ASEAN
Journal of Scientific and Technological Reports
Online ISSN:2773-8752

Research article

Gid Crypto: Application for End-to-End Encrypt and Decrypt E-mail and Data

Alain Jean[1], and Tossaporn Alherbe[2*]

[1] Department of Mathematics Statistics and Computer, Faculty of Science, Ubon Ratchathani University, Ubon Ratchathani, 34190, Thailand; alain.j@ubu.ac.th
[2] Department of Mathematics Statistics and Computer, Faculty of Science, Ubon Ratchathani University, Ubon Ratchathani, 34190, Thailand; tossaporn.c@ubu.ac.th
* Correspondence:tossaporn.c@ubu.ac.th

Citation:
Jean, A.; Alherbe T.; Gid Crypto: Application for End-to-End Encrypt and Decrypt E-mail and Data. *ASEAN J. Sci. Tech. Report.* **2024**, *27*(2), 90-102. https://doi.org/10.55164./ajstr.v27i2.251127.

Article history:
Received: October 2, 2023
Revised: February 27, 2024
Accepted: February 28, 2024
Available online: March 1, 2024

Abstract: Privacy and security in our digital environment have become essential today. Some data stored on hard disks, in flash memory, or exchanged through our communication channels is sensitive and should be kept confidential. Additionally, documents containing confidential information, such as business algorithms or new application coding, require adequate measures to ensure privacy and security. Therefore, data encryption is crucial in protecting information from being intercepted, misused, or stolen. While encryption tools are available for business e-mails, using these tools often requires subscribing as a member and paying extra fees. Effective tools are not limited to e-mail alone. Still, they can also be applied to any text data that requires a high level of security and complexity to ensure the protection of the data for a duration longer than it needs to be safeguarded. Our research aims to develop a stand-alone application that encrypts data to ensure its confidentiality without additional subscriptions or fees. The application is designed to be user-friendly, with fast encryption and decryption capabilities. It can be used not only for encrypting and decrypting e-mail messages but also for programming languages and Excel data. The application development utilizes techniques such as Diffe-Hellman, Mersenne Twister, and AES, arranged in layers to generate complex ciphertext that is difficult to decrypt without the developed application. LabVIEW was used for testing encryption speed. The attempt to attack encrypted data was subjected, but the result yielded no successful key retrieval.

Keywords: Encryption and decryption; Diffie-Hellman; Mersenne Twister; Advanced Encryption Standard; LabVIEW

1. Introduction

Since E-mail was developed in 1971 [1], it has become one of the primary communication channels and is widely used on the internet. According to data research from Raicati Group [2], in 2022, the number of worldwide e-mails, including both business and consumer users, was 4.26 billion, with the daily volume of sent/received e-mails per day being 333 billion. Along with the rise in the number of e-mail users, it is predicted to reach 4.6 billion users in 2025.

Currently, e-mail has mainly been used in any kind of online activity for personal and group communication. It is used not only for instant messaging but also for signing up for online commerce, sending updated news notifications,

and authorizing other forms when one is present on the internet. While users use e-mail for communication, they may be imprudent to send the most sensitive kind of secret data such as credential numbers, application passwords, birth dates, important business data, personal contact names, numbers, or medical health data. Unfortunately, they are unsecured since regular e-mail and data sending are not encrypted. It allows for phishing and other service providers unauthorized access.

There are 2 protocols commonly utilized for e-mail encryption: Encrypting an e-mail while it is in transit (Transport Layer Security: TLS) and end-to-end e-mail encryption. TLS is a cryptographic protocol that provides security for data sent between mail server to mail server. It uses public critical infrastructure (PKI) to encrypt messages while they are in transit from sender to recipient. TLS stops e-mails from being read after the sending but before the messages are delivered to the end system. It can avoid eavesdropping or altering e-mail contents while delivering data over the internet. The advantage of using TLS is that it is easy for users to send e-mails without extra work beyond clicking the send button, as it is provided by e-mail providers such as Google (Ghali et al. [3]) and Apple [4]. However, if a cybercriminal can attack an e-mail account through phishing, they can still read e-mail. Since TLS does not encrypt e-mail messages and cannot hide the messages from the e-mail servers themselves, e-mail providers can still access the contents of our messages.

End-to-end encryption is also known as public key encryption. It is a method where e-mail messages are encrypted by the stand-alone on the sender's system, and only the intended recipient can decrypt and read them on their device. End-to-end encryption provides the highest level of confidentiality as it prevents any third party or cybercriminal from reading or accessing messages at any delivery state. The encrypted messages also cannot be read by the e-mail server. Moreover, the attackers cannot access data on e-mail servers since they are encrypted on users' endpoints. If an attacker tries to retrieve data from the servers, they will get gibberish. End-to-end encryption makes communication safe. One may want to ensure sending financial and private communication using end-to-end encryption since it is difficult for cybercriminals to retrieve sensitive information.

Our work presents the Gid Crypto application, a stand-alone text data encryption and decryption; it is easy to use with any e-mail text body. This tool ensures the generation of secure e-mails and encrypted text data. We utilize various techniques, such as Diffie-Hellman [5] for key exchange, Mersenne Twister [6] for issuing genuinely random numbers, and Advanced Encryption Standard (AES) [7] for creating cipher texts, to ensure the utmost security of the information content.

