a), and (g^{d*a}) or (g^{i*a}) (for i = 0, 1, 2,..., d). This algorithm is also applied to the Diffie-Hellman problem on an abelian group of prime order p, reducing the complexity of recovering the secret key from O($\sqrt{p}$) to O($\sqrt{p}$/d) for specific schemes, such as Boldyreva's blind signature and the original ElGamal scheme, with specific requirements on the divisor d.

[4]This paper discusses the significance of image encryption for secure Internet transmission, particularly in real-time scenarios involving significant quantities of data. The article examines image encryption techniques in spatial, frequency, and hybrid domains, employing complete and selective encryption strategies.

[5]The document emphasizes the significance of data security in the evolution of information technology and the business world. Numerous techniques have been developed to enhance networks' dependability, including the Diffie-Hellman technique, which facilitates the exchange of a secret key between two users over an open communication channel. However, this algorithm is susceptible to attacks from eavesdroppers, and its security relies on a client-selected random prime number. In contrast, the RSA algorithm is typically employed for encryption and decryption and depends on the difficulty of locating the factors of large integers. The paper proposes a hybrid cryptography algorithm incorporating the Diffie-Hellman and RSA algorithms for securely exchanging messages using a secret shared key. This hybrid strategy modifies the value of N in the RSA algorithm to thwart mathematical attacks. The purpose of this paper is to enhance the security of network communications and the dependability of networks.

[6]The paper examines the data protection problem in digital communications and multimedia applications, emphasizing audio data security. It covers security analysis, computational complexity, and quality analysis of recent contributions to audio encryption algorithms utilizing chaotic map-based techniques. The paper contrasts the strengths and weaknesses of various audio encryption and decryption techniques based on chaotic maps, including digital and analog audio algorithms. It also discusses several proposed projects for audio encryption utilizing chaotic maps, emphasizing their sensitivity to initial conditions and quasi-random behavior.

[7]This paper proposes a real-time selective video encryption solution for the SHVC video coding standard. The proposed scheme encrypts sensitive SHVC parameters at the CABAC binstring level with minimal overheads in terms of latency and complexity. The encryption procedure satisfies constant bitrate and format-compliant video encryption requirements and maintains all SHVC capabilities, including bitstream extraction and error resilience. The performance of the proposed scheme is evaluated concerning various encryption criteria, scalability configurations, and HD video sequences. Comparisons are made between the proposed solution and three selective SHVC encryption schemes, and the processing complexity is assessed in the context of a real-time SHVC decoder. The proposed solution exhibits high security and minimal processing complexity, with a real-time decoding overhead of no more than 6 percent.

[8]This paper proposes a video encryption algorithm based on moving objects to assure the security and efficacy of real-time video transmission. The algorithm extracts moving objects and encrypts them with an algorithm based on entropy coding. The experimental results demonstrate the algorithm achieves favorable tradeoffs between security and coding efficiency while meeting real-time requirements.

[9]The article discusses the need for video encryption on various platforms. It introduces the Quasigroup Video Encryption Scheme (QuVench), a new encryption scheme that can accomplish near real-time encryption/decryption rates while maintaining a small footprint. The algorithm is appropriate for devices with limited resources, such as UAV and drone applications. The article describes the algorithm, hardware implementation, and security analysis of Quasigroup Video Encryption cipher (QuVench) extended to the compressed domain.

[10]The paper proposes a two-phase encryption algorithm for real-time videos employing the H.261 codec for compression/decompression and the chaotic logistic map

system for encryption/decryption operations. The algorithm encrypts intra-frames while leaving inter-frames unchanged to reduce processing time. In addition, a new strategy for selecting bytes for encryption is presented. Finally, tests based on statistics validate the algorithm's optimistic performance regarding security level and time required for encryption/decryption processes.

[11]This paper investigates the computational and decisional Diffie-Hellman problems and offers reductions between them in a high-granularity setting. It demonstrates that all three variants of the computational Diffie-Hellman problem are equivalent to optimal reduction. It also examines variants of the decisional Diffie-Hellman problem in single-sample and polynomial-sample contexts. It demonstrates that all variants are equivalent, except for the DDH SDDH argument, which remains an open problem.

