УДК 004.492

Application of high-performance calculations for the solution of cryptography problems

Baimuldina N. S.
candidate of pedagogical sciences,
associate professor of the department of Informaton Systems,
Al-Farabi Kazakh National University, al-Farabi av., 71,
tel+7 (727)7472926058 e-mail: baimuldinanaziko@mail.ru
Baisholanova K.S
doctors of economic of economic sciences,
associate professor of the department of Informaton Systems,
Al-Farabi Kazakh National University, al-Farabi av., 71,
tel+7 (727)7026159530 e-mail: karlygash.baisholanova@kaznu.kz
Maksutova B.A,
senior teacher of the department computer science
Al-Farabi Kazakh National University, Kazakhstan, Almaty.
bam.bota@mail.ru,
Baitenova S.A.
senior teacher of the department computer science
Al-Farabi Kazakh National University, Kazakhstan, Almaty.
sbaitenova@mail.ru
Zakariyanova N.B.
senior teacher of the department Informatics and Informatization of Education
Abai Kazakh National Pedagogical University, Kazakhstan, Almaty
znazb@mail.ru

ABSTRACT

This paper addresses the issues is a Cryptography and Information security. Their implementation on C++ programing language and comparison of work. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is plaintext(usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data.

Keywords: Cryptography, information security, encryption algorithm, Microsoft Visual Studio 2013, C++, Key.

1.INTRODUCTION

Computer systems need to store user passwords somewhere. However, storing passwords in plaintext is never recommended, as an attacker could simply find the password file and thus have access to all passwords stored on the machine! Therefore, passwords are first encrypted using strong crypto hash functions. Given a generated crypto hash, it is hard to crack the hash to generate the original plaintext. Therefore they are the natural solution to password storage. Though there are tools to recover passwords from crypto hashes, most of them are sequential and CPU based. As passwords get more complex, these CPU based solutions fail to recover passwords in a reasonable amount of time. Indeed it can take hours or days to recover a password as simple as on some

implementations. I decided to make such a password recovery tool for the MD5 hash using CUDA to see if we could do any better.

As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is [1] Plaintext (usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. This enciphered data can be stored in database, host server and if a 3rd party entity will gain access to this data, he will not be able to recover it back to entire form, as he don't know if the data he gained access to is enciphered by hashing Algorithm, encryption Algorithm. The end result of Encryption Algorithm (how data looks after Encryption) differs from other similar Algorithms, Hashing function results due to appliance of different computation blocks, formulas. And mostly the 3rd party entity must therefore to detect what enciphering Algorithm was used on server, it requires testing of sample data by numerous enciphering Algorithms and Key's which is almost impossible. And nowadays the presence of Encryption in any computer system (application/site/server) is an important feature to distribute and use this system in real life. And currently despite of appliance of any computer system or application the embedding of Encryption Algorithm is a vital part.

Usually the most of online attacks are intended on web servers/e-commerce sites as mostly the transmitting packets are not enciphered, it's easy to deploy not authentic links with forms for entering credit card numbers, data to collect such information. [1]

This cyber attacks intended on collection of data are costly, as in USA at 2014 the loss of E-commerce market was reached up to 20.8 million $. In Kazakhstan, according to official sources for the year 2016, this value climbed to 12 billion 348 million tenge.

As most of users are still uses 'dictionary' passwords, weak passwords (password that contains similar characters) the deciphering of stolen database are becoming time efficient, since the sample data are covered from dictionary and it minimize the time of hacker needed to decipher data. The target of hackers are considered mostly on credit card information, due to easy deployment of not genuine web forms for collecting password, presence of E - commerce sites with disabled enciphering software Considering all above the Implementation and further deployment of Encryption Algorithms becoming more vital and actual for every type of application or web service.