The rest of the paper is organized as follows: Section 2 describes the material and methods used to develop the tool. The results and discussion of our work are presented in Section 3, and the conclusion is provided in Section 4.

2.Materials and Methods

2.1 Encryption and Decryption Principles

Encryption is turning plain text content, like text messages or e-mails, into cipher text/scrambled text to protect it from being read by unintended parties. When the intended recipient receives the message, only the person with the appropriate private key that matches the public key used to encrypt the message can read it, which is called decryption.

Our goal is to make an application that is easy to use and maintains the highest security and cryptographic standards. We follow 3 principles when developing this application. The first is implementing Diffie-Hellman as a key exchange protocol; the second is issuing a new and unique encryption key with each contact so we do not establish a similar pattern; we use Mersenne Twister. Finally, we use the Advanced Encryption Standard (AES) to generate cipher and decrypted texts.

2.2 Diffie-Hellman Principles

Diffie-Hellman (DH) [5], or exponential key exchange, was one of the first public-key protocols conceptualized initially by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. It is a method for securely exchanging cryptographic keys over an insecure public channel. For example, if two people, Alice and Bob, wish to communicate over the internet, they need a way to exchange information that will be known only to them. To do that, Alice and Bob first generate their private and public keys and then send them to the

other side. After both parties obtain copies of each other's public keys, they compute shared keys. The shared keys will be used for symmetric cipher communication.

These steps can be understood with the diagram depicted in Figure 1. Our work uses the DH method of securely exchanging cryptographic keys on a secure channel and even over an insecure channel.

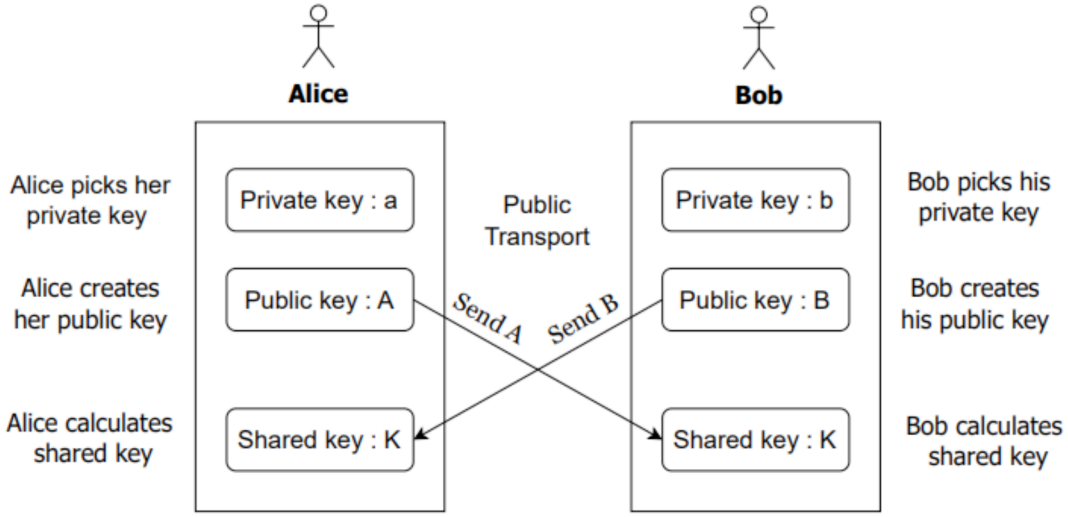


Figure 1. Diffie-Hellman Key Exchange

There are some methods for the key exchange, such as Elliptic Curve Diffe-Hellman (ECDH), RSA, Lattice-based Cryptography (LBC), and Post-Quantum Cryptography (PQC). We selected DH for its simplicity, high effectiveness, and widely standardized. Moreover, even ECDH [8] offers more efficiencies but would not offer more secure computation, while RSA [8] is used for key exchanges that lack forward secrecy. LBC [9] holds some good promises for the future, but like RSA, it is more of a classical Public Key systems certificate. On the other hand, PQC [9] is mainly to thwart attacks from a quantum computer, which is not yet widely accessible, regardless of whether it is not a key exchange mechanism, and AES-256 has so far demonstrated robustness against quantum threats. Our development work aims to make it usable on general-purpose computers. With the Gid Crypto, we would like to create a stand-alone encryption system with no certificates.

2.3 Mersenne Twister (MT)

Mersenne Twister is the most widely used pseudo-random solid numbers generator (PRNG). A version of Mersenne Twister, MT19957, can generate an extremely long sequence with a period of $2^{1,9937}$ -1 numbers before the identical series of numbers repeats [6]. If we compare MR19957 to the native Excel random number generator function, (RND), it can only generate up to 2^{24} sequences of numbers [10]. To ensure a higher level of security, it is important to consider the length of sequences. The shorter the periods of sequence, the easier the sequence can be detected and potentially reused by attackers. Therefore, we used MT as it provides a more extended period and enhances the generation of truly random numbers.

While the MT is considered the default random number generator (RNG) in Python, it does have some drawbacks, as outlined below [11] :

- If one can retrieve its previous bits, it is possible to predict the subsequent bits.
- Despite its massive size, MT may fail to meet specific statistical tests.
- It is not flexible; it follows a specific mechanism.