III. Methodology

The following section describes the entire process and methods used to complete the real time cryptography algorithm, The section also decribes the modules used in order to complete the same.

A. Mathematical Modelling

Let the audiovisual file to be encrypted be X with n number of frames. First, X is split into its constituent parts, i.e., video file V and audio file A. Then V is split into its different frames, F_1, F_2, ..., and F_n. Along with this, two mappers are generated, video and audio mapper, with 256 and 16 values respectively.

Once all these keys and frames are generated, the encryption process of each frame F_i starts by replacing all the pixel values with the mapped values in the video mapper. On substituting these values, we get the encrypted frame f_i; then all the encrypted frames, i.e., f_1, f_2, ..., f_n, are merged to get encrypted video file V'. In a similar manner, the audio file is also encrypted by substituting the audio mapper values in the audio stream. This results in the formation of the encrypted audio file A'. Finally, the encrypted video file V' and the encrypted audio file A' are joined to generate the encrypted audiovisual file X'.

$X = V + A$
$V = F_1 + F_2 + F_3 + \ldots + F_n$
Generate video and audio mapper
F'_i –> F_i *(substitute pixel values in video mapper)*
A' –> A *(substitute values from audio mapper)*
$V' = F'_1 + F'_2 + F'_3 + \ldots + F'_n$
$X' = V' + A'$

As for the decryption, the encrypted audiovisual file is split into video and audio files. The Video file is then split further into frames. Then, the reverse video and reverse audio mappers are created using the seed vlue and reversing the video and audio mappers respectively. Each frame is then decrypted by substituting the reverse maper values with the pixels in the frame. This process generates the decrypted video frame. At the same time, the audio file is decrypted with the help of the reverse audio mapper by substituting the values. On decrypting the frames, they are combined to form video V; and audio file A is received on decrypting the audio file. Finally, V and A are combined to regain the original decrypted audiovisual file. The mathematical form of the same is given below.

$X' = V' + A'$
$V = F'_1 + F'_2 + F'_3 + \ldots + F'_n$
Generate reverse video and reverse audio mapper
F_i –> F'_i *(substitute pixel values from reverse video mapper)*
A –> A' *(substitute values from reverse audio mapper)*
$V = F_1 + F_2 + F_3 + \ldots + F_n$
$X = V + A$

B. Key Exchange

1) Explanation for need for modification in Diffie Hellman: Fig. 1 represents the Diffie Hellman algorithm. It shows how the process of key exchange takes place between User 1 and User 2. However, this method is susceptible to an attack called a 'Man-In-The-Middle' attack.

Fig. 2 represents how a man-in-the-middle attack can occur in Diffie Hellman algorithm. Hacker can intercept the message and change the shared secret key for both sender and receiver, which can be very dangerous because, if needed, the hacker can even give an illusion to the sender and receiver that the communication between them is not intercepted and continue to extract data from both of them until he wishes to do so.

In order to overcome the issue of 'Man-In-The-Middle' in Diffie Hellman algorithm, modifications have been made in it as explained below and represented in Fig. 3.

The Fig. 3 shows the process of key transfer between the sender (User 1) and to receiver (User 2). This is a modified Diffie-Hellman key exchange protocol; two known prime numbers are combined with the private keys of the sender and receiver to generate public keys for each. These public keys' RSA-encrypted SHA-1 hash value is then concatenated with an additional value. The sender then transmits these concatenated values to the receiver and vice versa. The concatenated values are then separated, and the encrypted hash value is decrypted before the public key is hashed using SHA-1. If these two values match, we can conclude that a man-in-the-middle attack did not occur.

C. Encryption

The Fig. 4 shows the process of encryption in the proposed methodology. The Plain Audiovisual stream is first divided into plain audio and video streams. Then, it is encrypted

