International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

A Reliable Time Lock Encryption Scheme for Relatively Weak Computational Resources

Remyasree N, Chungath Srinivasan

Abstract*: Time-lock encryption is a type of encryption in which the process is bound by a factor of time that enables previously impossible applications such as secure auctions, mortgage payment, key escrow, or fair multiparty computations. Existing solution approaches of time lock either employ computational overhead to calculate time or use analogues to map the real-world time, hence lacks reliability. We propose a reliable time-lock encryption scheme, where even receivers with relatively weak computational resources can decrypt the cipher after an accurate real-world deadline, without any interaction with the sender. Proposed solution uses time fetched from timeservers over secured https channel for time lock accuracy and strong AES-256 encryption/decryption techniques for reliability. The paper briefly discusses a java based prototype implementation of the proposed approach and the experimental results.

Index Terms*: Time-Lock encryption, Time-Release Cryptography, Time-Lapse Encryption.

I. INTRODUCTION

Time Lock Encryption can be considered as an encryption where we are sending messages to future. The message encrypted by a time lock encryption process can only be decrypted after the preset deadline has passed. Any adversary with or without powerful computational resources should not be able to get the message before the deadline. However, even receivers with relatively weak computational resources should immediately be able to decrypt the cipher after the preset deadline, without any interaction with the sender, other receivers. Such a strong concept of secure encryption is impossible to achieve using a plain standard model of computation, as it lacks any equivalent to real-world time or a mapping to the real world time, which is provided in the real world by reference clocks. The first time lock encryption scheme published was based on the idea of time lock puzzles. In 1996, Ronald L Rivest, Adi Shamir and David A Wagner published a paper on time lock puzzles and Time Release Cryptography. They explored an approach based on computational complexity and studied the problem of creating computational puzzles called time lock puzzles, that require a precise amount of time to solve. This approach employs puzzles with a solution time that is only approximately controllable since different computers work at different speeds. A major drawback of the approach was that computational overhead of having a system work relentlessly on solving the puzzle. In existing systems, time lock capability is measured using the computational power of the decryption systems or using analogues for time. Though these solution approaches are able to perform Time locked encryption successfully, they have highly significant drawbacks. Thus the need to introduce the design of a scalable, economic and reliable solution approach which can accurately map real world time and employ less computational power. We propose an approach using real world time reference from NIST NTP servers, which provide accurate time with reference to the atomic clocks, and employs strong AES 256 encryption techniques. This combination helps us map to the real world time with the highest precision and uses only limited computational power. A predefined lock time is stored on a central system using a secured channel at encryption and is retrieved and validated with the response from NIST Server, this validation works as the trigger for data decryption. A prototype implementation of this approach was built and tested on variations of data formats and time periods. The paper is organized as follows. “Related Work” section introduces some of the existing solution approaches and designs. Afterwards, in “Solution Approach” section, we briefly discuss on the proposed Architecture. “Prototype” section describes Implementation of the prototype design. Then, in “Application Areas” section, we present the functional scope. Finally, in “Conclusion” section, we draw our conclusions.

II. RELATED WORK

In this Chapter, we discuss about three papers which deals with different methods of implementing Time lock Encryption. We shall also go through some of the basic components of the proposed solution approach such as NIST.

A. *Time Release Crypto*

An approach backed by computational complexity, based on problem of creating computational puzzles, [1] called time-lock puzzles that require a precise amount of time to solve. The solution to the puzzle reveals a key that can be used to decrypt the encrypted information. This approach has the obvious problem of trying to make CPU time and real time agree as closely as possible. The major difficulty to overcome is that those with more computational resources might be able to solve the time-lock puzzle more quickly,

Revised Manuscript Received on 30 July 2019.
* Correspondence Author
Remyasree N*., TIFAC-CORE in Cyber Security, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India.
Chungath Srinivasan, TIFAC-CORE in Cyber Security, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India

by using large parallel computers. The approach to building puzzles that appears to be intrinsically sequential in the desired manner. Of course, the approach yields puzzles with a solution time that is only approximately controllable (cannot map to real world time accurately), since different computers work at different speeds.

B. *Time Lapse Crypto*

A token-controlled public key encryption, first introduced by Baeketal [2]. In token-controlled encryption, where message encryption is done with both a secret token and a public encryption key, and once the token is released, it can only be decrypted with the private decryption key. For generating and distributing the secret tokens securely time-lapse crypto approach can be used. Time-lock puzzles together with a secret decryption key for time-lapse cryptography is a first. In the former scheme, public and private keys at appropriate time are created ad distributed by trusted third party. Here the protocol is similar, instead of the trusted third party the approach uses network of parties and an assumption that no less than a specified number of these parties need to be trusted.