After conducting numerous thorough tests, studying other advancements, and using our experience to prevent the retrieval of bits by utilizing an MT random number generator, we iterate the process of generating a random number using the MT output as an input 10 times. This ensures enhanced randomness and makes it exceedingly difficult to reconstruct the original bits.

2.4 Advanced Encryption Standard (AES)

The AES is a standardized symmetric block cipher developed by the U.S. National Institute of Standards and Technology (NIST) [7]. It operates with a fixed data block size of 16 bytes, with each block being converted by using key sizes of 128, 192, or 256 bits long. After these blocks are encrypted, they are combined to form a ciphertext. Other features of AES are also concerning:

- Stronger and faster than Triple-DES
- It comes with detailed specifications and design.
- It is highly resistant to brute-force attacks, as it would take 36 quadrillion years to crack a 128-bit key [12]. Even with substantial computing power, cracking a 256-bit key would require more time.

AES encryption is widely recognized as the standard for symmetric encryption due to its fast and secure nature. It requires a 32-byte key, a 16-byte IV, and a ciphertext as input to generate the corresponding plaintext output. The encryption levels can vary from 64 to 256 bits. In our work, we employ a 256-bit key length.

2.5 Database Technology

To make it practical for users, we added a database so the different keys (private, public, and shared) could be saved, and the key exchange only had to be done once or at the users' discretion, making recurrent e-mail easier or the saving of multiple documents.

We use MongoDB as a local light and flexible database server. Moreover, PyMongo, the standard MongoDB driver library for Python, is easy to use; it offers an intuitive API and interface, as shown in Figure 2, for accessing databases, collections, and documents. It also lets administrators encrypt data in transit and data in permanent storage. Objects retrieved from MongoDB through PyMongo are compatible with dictionaries and lists, so one can easily manipulate, iterate, and print them.

Figure 2. Contact list in MongoDB

2.6 Interface

We aimed to create an interface that is both intuitive and simple. To achieve this, we applied UX/UI technology, starting with creating a persona and an interaction design focused on simplicity of use. The Gid Crypto encrypted and decrypted program has multiple interface screens: Contacts, Role Setting, Contact Name, Keys generation, and encrypted and decrypted screens.

2.6.1 Contacts

The initial page is the Contact page, shown in Figure 2, where users can create new contacts to initiate communication. For those who have interacted previously, a window displays a list of contacts and their corresponding encrypted keys on the right-hand side.

2.6.2Role Setting and Contact Name

If a user selects to create a new contact, the user has to determine if he is a sender or receiver. Thus, the ROLE SETTING page is utilized. Once the user selects their role, they must input the contact name for communication records, as seen in Figure 3.

Figure 3. Role Setting and Contact Name pages

2.6.3 Keys Generation

The Keys Generation page is used when creating a new contact. This page is designed to work based on the Diffie-Hellman key exchange algorithm. In this process (Figure 4), the sender, responsible for initiating the communication, selects any desired private key and clicks the 'Save' button to save it to the database. Next, the sender can generate a public key automatically by clicking the 'Get Public key' button. The next step involves the sender sending their public key to the recipient on the other side. The recipient will then follow a similar process by selecting a private key and generating their public key. The recipient will send their public key back to the sender.

To generate the shared key on the sender's side, the private key of the sender, the public key of the sender, and the public key of the recipient are used.

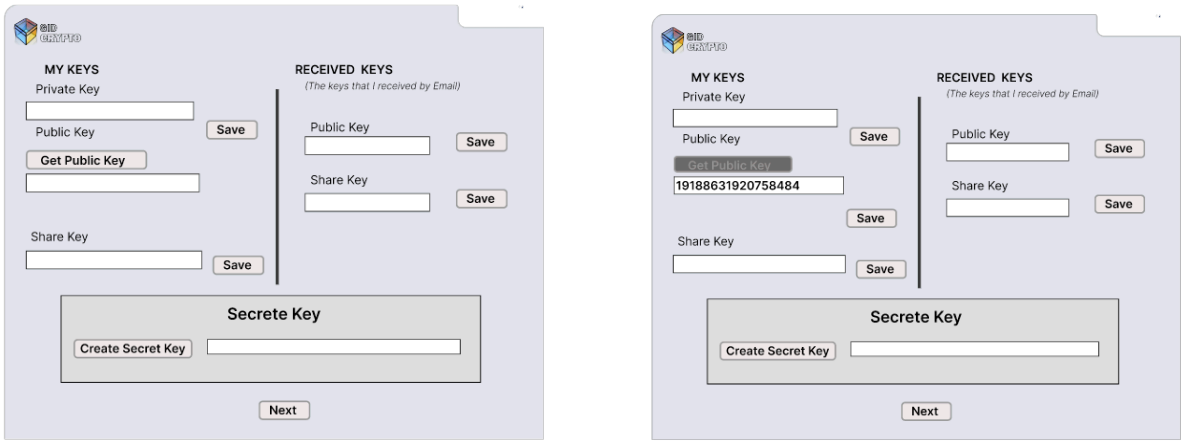


Figure 4. Keys Generation page

Figure 4 illustrates generating a Secret key at the bottom of the screen. To make it practically impossible to crack, we designed to have another key, the Secret key. The Secret key is generated by adding a 2-layer extension of the Diffie-Hellman key exchange. The first layer, the MT algorithm, generates random numbers, which are inputs for generating another random number for many rounds. This procedure makes it