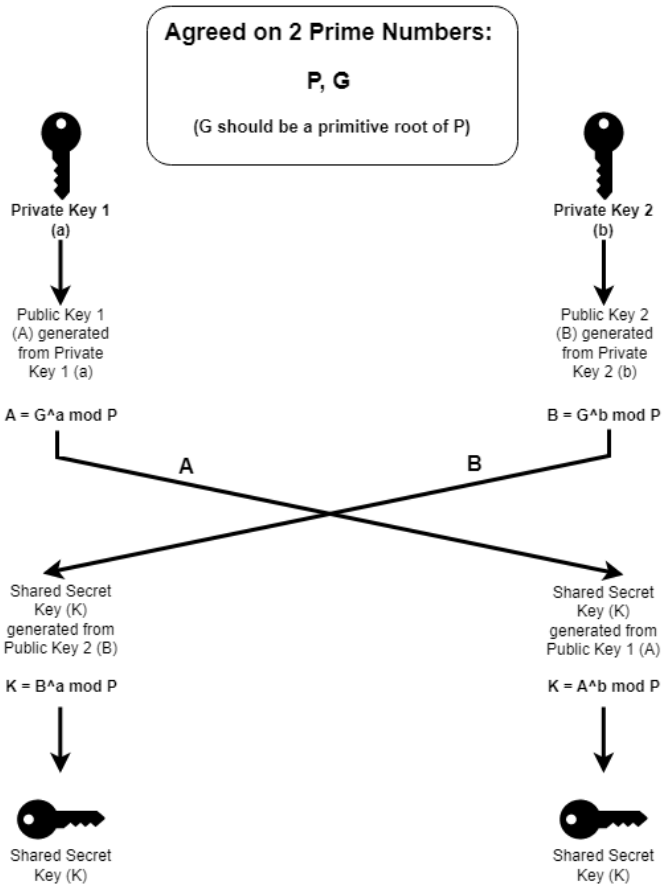

Fig. 1. Diffie Hellman Algorithm

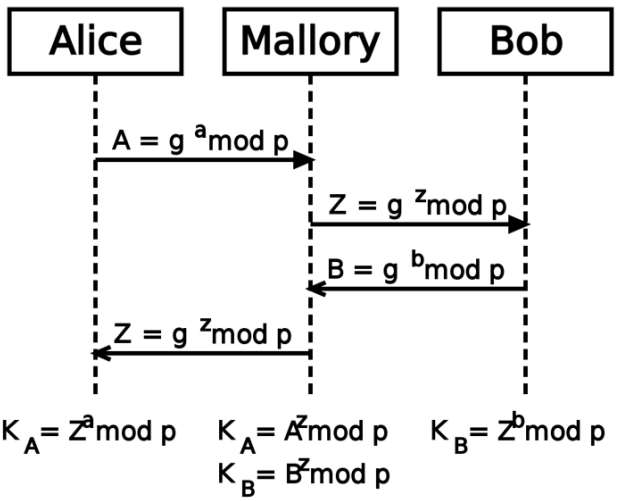

Fig. 2. Man In the Middle Attack

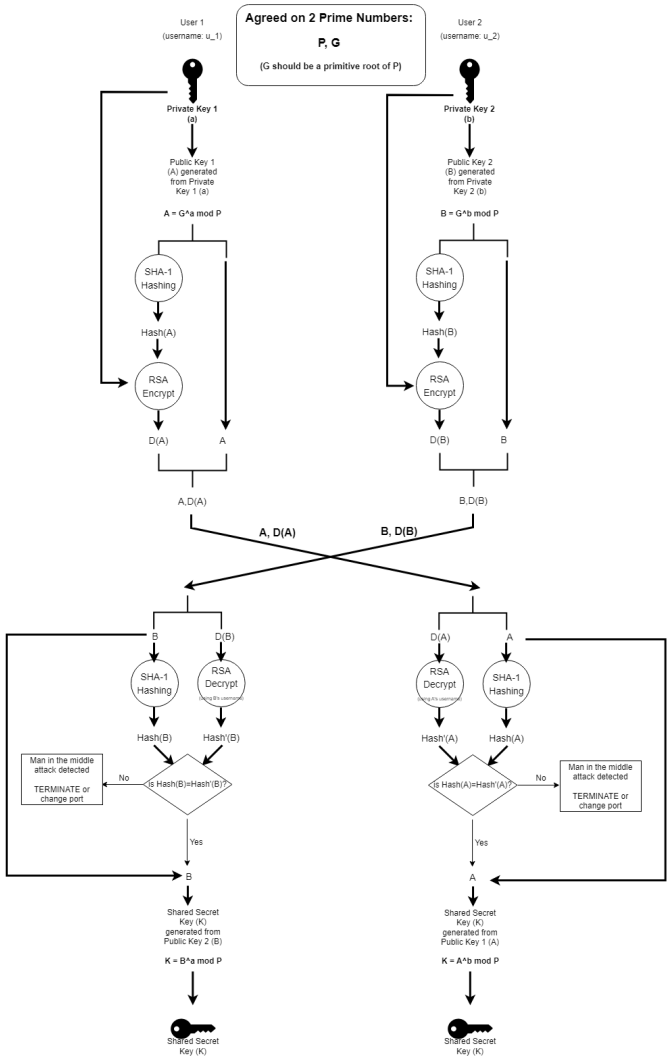

Fig. 3. Key Exchange Process

using the audio and video mappers, which were generated using a random key generator with seed value as the value of the shared secret key. The encryption is done in the following manner. First, the video stream is broken into its constituent parts, out of which each frame is taken. Then, the values in the frame are substituted with other values using the video mapper, converted into a cipher frame, and merged into the stream.

The audio file is also encrypted similarly to the video file, but the entire stream is encrypted by substituting hexadecimal values in the audio mapper.

D. Decryption

The Fig. 5 shows the process of decryption in the proposed methodology. The Cipher Audiovisual stream is divided into cipher audio and video streams. Then, it is decrypted using the reverse audio and reverse video mappers, respectively. They are generated by using a random key generator with