C. *Time Lock based on Block chain*

Computational reference clocks as an extension of the standard computational model, which may provide a novel and very realistic method to emulate real-world time in a computational model [8]. The widely-used crypto currency Bitcoin provides a practical example of such a reference clock, which shows that the assumption that these objects exist in practice is very reasonable. The challenge in constructing time-lock encryption is to find a reasonable equivalent of real-world time in a computational model [3]. The key idea behind our construction of time-relation lock encryption is to combine a computational reference clock with witness encryption. We define an NP-relation R such that

1. For $x \in N$, statements have the form 1^x, that is, x in unary representation.
2. Any valid Bitcoin block chain $w = (B_1, B_2..., B_x)$ of length at least x is a witness for $(1^x, w) \in R$

III. SOLUTION APPROACH

A. *Introduction*

In existing systems, the time lock capability is measured using the computational power of the decryption systems or using analogues for time. Though these solution approaches are able to perform Time locked encryption, they have highly significant drawbacks. The Time locks using computational clocks requires resources that need to be kept alive and running for the whole period the time lock is designed for and hence it is not an economic or practical solution approach. The fact that computational clocks are based on a puzzle based approach does not provide a reliable mapping of real world time. Block chain based time locks, though makes use of the block chain mining computation reduces resource dependency, fail to map the real world time as the concept revolves around the length of the block chain rather than an accurate source of time reference. It lacks precision of the time lock duration. Hence the need for the proposed solution to provide as a scalable, economic and reliable solution approach which can accurately map real world time. The reliability, accuracy and precision of time lock is achieved via the NIST NTP. NIST Time are based on Atomic clocks with a precision of will not lose a second in at least 300 million years. The message data is encrypted with AES 256 (CBC). A random generated key is combined with the Date time using a *G*-function to generate a 128-bit key which is then passed through a sponge function to create a 256-bit hash and is used as the encryption key.

B. *Algorithm*

1. *Encryption*

Input:

- Consider the input message to be encrypted $m \in M$, Where M is the set of characters in message space.
- The date time until which the message is to be encrypted is to be given in [dd:mm: yyyy:hh:mm:ss].

The encryption scheme has a random key generator of 32 bits, r and t the date time input of 96 bits is given to a *G*-Function to get a 128-bit value.

$$G(r, t) \rightarrow g_{r,t}$$

The 128-bit value obtained is hashed (SHA-3) and the final hashed value acts as the key for encryption. AES-256 is the encryption scheme used with cipher block chaining as the mode of encryption and PKCSS as the padding scheme.

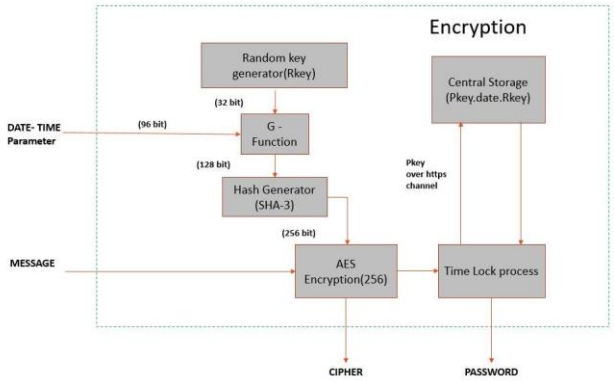

Figure 1: Encryption Architecture

After the encryption is completed, the date time parameter the random keys are stored separately in the different dBs over an https channel s/m generates a unique value as the password.

Output:

- After the whole of the encryption is completed, the user obtains the cipher text (.TL) extension.
- The trigger password for the particular (.TL) file.

2. *Decryption*

Input:

- The (.TL) File obtained
- The password trigger shared to the decryptor.

When user inputs the .TL file and the triggered password, the application gets the real time parameters from the NTP server [NIST servers has been used in the sample application developed] over an encrypted channel.

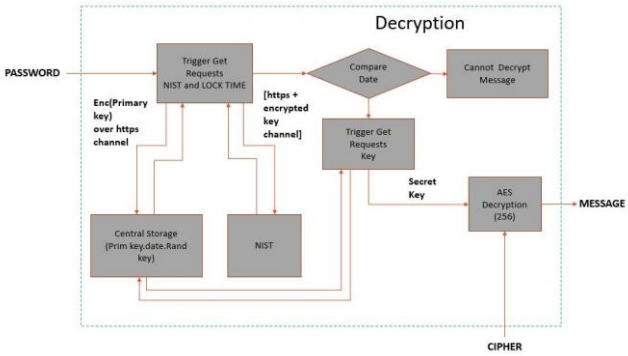

Figure 2: Decryption Architecture

The system compares the time parameters with the date time parameters of the .TL file, if the time limit has crossed; only then the random key is fetched to construct the decryption key, and decrypts the .TL cipher to the original message/file. If the time limit has not reached, the system aborts the process of decryption and sends an error message back.