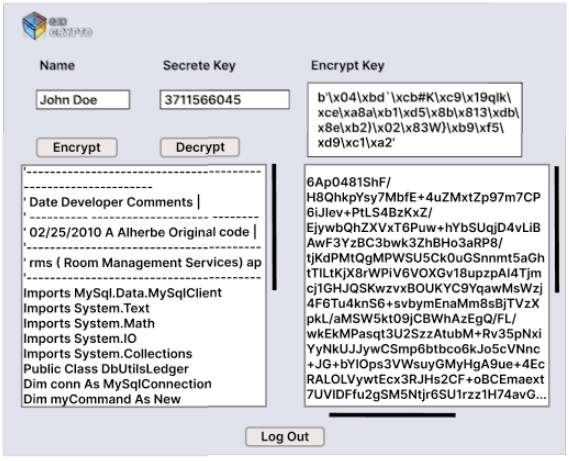
difficult for malicious individuals to calculate back to the initial MT numbers. The second layer is used with the SHA-256 algorithm once a random number is obtained to generate the final Secret key. Hence, our developed application utilizes a variety of algorithms to achieve highly efficient message encryption. As a user, all they have to do is click the 'Create Secret Key' button, and Gid Crypto will automatically generate the Secret key based on the previously described processes. In this paper, implementation details and code samples will not be shown to keep the application secure for anyone using it.

2.6.4 Encrypted and Decretep Interface

The encrypted and decrypted interface is designed to provide ease of use for users. To access this interface, there are two options: selecting a contact list in Figure 2 or following the steps in Figure 4 to create a new contact. Using this window is straightforward, as shown in Figure 5. Users can easily encrypt a message by typing the desired text in the left-hand text box and clicking the 'Encrypt' button. The message will be encrypted and displayed in the right-hand text box. Similarly, to decrypt a message, users can paste the encrypted text in the left-hand box and click the 'Decrypt' button, and the original message will be displayed in the right-hand box. Examples of encrypted and decrypted interfaces for text e-mail, a CSV file, and coding are displayed in Figure 5.

Text encryption

CSV decryption



Code encryption

Figure 5. Encrypted and decrypted Interfaces

To use the application, if one wants to keep his e-mail message secure, he will type his message into Gid Crypto for encryption. Then, he can paste the encrypted message into his e-mail body and send it to the recipient, who can decrypt it using his version of Gid Crypto.

3. Results and Discussion

3.1 Testing: AES Block Ciphers

We need to understand that AES relies on the substitution-permutation network principle to test our encryption strength. This means it performs linked operations involving replacing and shuffling input data. To evaluate the encryption capabilities of a stream cipher, it relies on 3 main components :

- Generating greater uniqueness and sensible cryptography
- AES Information Blocks testing
- To test the encryption speed

The details of each test are as follows.

3.1.1 Generating greater uniqueness and sensible cryptography

To create the maximum entropy for our secret key, we employ a coding process consisting of four consecutive steps; however, the details of these steps are not provided here. This approach is essential because a possible attacker could compromise the public and shared keys. In the event of an attack, the adversary's best strategy involves attempting to guess the key via brute force, Padding attack (PA), Chosen plaintext attack (CPA), or Chosen ciphertext attack (CCA) [13-15]. For the chance of being right, it needs to compute the stream cipher, then check to see whether it produces the observed outputs, and if so, guess the next bit it produces. So, it is a lengthy and highly complex process and seems unlikely to succeed.

We apply the Mersenne Twister on the resulting secreted key for E-mail, Coding-data, and database-data as follows :

1. Text E-mail: The application was tested for descending regular text e-mails. We first exchanged our Public key, then made our Shared key, exchanged that key also, then encrypted regular text and sent it. The party receiving the e-mail on a different computer had no problem decrypting the text.

2. Coding-data*:* In a second test, we took code in C# and Python and did the same. We sent our encrypted code through e-mail and saved it in a directory to keep the code private.

3. Database-data: In our third test, we export data from the database/excel table and save it in "comma-separated" format, encrypted and like with code before, e-mailing it to a third party or saving it in a local directory for safekeeping.

According to our attempt to test it in a brute force attack covering 4 days, the testing yielded no success in obtaining even the first 16 bits of the key. While it remains a theoretical possibility, achieving such a feat would require an exceptionally powerful computer and extensive time and resources that were not at our disposal during the testing process.

Our application uses a 256-bit key length to encrypt and decrypt a block of messages. The process involves 14 rounds of 256-bit keys, each including processing steps that entail substitution, transposition, and mixing plaintext to transform it into ciphertext. By selecting $k = 256$, we aim to provide robust protection against potential adversaries. While standard cryptography typically opts for a key length of $k \geq 128$ for security purposes, our implementation surpasses this standard by choosing $k \geq 256$. This decision helps mitigate multi-target attacks and potential future quantum cryptanalysis, ensuring high security.