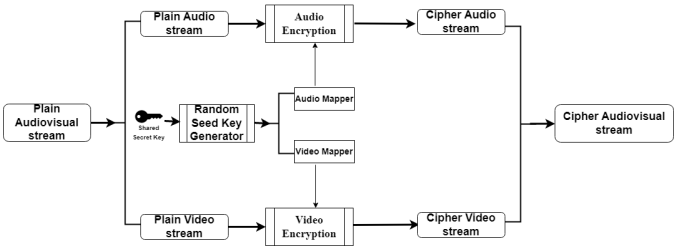

Fig. 4. Encryption Process

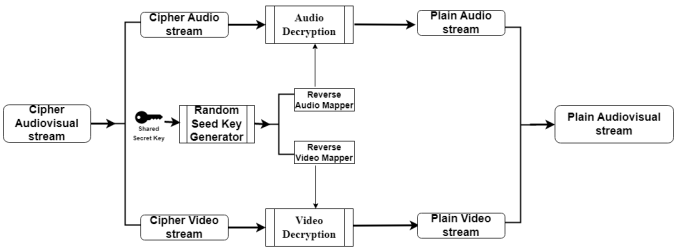

Fig. 5. Decryption Process

seed value as the value of the shared secret key and then reversing it. Now, the encrypted video stream is broken into its constituent parts, out of which each frame is taken. The values in the frame are substituted with other values in the reverse video mapper, converted into a decrypted frame, and merged into the stream.

The encrypted audio file is also decrypted similarly to the encrypted video file, but the entire stream is decrypted by substituting hexadecimal values in the reverse audio mapper.

E. Modules Used

The list of python modules used for the completion of the cryptography algorithm and making the application prototype to run the algorithm on are stated as follows:

- OpenCV: OpenCV (Open Source Computer Vision Library) is a free computer vision and machine learning software library. OpenCV was created to provide a standardized infrastructure for computer vision applications and expedite machine perception incorporation into commercial products. In addition, OpenCV's Apache 2 license makes it simple for businesses to utilize and modify the code.
- PyAutoGUI: PyAutoGUI is a Python package that provides the ability to simulate mouse cursor movements and actions as well as keyboard button inputs. It is compatible with Windows, MacOS X, and Linux. Used for capturing a screenshot of the screen at each instant and transmitting it via the screen share stream.
- Numpy: NumPy is an indispensable Python module for scientific computing. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and several routines for fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, introductory linear algebra, fundamental statistical operations, and random simulation, among others.
- Socket: This module provides access to the BSD socket interface. It is compatible with all modern Unix systems, Windows, MacOS, and likely additional platforms. The Python interface is an object-oriented translation of the Unix system call and library interface for sockets: the socket() function returns a socket object whose methods implement the numerous socket system calls. As with read() and write() operations on Python files, buffer allocation is automatic on receive operations, and buffer length is implicit on transmit operations.
- Pickle:Python's Pickle is predominantly utilized for serializing and deserializing object structures. In other words, it is converting a Python object into a byte stream to store it in a file/database, maintain a program state between sessions, or transport data over a network. Unpickling the pickled byte stream can recreate the original object hierarchy. This entire operation resembles object serialization in Java or.NET.
- Threading:Python's threading module includes an easy-to-implement synchronization mechanism that enables the synchronization of threads. In our case, threading is used to transmit audio and video simultaneously.
- PyAudio: PyAudio provides Python bindings for the cross-platform audio I/O library PortAudio v19. PyAudio makes it simple to play and record audio on multiple platforms, including GNU/Linux, Microsoft Windows, and Apple macOS. In this case, it is utilized to record audio and convert it to hexadecimal format.
- tkinter:The tkinter module ("Tk interface") is the standard Python interface to the Tcl/Tk graphical user interface toolkit. Tk and tkinter are available on the majority of Unix platforms, including macOS, as well as Windows operating systems. Tkinter is not a thin wrapper; instead, it adds a significant quantity of its logic to make the experience more Pythonic.
- Random: Python's built-in Random module is utilized to generate random integers. These numbers are not genuinely arbitrary, as they are pseudo-random. For example, this module can generate random integers, produce a random value for a list or string, and much more.

IV. Results and Discussions

A. Outputs

The following images are the depiction of the encrypted and decrypted output of the video at a given instant. This encryption is done for every frame during the entire video call. Fig. 6 and Fig 7 represent the encryption and decryption of a video call respectively.

Fig. 8 and Fig. 9 represent the encryption and decryption of a shared screen respectively.

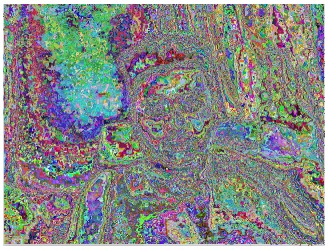

Fig. 6. Encrypted Frame (video call)

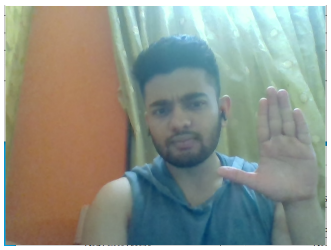

Fig. 7. Decrypted Frame (video call)

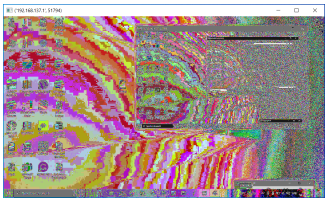

Fig. 8. Encrypted Frame (screen share)

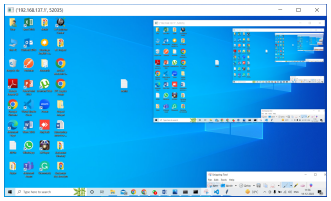

Fig. 9. Decrypted Frame (screen share)

B. Cryptanalysis

Let us assume we have a hacker who is trying to intercept the real-time communication between User 1 and User 2. In order to intercept the video call and make sense of the conversation taking place, the hacker has to decrypt the encrypted audio and video stream. Let us consider cryptanalysis for audio and video separately.

1) Video: In order to decrypt the video stream, the hacker has to decrypt each frame of the video. To do so, the hacker has to generate his own "Reverse video mapper". To generate the correct reverse video mapper, the hacker has to try every permutation and combination of numbers possible i.e., each possible substitution for numbers 0 to 255. Therefore, the number of attempts the hacker will have to make to get the correct reverse video mapper will be 256! ($8.578177753*10^{506}$)

2) Audio: In order to decrypt the audio stream, the hacker has to decrypt each hexadecimal value of the audio. To do so, the hacker has to generate his own "Reverse audio mapper". To generate the correct reverse audio mapper, the hacker has to try every permutation and combination of numbers possible, i.e., each possible substitution for numbers 0 to 9 and 'a' to 'f' (all hexadecimal values). Therefore, the number of attempts the hacker will have to make to get the correct reverse audio mapper will be: 16! ($2.092278988*10^{13}$)

These values indicate that it is computationally infeasible for the hacker to crack the values of keys and check each permutation. Therefore, it becomes next to impossible for the hacker to decipher the message that is being shared either in the audio, or in the video stream.