The project area is a Cryptography and Information security. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is Plaintext(usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. This enciphered data can be stored in database, host server and if a 3rd party entity will gain access to this data, he will not be able to recover it back to entire form, as he don't know if the data he gained access to is enciphered by hashing Algorithm, encryption Algorithm. The end result of Encryption Algorithm (how data looks after Encryption) differs from other similar Algorithms, Hashing function results due to appliance of different computation blocks, formulas. And mostly the 3rd party entity must therefore to detect what enciphering Algorithm was used on server, it requires testing of sample data by numerous enciphering Algorithms and Key's which is almost impossible. And nowadays the presence of Encryption in any computer system (application/site/server) is an important feature to distribute and use this system in real life. And currently despite of appliance of any computer system or application the embedding of Encryption Algorithm is a vital part. The aim of diploma work to specify the set of most secure Encryption Algorithm and Implement them in Microsoft Visual Studio environment on C# Programming language. Someone will say that presence of secured network, private VPN are already enough to provide security from 3rd parties

entities.But what if this level of securities are already bypassed on network layers, the core importance to provide security on application layer and make information visible only by certain authorized entities. Usually the most of online attacks are intended on web servers/e-commerce sites as mostly the transmitting packets are not enciphered, it's easy to deploy not authentic links with forms for entering credit card numbers, data to collect such information. This cyber attacks intended on collection of data are costly, as in USA at 2014 the loss of E-commerce market was reached up to 20.8 million $. [1] As most of users are still uses 'dictionary' passwords, weak passwords(password that contains similar characters) the deciphering of stolen database are becoming time efficient, since the sample data are covered from dictionary and it minimize the time of hacker needed to decipher data. The target of hackers are considered mostly on credit card information, due to easy deployment of not genuine web forms for collecting password, presence of e-commerce sites with disabled enciphering software. Therefore between October and May of 2013-2014 a 350000 card information and log in information of 233 million customers of online retail sector in USA was stolen. [1] Considering all above the Implementation and further deployment of Encryption Algorithms becoming more vital and actual for every type of application or web service.

2. MD5 ALGORITHM

In MD5, the input message is broken up into chunks of 512-bit blocks (each with sixteen 32-bit sub-blocks). After a series of operations, MD5 produces a 128-bit message digest with four concatenated 32-bit blocks for the integrity of a _le. To compute the digest of a message, padding bits are appended first to make the messages length congruent to 448, modulo 512, and then the length bits. A 64-bit portion is appended to indicate the length of the actual message. MD5 algorithm operates on a 128-bit state which is divided into four 32-bit words (denoted as A, B, C and D) and initialized. Each 512-bit message block is applied in turn to modify the state. The processing of a message block consists of four similar rounds, each of which is composed of 16 similar operations based on a non-linear function F, modular, addition, and left rotation. At last, MD5s output is produced by cascading A, B, C and D after the final round.

CUDA Programming Model CUDA (Compute Unified Device Architecture) [4] is a GPGPU technology (General Purpose Computing on Graphics Processing Units). Instead of executing an application exclusively on the central processor (CPU), some computational intensive parts of the application can be transferred to the graphic processor (GPU). Using the GPU for high performance computing has been in practice for years already, but the lack of a suitable API made it a painstaking experience for the programmer, formulating his ideas in an API designed for pure graphics programming. In contrast, CUDA uses as API an extension of C or Fortran, which makes general-purpose programming on the GPU a lot easier. However, to program the GPU efficiently, a good knowledge of the internal workings of the GPU is still necessary. Some NVIDIA GPU Internals. An extensive description of hardware details of NVIDIA GPUs can be found at [14, 15], and we will restate some of the main points here in the context of the GPU model used for this paper, the GTX 295. But even for the new GPU architecture, called Fermi, the description will be adequate. At the highest level, the GPU consists of a number of so-called streaming multiprocessors (SM). Each SM contains a fixed number of so-called streaming International Journal of Security and Its The GPU supports hardware-multithreading: a stalled thread waiting for data can be quickly replaced by another thread in exactly one processor (SP) cycle. The threads are organized in bigger units, so-called warps, which typically consist of 32 threads. Warps are the units which are actually scheduled on an SM. They are in turn organized in blocks (cooperative thread arrays (CTAs)) of typical sizes of 128 up to 256 threads or 4 to 8 warps. Threads belonging to the same CTA can communicate via a special and very fast memory area called shared memory. There is no way for the CTAs themselves to communicate: CUDA requires the thread blocks to be independent to allow them to be executed in any order, which leads to portability and scalability of CUDA programs. Using shared memory is in theory as fast as using registers. In practice, some care is advisable,