3.1.2 AES Information Blocks

It is important to note that our primary objective is not to safeguard a network against penetration but rather to protect the privacy of a text message. Therefore, our focus is on establishing a robust encryption key. Even if decryption were possible, the considerable time and specialized hardware required to achieve it would render the effort futile, ensuring the preservation of the message's confidentiality.

For this, we use Symmetric Encryption AES-256 to provide quick encoding or decoding operations for those who want to make an easy first choice for local or e-mail data encryption. As we mentioned above, AES is an algorithm for block encryption that is in widespread use. There are six modes of operation of the AES algorithm that were standardized and are the most well-known: ECB (Electronic Code Book), CBC (Cipher

Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback), CTR (Counter), and GCM (Galois-Counter-Mode).

We evaluated those 6 encryption modes to determine the most suitable option based on the level of security provided. Each mode offers a different level of security, so we examined the data from each mode to make our decision. CTR and GCM stood out among the modes considered since AES-GCM is a well-known and documented process. The GCM utilizes symmetric block encryption with CTR as the underlying cipher mode. It also integrates authentication to protect the cipher, eliminating the need for separate integrity measures and significantly enhancing encryption efficiency and security. While the time difference between the two modes was minimal, we ultimately chose GCM due to its stronger security capabilities. Using GCM, we could measure the time it takes to encrypt and decrypt data in LabView [16].

3.1.3 Encryption speed

We used LabVIEW [16], an application for measurement, control, and testing systems, to test our encryption speed (as depicted in Figure 6). This testing aims to confirm that encryption of lengthy text can be completed in a reasonable amount of time. Table 1 shows our testing speed, as demonstrated by various examples. Those examples have been chosen to illustrate various approaches for testing text e-mail, CSV, and source code files.

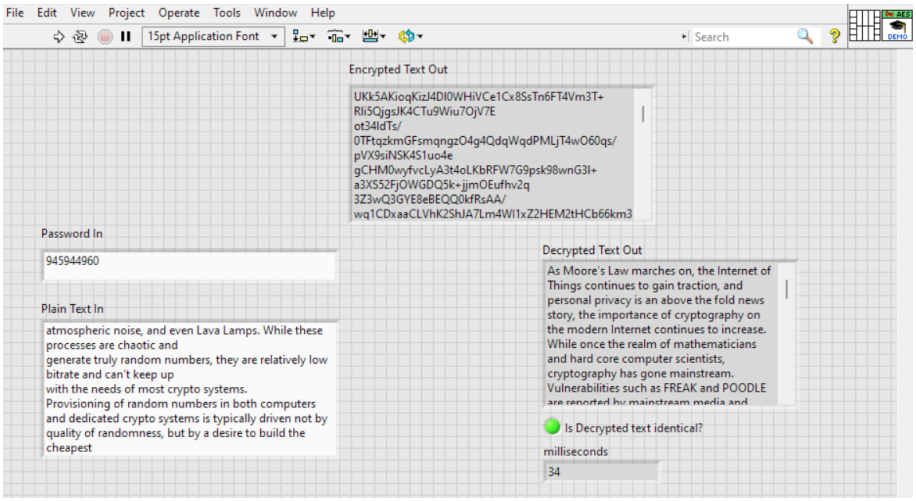


Figure 6. LabVIEW module for test speed: Mode-GCM/e-mail: text of 1945 characters with the encrypt and decrypt speed of 34 milliseconds

Table 1. Testing speed

Clear Text	Length (char)	Encrypted text	Length (char)	Encryption/ Decryption Time (ms.)	Mode
A stream cipher—a deterministic mathematical function—is considered secure if an adversary who does not know the k-bit key is chosen uniformly at random and can only …	425	pRvh7gqfHbrQTl1Qz/sVk7Og7PZ7 RKN//jbw2zlPUP2H1Q8DES2qhio Lg+e/mfpdfD6hvhtCz3YqIN4/TMz Bk9kbMhl0kQS/RjY9wF502iH3xI3 YZ/zDan2YHh2k3MAk3IEVpOYu bR4xlZZQAFAPhLe7zkf4aZwZ8Nj k7Z3mFQUtqjhTXNgSzaGDY4JFG zYyWdTu…	1088	3 ms. 5 ms.	GCM/e-mail: text