C. Evaluation

In order to test our algorithm, we need to first divide the project into two parts:

1) Video Cryptography
2) Audio Cryptography

The process of evaluation for Video Cryptography can be broken down ultimately to the evaluation of encryption of one single Frame of an image. Evaluation of cryptography on single image frame will be done based on the following parameters:

1) Computation time
2) Correlation Co-Efficient
3) Histogram Deviation
4) Peak-Signal-To-Noise (PSNR) Comparison

These parameters have been chosen after careful consideration and thorough research by the authours of the paper [4], and we intend to follow the same.

For evaluation of the Audio Cryptography, we will use the following parameters:

1) Computation time
2) Shannon Entropy
3) Statistical Parameters (Mean and Standard Deviation)

The results of the Video Cryptography tests can be seen in the Table I. All these tests are performed on multiple 1280*720 RGB images. A sample of the original images that were used for testing is displayed in Figure 10, Figure 13, and Figure 16. The encrypted images generated by our algorithm for these original images are displayed in Figure 11, Figure 14 and Figure 17. Finally, the decrypted images are represented by the Figure 12, Figure 15 and Figure 18.

Computation time here is the average of the encryption and decryption time and turns out to be just 0.0185 seconds (approximately). This is really fast when compared with other algorithms (discussed later).

The correlation coefficient is a quantitative measure used in statistics to assess the strength and direction of the linear relationship between two variables. Given that the average correlation coefficient between the original and encrypted image is just 0.0925 (approx), we can infer that there is hardly any correlation between the original and encrypted image, which states that the images are very different.

We can also see that the histogram deviation between the original and encrypted images is almost near 1, which means pixel distribution is also very different in both images.

A reduced PSNR value between the initial and encrypted image frames is preferred in evaluating image frame encryption algorithms. This suggests that the encrypted image frame exhibits a reduced level of fidelity relative to the unencrypted image frame, thereby rendering the image recovery task more arduous for a potential attacker. We have a value of 27.92 (approx), which is actually less.

The results of the Audio Cryptography tests can be seen in table II. All these tests are performed on multiple 1s 235KB Audio clip files.

TABLE I
VIDEO CRYPTOGRAPHY EVALUATION

Evaluation Parameters	Values
Computation time	0.02059 seconds
Correlation Co-Efficient (Original and Encrypted)	0.06298
Histogram deviation (Original and Encrypted)	0.99604
PSNR (Original and Encrypted)	27.87296

Fig. 10. Original Image 1

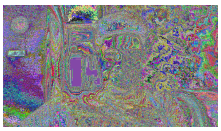

Fig. 11. Encrypted Image 1

Fig. 12. Decrypted Image 1

Computation time here is the average of the encryption and decryption time and turns out to be just 0.007413 seconds (approximately). This shows that the algorithm is really fast. Shannon entropy is a measure of the randomness of a signal or data stream. In our case, we get the Shannon entropy of the original and the encrypted algorithm the same up to the last decimal point. This means we are not introducing any additional unnecessary noise to the audio while encrypting it. We can also observe that statistical parameters like Mean and Standard deviation of the original and decrypted audio are exactly the same. This means that we are encrypting and transmitting the audio data with no compromise in the quality of the data.

D. Comparison

There is a necessity for a newly designed algorithm to be compared with the existing state-of-the-art algorithms. Therefore, we have chosen three other algorithms for comparison which are being used for image encryption for a long time. These are:

1) AES
2) RC4
3) ChaCha20

Fig. 13. Original Image 2

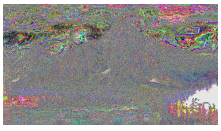

Fig. 14. Encrypted Image 2

Fig. 15. Decrypted Image 2

Fig. 16. Original Image 3

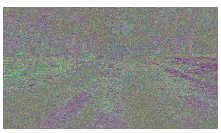

Fig. 17. Encrypted Image 3

Fig. 18. Decrypted Image 3

TABLE II
AUDIO CRYPTOGRAPHY EVALUATION

Evaluation Parameters	Values	
Computation Time	0.007413 seconds	
	Original	**Encrypted**
Shannon Entropy	6.84870	6.84870
Statistical Parameters	**Original**	**Decrypted**
Mean	-0.70	-0.70
Standard Deviation	3073.84	3073.84