Output:

- The original message.

Security Parameter used:

- 32-bit random key to ensure randomness of the decryption key.
- SHA-3 (Hash-One [4] can also be used) hashing algorithm to ensure the integrity of key
- AES-256 with CBC-PKSS to generate the cipher suite.
- Secure dB storage over encrypted channel, with data at rest encryption.
- NTP servers with secret encrypted channel as transmission channel and IP whitelisting.

C. *NIST*

The National Institute of Standards and Technology (NIST) [5] is a physical sciences laboratory, and a non-regulatory agency of the United States Department of Commerce. NIST provides as reliable and secure source for real world time for the internet using Network Time Protocol (NTP) and is synced to the Atomic clocks at NIST. NIST F2 and NIST-F1, cesium fountain atomic clocks serve as the primary time and frequency standard. It is expected to neither gain nor lose a second in more than 100 million years.

Table 1: Nist Ntp Server List

Name	IP Address	Location
ntp-b.nist.gov	132.163.96.5	NIST, Boulder, Colorado
ntp-wwv.nist.gov	132.163.97.5	NIST WWV, Fort Collins, Colorado
ntp-c.colorado.edu	128.138.141.177	JILA, Univ. of Colorado, Boulder
ntp-d.nist.gov	129.6.15.32	NIST, Gaithersburg, Maryland

IV. PROTOTYPE

A prototype system was developed based on the proposed architecture using Java and rest service API's. The prototype system supports time lock encryption for files of multiple extensions and with time lock duration accurate to minutes. The system has a very simple and concise UI for user interaction, which has been modelled in java. We used Restdb.io, an open source platform provides as the central storage system for the application. It takes care of storing the file information, lock time and the cipher key securely. Data are stored as collections in the mongo dB of the restdbio. This data is fetched over https channel as and when required, which is achieved via rest service APIs authenticated via API key provided by the restdbio. Real world time reference is made available for the system using NIST server NTP service. The NIST server time compared to the lock duration time fetched from the central storage and decrypt process is initiated based on the outcome. The NIST server provides IP white listing and the time is fetched over an https channel encrypted with a secret key shared between NIST and the client System. The message file is encrypted on the server side with AES 256 using a key based on the time lock duration time and a random generator. Once the time lock duration is complete, user requests the file decryption using the trigger password from client side application. The time from NIST and central storage is compared, decryption process initiated, and the decrypted file is then made available to the user. If a user request is triggered for decryption of file before the time lock duration period is complete on the client side, the time compare validation with NIST reference kicks in, kills that request, and exits out of the flow. Hence, password used by the client user is only a trigger and does not enable the user to view/decrypt the true message until the time lock duration is passed. The prototype model was successfully tested across different file types, operating systems and lock duration. Even relatively large files were processed within a few seconds and there was no overhead processing on the system. On an average the prototype system delivered an accuracy of a few seconds on the time lock duration. Further enhancements can be done to the prototype to develop a complete reliable robust time lock encryption application.

Table 2: Application Testing

Test Case	Format	Size	Time	OS
1	TXT	10MB	4sec	Windows
2	PDF	100MB	7sec	Linux
3	MP4	500MB	10sec	Linux
4	FLV	1024MB	15sec	Windows