Clear Text	Length (char)	Encrypted text	Length (char)	Encryption/ Decryption Time (ms.)	Mode
As Moore's Law marches on, the Internet of Things continues to gain traction, and personal privacy is an above-the-fold news story, the importance of cryptography…	1946	E51Lv/WFLiFOi3KpBzdLXp5q+h0 vG578z3asuLAw9jgYcBYzbHICQZ rKgoems7ssgo/zf0z9+VnOaD2f9N4 G/4WKxjLyhzce1j7Ao9V5Fbuau9 fjcFQUjgBJr0KwL87hFULo0jhohqR 1FuDv9xrEmdm UhW1eynz2BldJsXl6KF …	4748	15 ms. 21 ms.	GCM/e-mail: text
spinoza-francis.net,2020-08-26,http://www.escobar.org/2,10dAcafEBbA5FcA,Kristina,Ferrell,"Horn, Shepard and Watson",Aaronville,Andorra,932-062-1802,(209)172-7124x3651,xreese@hall-donovan.com,2020-04-27,https://tyler-pugh.info/3,67DAB15Ebe4BE4a,Briana,Andersen,Irwin-Oneal,East Jordan,Nepal,8352752061,(567)135-1918,haleybraun@blevins-sexton.com,2022-03-…	1699947	wXNMWiGpjIdI2c0ah5hm3ClO6Z Hiooo7dt2v3GCycqez49tEldm8X6 MazuDoh0kjoRfUMLpfuXu2aPS/2 cqDBkfz69p+TiaOXMKt1WWK15L QYtjyTmsV8dx5jTeLCcBg9YWCV M6x3L19IzUd0EqeUVr2S61gnXIte mJadnlEubmRETD58sJRADLPY2Z vkVuEqEnNdmYE4Gd3ay+cU//1W JJGR21sNBKGxD/12WmgIRVDsXS WQmSispMuLz6igw7rQ0+myxNh pOaf5QcCYmeWgUO3nDlLL4Cqg LkqAHB7OnUI8J/HBPP97rFYrU8 OfXAmTQ/4gA+E+tItY02vXpwagc CmfIotAzbn5HH5RVsD3zzI24rCG BF7bc+PqNntjfE46Y+XVLlpswjoJgy vHfiD+ …	4053272	117 ms. 161 ms.	GCM/ CSV: text
'---------- ' Date Developer Comments | ' ---------- ---------- ---------- | ' 02/25/2010 A Alherbe Original code | '---------- ---------- | ' rms (Room Management Services) application/web service | | '---------- Imports MySql.Data.MySqlClient Imports System.Text Imports System.Math Imports System.IO Imports System.Collections Public Class DbUtilsLedger Dim conn As MySqlConnection …	7334	6Ap0481ShF/H8QhkpYsy7MbfE+4 uZMxtZp97m7CP6iJIev+PtLS4BzK xZ/EjywbQhZXVxT6Puw+hYbSUqj D4vLiBAwF3YzBC3bwk3ZhBHo3a RP8/tjKdPMtQgMPWSU5Ck0uGSn nmt5aGhtTlLtKjX8rWPiV6VOXGv 18upzpAI4Tjmcj1GHJQSKwzvxBO UKYC9YqawMsWzj4F6Tu4knS6+s vbymEnaMm8sBjTVzXpkL/aMSW 5kt09jCBWhAzEgQ/FL/wkEkMPas qt3U2SzzAtubM+Rv35pNxiYyNkU JJywCSmp6btbco6kJo5cVNnc+JG+ bYlOps3VWsuyGMyHgA9ue+ EcRALOLVywtEcx3RJHs2CF+oBC Emaext7UVlDFfu2gSM5Ntjr6SU1r zz1H74avGP96Grd61H7v24umKC ZBs/SqKjFy5WIijsyEmXg+WlmnV uXpkj7xvQgcRjyz3hFDSGYUNYu dY0HPiUrPp6e+DBFejg06l2UOkH xlycepISyCH6yW7T1x7HOJ9cCNP calsjZx0nji5QkpD3CJsXgeZ1s/7DV Ef0MLPvX1UidjWOzDuCFNVTTU u7cUPVJyzv6q53/OIcoQu+htp9GC +Ne08Pv7BDUS …	18108	53 ms. 73 ms.	GCM/ code: text

AES-256 GCM was tested versus CTR, as we expected GCM to require slightly more processing times (0.059896 milliseconds) than Mode CTR because of its key-chaining nature.

We test 2 factors: the first is the issuing of the shared key and the second is the text encryption. Testing was done with the keys' bit length with a system running 16 GB RAM and Intel core I5. A private (secret) different key length from 4 to 10 digits, with a public key length from 8 to 16 digits. The bigger the private and public keys, the longer the Shared key calculation time.

For the best results, see Figure 7. Testing has shown that a 6-digit private key and 16-digit public key offered the best security versus functionality and speed (less than 1 second to create a 16-bit share key). The best results were obtained with a 4-digit to 6-digit private key and 2 full 64-bit public keys.

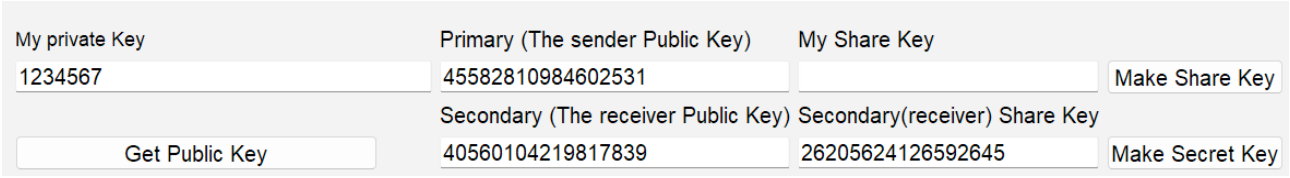


Figure 7. 64-Bits Shared key

3.2 Related work

Our work proposed Gid Crypto, a stand-alone encryption end decrypt application that runs independently of network connectivity. The systems that operate similarly to Gid Crypto are rarely found. Most available systems utilize certificates, a feature that Gid Crypto does not employ. Other applications that work on cloud platforms or utilize key servers are as follows.

Microsoft Outlook [17] and Yahoo [18] are examples of services that use encryption in transit, which means they protect the message only while in transit and cannot guarantee that it stays encrypted after the messages reach the recipients. However, Microsoft 365 [19] provides options that use OME, IRM, and S/MIME technology for users to set up e-mail encryption communication.