Also, the standard parameters that we will be using for Comparison:

1) Mathematical Complexity
2) Encryption time
3) Decryption time

In AES, the image is divided into blocks of fixed size, typically 128 bits. The encryption algorithm then applies a series of transformations to each block, using a symmetric key to generate a ciphertext that is difficult to decrypt without the key. The transformations used in AES include substitution, permutation, and bitwise operations.
In RC4, the image is encrypted by applying a series of keystreams to the plaintext using a symmetric key. The keystreams are generated using a key-scheduling algorithm (KSA), which transforms the key into a permutation of all possible bytes. The keystream is then generated by a pseudo-random generation algorithm (PRGA), which combines the permutation with an initialization vector (IV) to generate a stream of pseudo-random bytes that are XORed with the plaintext to produce the ciphertext.
In ChaCha20, the image is encrypted by applying a series of transformations to the plaintext using a symmetric key. The transformations involve a series of XOR and addition operations that are applied to a 512-bit state, which is initialized using a key and a nonce. The result of each transformation is a new 512-bit state that is used in the next transformation.
Results for the comparison are displayed in Table III. We can clearly observe that our algorithm outperforms AES, RS4 and ChaCha20 not only in terms of computation time (average of Encryption and Decryption time) but also in terms of Mathematical Complexity (Note: 256! is greater than 2^{80}, 2^{256} and 2^{352})

TABLE III
COMPARISON ANALYSIS OF VARIOUS IMAGE ENCRYPTION ALGORITHMS

-	Our Algorithm	AES	RS4	ChaCha20
Encryption time (in seconds)	0.02084	0.02733	0.94050	0.02262
Decryption time (in seconds)	0.02033	0.03277	0.93103	0.02770
Mathematical complexity	256!	2^{256}	2^{80}	2^{352}

V. Conclusion

In this project, we employ our understanding of Information Security to ensure that the fundamental objectives of confidentiality, data integrity, and data accessibility are met. In addition, we create a prototype application that functions similarly to zoom or facetime with a high level of encryption to protect the privacy of the information shared between the two communicating parties.

We employ mono-alphabetic substitution cyphers with random key values generated from a seed value and the RSA Digital Signature modified Diffie-Hellman key exchange protocol to encrypt and decrypt the video and audio of the call in real-time without experiencing any latency. This enables the application's users to utilize it without worrying about being attacked or their data being compromised.

References

[1] G. I. Ighalo, "Using mono-alphabetic substitution to secure against threats and risk in information technology," *IOSR Journal of Electronics and Communication Engineering*, vol. 5, 2013.

[2] N. A. Lal, "A review of encryption algorithms-rsa and diffie-hellman," *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH*, vol. 6, 2017.

[3] J. H. Cheon, "Security analysis of the strong diffie-hellman problem," vol. 4004 LNCS, 2006.

[4] G. Subbiah, P. Punithavathi, A. Infanteena, and S. Sindhu, *A Literature Review on Image Encryption Techniques*, 01 2021, pp. 2091–2136.

[5] A. Bhattacharjee, C. Khaskel, D. Basu, and P. M. D. R. Vincent, "Hybrid security approachby combining diffie-hellman and rsa algorithms," *International Journal of Pharmacy and Technology*, vol. 8, 2016.

[6] "A review on audio encryption algorithms using chaos maps-based techniques," 2022.

[7] W. Hamidouche, M. Farajallah, N. Sidaty, S. E. Assad, and O. Deforges, "Real-time selective video encryption based on the chaos system in scalable hevc extension," *Signal Processing: Image Communication*, vol. 58, 2017.

[8] W. Chen and C. Ding, "Research on real-time video encryption algorithm based on moving objects," *Open Cybernetics and Systemics Journal*, vol. 8, 2014.

[9] D. Haridas, D. S. Kiran, S. Patel, K. Raghavendra, S. Venkatraman, and R. Venkatraman, "Real-time compressed video encryption: Based on quasigroup on system on chip (soc)," *SN Computer Science*, vol. 2, 2021.

[10] I. Al-Mejibli and S. F. Ismail, "Innovative lightweight encryption algorithm for real-time video," vol. 36, 2019.

[11] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2836, 2003.