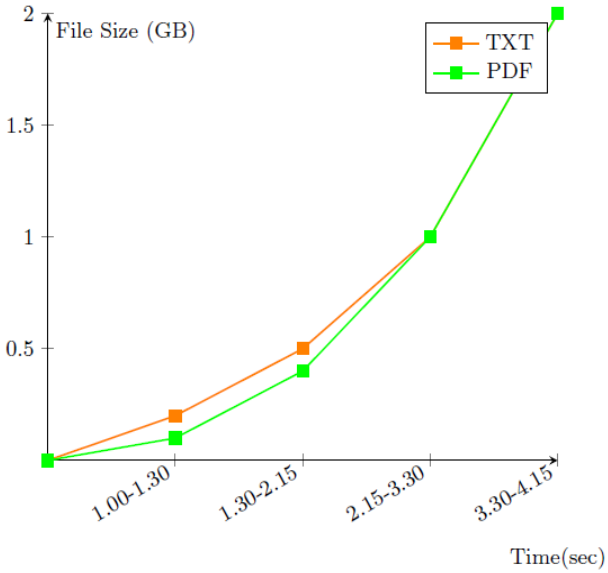

Figure 3: Average Encryption Time

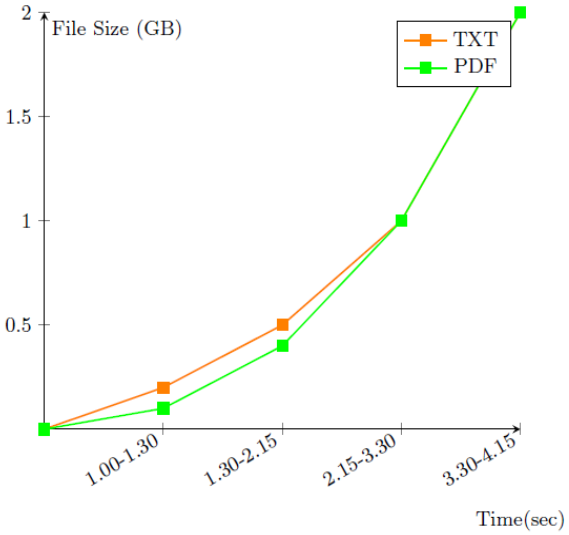

Figure 4: Average Decryption Time

V. APPLICATION AREAS

The functional scope of the proposed solution approach is directly in line to that of the concept of Time lock encryption. Since time lock encryption brings in the parameter of "Time" to cryptographic world, it is of great significance. The concept of securely locking a message for a duration of time, making it possible to decrypt only after the time window has numerous applications in the real world from something as simple as sending a message to be read in the future for securing complex critical legal documentations. Sending messages to the future is one of the most crucial but yet simple applications that can be achieved by the solution approach. A message will be encrypted and will be made readable by decryption only after a specified time period has passed, hence making it available to the reader only in future. Legal documentations like testaments can be locked down in time by the system for years together waiting for an event like the user to be of legal age or a period as defined by the legal system. Auctions are another area of interest where time locked auction transactions cannot be read until the duration is complete, and hence cannot be corrupted. Ensuring a secured and incorruptible legal system for auctions. Also Time locked documentation provides as a secured legal system on which the Mortgage payment can be processed. The Time lock solution can be used to generate the receipts and legal documentations of mortgage payments ensuring that trusted party is not required. Even E-Voting or Electronic Voting used in many different democracies, where there is widespread mistrust to the process, can be achieved with a time locked encryption system. It can restrict the access to the votes for a given period, as it is impossible to decrypt the data before the duration is complete, minimizes the risk of corruption in the system to 0.

VI. CONCLUSION

As detailed in this paper, the proposed approach serves as a reliable, accurate and practical solution approach to the problem of time lock encryption for weak computational resources. The reference from reliable time servers provides a solid and accurate reference to the real world time and hence ensures time lock duration to be accurate to a few seconds, which was never achievable through the earlier approaches. The trigger driven decryption methodology ensures that the checks and computation is performed only when triggered by a password, thereby reducing the overhead of resource usage or memory/service utilization. The prototype implementation of the proposed approach was successfully designed and tested on systems with low computational power. Thus, the solution approach proposed serves as a reliable Time lock encryption scheme, which is irrespective of the computational power (string/weak) and can map real world time accurate to seconds. Self Decryption Files is an area which requires further attention as the currently proposed architecture lacks the facility to self-decrypt the file and needs to be triggered by a password. Future scope would include automated decryption at the end of time lock duration and new complex encryption algorithms to ensure improved reliability.

REFERENCES

1. G Ronald L. Rivest , Adi Shamir , and David A. Wagner, Time-lock puzzles and timed release Crypto (1996)
2. Michael O. Rabin,Christopher Thorpe, Time-Lapse Cryptography Technical Report TR22-06 (2006)
3. Tibor Jager, How to Build Time-Lock Encryption; Designs, Codes and Cryptography-Volume 86(2018)
4. Dr. M. Sethumadhavan, Megha, P., Dr. Sindhu M., and Dr. Srinivasan C., "Hash-One: a lightweight cryptographic hash function", IET Information Security, vol. 10, 2016
5. NIST : https://tf.nist.gov/tf-cgi/servers.cgi
6. William Clarkson, Time-Lock Cryptography (2013)
7. Mohammad Mahmoody, Tal Moran, and Salil Vadhan. Time-lock puzzles in the random oracle model (2011)
8. Schwenk J., Modelling time for authenticated key exchange protocols.(2014)
9. Todd, P, Timelock encryption incentivised by Bitcoin.(2014)
10. Unruh D, Revocable quantum timed-release encryption.(2014)
11. Liu J., Kakvi S.A., Warinschi B., Time-release protocol from Bitcoin and witness encryption for sat.(2015)
12. Katz J., Miller A., Shi E., Pseudonymous secure computation from time-lock puzzles.(2014)

AUTHORS PROFILE

Remyasree N. is pursuing her master's degree M.Tech with specialization in Cyber Security at TIFAC-CORE in Cyber Security, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India.

Chungath Srinivasan received his Master's Degree from University of Calicut and Ph.D. from Amrita Vishwa Vidyapeetham, India. Currently, he is an Assistant Professor at Amrita Vishwa Vidyapeetham, India