There are commercial tools that provide end-to-end encryption, such as Posteo [20], Tutanota [21], and Lavabit [22]. Those e-mail providers generate closed platforms where the services handle key management on their servers. Only users using the same platform can encrypt the messages to each other. To use their services, one needs to register on their domains. Posteo has a trial account and later enforces an extra monthly charge if users prefer additional storage space. Tutanota provides a free account for 1GB storage with basic private communication. If users prefer multi-user support options, they must pay for a business account, while Lavabit does not have a free trial account.

Trustifi [23] is cloud-based end-to-end security; it uses 256-bit AES encryption as our work. The same as Posteo and Tutanota, it requires users to register on the system to use the service on their platform; both senders and recipients must be members of Trustifi to communicate with each other; otherwise, the sender must issue an advanced method on how to secure the sending message, for example, setting up a password or phone number; yet merely 10 countries are covered. Trustifi service charges vary depending on the number of users and the size of the organization.

Privilege [24] is an encrypted file-sharing and e-mail application. For e-mail, after Preveil is installed, it creates a set of mailboxes for encrypted messages stored on PrevVeil's server. It can be used with Gmail and Outlook e-mails. If a PreVeil user sends an e-mail to another PreVile member, the message will be automatically encrypted; if the recipient is not a PreVail user, the message will be sent as regular mail or selected to be sent encrypted, then the recipient will be invited to join PreViel. It is free for basic features usage. For a business/professional level, a monthly payment is required. Using PreVeil, one needs to have an e-mail account while our work is free from applying for an account.

Signal [25] is a free encrypted application for private messaging on Android and iOS mobile and Desktop. However, to use Signal on a desktop, one must first install it on his mobile phone. Signal supports secure conversation, shared photos, and video calls.

Work by (Livington et al. [26]) also has a class of encryption that provides a client-side encrypt-on-receipt mechanism. Still, as we learn from Snowden [27], most of the work now is done at the server level or by infiltration or official request, as it was done with "Silent Mail" whereby the data is checked as soon as it reaches the Imap server, and before encryption whatever you are sending or receiving mail. In the case of receiving, the user receives the ciphertext, the key decrypts it, and the spy software reads it. Whether working for the government or not, a good hacker can infiltrate a mail server. This type of encryption will protect your data during transport but will not offer the complete security of someone with sensitive data.

4. Conclusions

As our world becomes increasingly digital, the importance of privacy and security has grown significantly. Communication has been one very important facet of coping with the challenges induced by societal digitization. We need a multidisciplinary approach to develop the necessary tools for communication. It is crucial to balance usability and security, so Gid Crypto was designed with these considerations in mind. We aimed to make the application as secure as possible while being user-friendly and easy to navigate. Thus, Gid Crypto was designed to encrypt e-mail messages, text files in .doc or .docx, and CSV file formats. We also must remember that we live in a world of constant evolution where progress is advancing rapidly. So, Gid Crypto was created with a flexible architecture that allows updates and upgrades.

Gid Crypto is a compact yet potent application explicitly designed for users seeking absolute control over their data encryption process. It stands apart from other e-mail encryption systems, which sometimes have potential backdoors compromising security. Gid Crypto guarantees no such vulnerabilities exist, offering users peace of mind. It is an ideal choice for ensuring the seamless operation and enhanced security of a Silent Mail system. With its advanced encryption algorithms and robust functionality, Gid Crytpo is exceptionally well-suited for safeguarding the privacy and security of sensitive data. It provides a reliable solution to complete a Silent Mail system and ensure the secure exchange of information.

We understand that our current work has limitations, particularly in terms of the accessibility of data that needs to be encrypted and the evolving field of cryptanalysis. To overcome these limitations, our goal is to enhance our work by introducing a function that enables the direct importation of file formats such as .docx, .pdf, and excel into the application. Additionally, the current version of our work does not negotiate any communication channel, but we are considering implementing this in future performance.

Currently, our main focus is creating a new method for enhancing security by using the distinct randomness present in images. By utilizing parallel processing and an efficient randomness extractor, images can exhibit a high level of randomness. Our ultimate aim is to create a complete and extremely secure cryptographic system that can be used to protect data.

5. Acknowledgements

Author Contributions: Conceptualization T.A.; methodology T.A.; software A.J. and T.A.; validation A.J. and T.A.; formal analysis T.A.; writing-original draft preparation A.J. and T.A.; writing-review and editing A.J. and T.A.; visualization T.A.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] cloudHQ. (2021, April 6). 50 Years of E-mail. *CISSION*. https://www.prnewswire.com/news-releases/50-years-of-email-301262676.html

[2] Radicaati Team (2021, February 22). E-mail Statistics Report, 2021-2025. *Statista Radicati*, https://www.radicati.com/?p=17209

[3] Ghali, C.; Stubblefield, A.; Knapp, E.; Li, J.; Schmidt, B.; Boeuf, J. (2017, December 27), Application Layer Transport Security, *Google Cloud whitepaper*. https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

[4] Nohe, P. (2018, October 15), Apple, Microsoft, Google Announce Plans to Disable TLS 1.0, TLS 1.1. *hashedout*. https://www.thesslstore.com/blog/apple-microsoft-google-disable-tls-1-0-tls-1-1/

[5] Just, M. Diffie-Hellman Key Agreement. In Van Tilborg H.C.A. (Eds.), *Encyclopedia of Cryptography and Security* (pp. 154-163). Springer. **2005**. https://www.academia.edu/63723313/Springer_Encyclopedia_of_Cryptography_and_Security

[6] Jagannatam, A. (2008), *Mersenne Twister – A Pseudo Random Number Generator and its Variants*. [Technical report, Department of Electrical Computer Engineering, George Mason University].

[7] Kak, A. *Lecture 8: AES: The Advanced Encryption Standard*, Lecture Notes on Computer and Network Security, Purdue University. **2020**. https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf

[8] O S. N.; Vankamanidi S. N. Comparative Analysis of MOD-ECDH Algorithm and Various Algorithms. *Journal of Industrial Engineering & Production Research*. **2020**, *31*(2), 302-308. https://do.org/10.22068/ijiepr.31.2.301

[9] Hamid N.; Nikil D.; Sandip R.; Francesco R.; Indranil B.; Rosario C. Post-quantum Lattice-based Cryptography Implementations: A Survey. *ACM Computing Surveys*. **2019**, *51*(6), 1-41. https://doi.org/10.1145/3292548

[10] Tain, X.; Benkrid, K. (2009, July 29 – August 1). *Mersenne Twister random number generation on FPGA, CPU and GPU NASA/ESA* [Conference session]. 2009 Conference on Adaptive Hardware and System, San Franciso, CA, United States. https://ieeexplore.ieee.org/document/5325420

[11] Vigna, S. (2019, November 14). Is it high time we let go of the Mersenne Twister. *Arxiv*. https://arxiv.org/pdf/1910.06437.pdf

[12] Rimkienė, R. (2022, August 29), What is AES encryption and how does it work?. *Cybernews*, https://cybernews.com/resources/what-is-aes-encryption/

[13] Schwenk, J. *Guide to Internet Cryptography: Security Protocols and Real-World Attack Implications*. Springer. **2022**. https://link.springer.com/book/10.1007/978-3-031-19439-9

[14] Kiltz, E., O'Neill A.; Smith, A. Instantiablity of RSA_OAEP Under Chosen-Plaintext Attacj. *Journal of Cryptology*. **2017**, *30*(1), 889-919. https://doi.org/10.1007/s00145-016-9238-4

[15] Huang, Z.; Liu, S.; Qin, B. Sender Equivocable Encryption Schemes Secure against Chosen-Ciphertext Attacks Revisited. *Journal of Applied Mathematics and Computer Science*. **2015**. *25*(2), 369-389. https://doi.org/10.1007/978-3-642-36362-7_23

[16] Latif, I. H. (2021, August 14). *Time Evaluation Of Different Cryptography Algorithms Using Labview* [Confernce session]. The 4th Postgraduate Engineering Conference in Materials Science and Engineering, Baghdad, Iraq. https://iopscience.iop.org/article/10.1088/1757-899X/745/1/012039

[17] Microsoft (2023, July 31). Preparing for TLS 1.2 in Office 355 and Office 365 GCC. *Microsoft*, https://docs.microsoft.com/en-us/microsoft-365/compliance/prepare-tls-1.2-in-office-365?view=o365-worldwide

[18] Egress (2021, January 17). How to encrypt Outlook e-mails in transit. *egress.* https://www.egress.com/blog/email-encryption/encrypt-outlook-emails-in-transit

[19] Microsoft 365 (2023, August 8), E-mail encryption. *Microsoft*, https://docs.microsoft.com/en-us/microsoft-365/compliance/email-encryption?view=o365-worldwide

[20] Posteo (2021, December 12). POSTEO: E-mail, calendar, address, book, notes. *POSTEO*. https://posteo.de/en

[21] Tutanota (2022, March 25). Tutanota Secure e-mail for everybody. *Tutanota*. https://tutanota.com/

[22] Lalabit (2022, June 10). Secure e-mail for the world. *Lavabit*. 2022. https://lavabit.com/

[23] Trustifi, (2022, October 7). Trustifi E-mail Security that is Easy to Deploy, Manage, and Use. *Trustifi*. https://trustifi.com/

[24] Preveil, (2022, August 19). Preveil Simple Secure Compliment. *Preveil*. https://www.preveil.com/

[25] Signal, (2022, Febuary 26). Speak Freely. *Signal*. https://signal.org/en/

[26] Livington, J. D.; Kirubakarn, E.; Johnraja, J. I (2021, August 4). *Implementing Client-Side Encryption for Enforcing Data Privacy on the Web Using Symmetric Cryptography: A Research Paper* [Conference session]. Third International Conference on Information Management and Machine Intelligence, Singapore. https://link.springer.com/chapter/10.1007/978-981-19-2065-3_2

[27] Moon, M. E.; Colonel, M. E. How America Lost its Secrets: Edward Snowden, The Man and The Thief By Edward Jay Epstein New York, N.Y.; Alfred A. Knopf. *Journal of Strategic Security*. **2017**, *10*(1), 143-147. https://www.jstor.org/stable/26466898