though. Shared memory is composed of DRAM: each read or write operation has to be followed by a refresh cycle. To allow for maximal parallelism, shared memory is therefore built up of so-called memory banks. As long as different threads of a warp (more exactly, of a half-warp) access different memory banks, there is no problem. Otherwise so-called bank conflicts occur, leading to a contention situation, which reduces performance. Another important fact for the GPU implementation is that threads in the same warp are always synchronized. That means that data written to shared memory by the threads of a warp are visible to all threads in the warp in the next computational step. Finally, the code to be executed on the GPU is called a kernel. It is a C (or Fortran) function which has to be called on the CPU. With older GPU models only one kernel can be active on the GPU at the same moment.

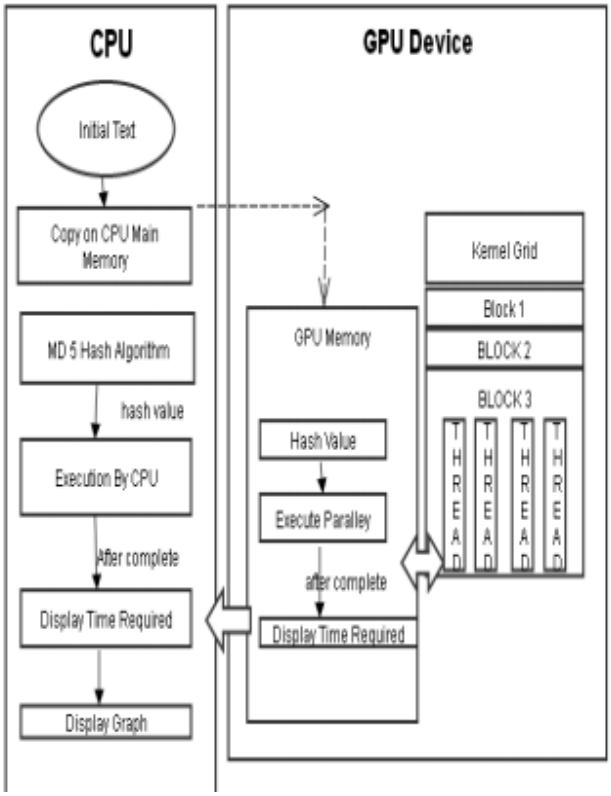

Figure 1 - MD5 System Architecture

There are four steps in parallelized hybrid MD5 encryption algorithm. First, key initialization and expansion are accomplished by MD5 key encryption module and key expansion module in Host (CPU). Thus the hashed and expanded new key is prepared and ready for the initialization procedure. Second, initialization procedure generates Sub-keys, which is also operated by the host (CPU). Third, Sub-keys and input data objects (a number of input data blocks) are copied into GPU Memory. Encryption Module is coded as CUDA kernel functions to encrypt the input message and generate outputs (encrypted message) on GPU. Finally, the encrypted message is copied back to the host and delivered to users as shown in figure 1.

In this report discuss the efficiency of parallel implementations of selected password recovery algorithms. The only reasonable technique for recovering a password from hash is to scan all potential password, compute their hash, and test the coincidence. In general, cryptographic hash functions include integer and binary operations such as: addition modulo power of two, bit shift and rotation, bitwise xor, bitwise or, bit negation and words permutation. All those operations are natively supported by GPU processors. Multiple tests were performed for parallel implementations of password recovery from MD5, SHA-1.The aim of the first series of tests was to compare the performance achieved by Intel core i3 central processing units (CPU).Then multiple tests were performed for password recovery algorithms. The performance of CPU-based and GPU-based versions of MD5, SHA-1, and hash techniques for hashes generation was evaluated. The multi-

threaded implementations of the MD5 and SHA-1 were executed on GPU, each composed on Nvidia GPU. Then compare the performance of CPU-based and GPU-based algorithms for password recovery. Two techniques for hash generation were considered: MD5 and SHA-1. The number of hashes generated per second running MD5 and SHA-1 algorithms by CPU and NVIDIA Gforce740m are presented in chart 2 and chart 7 respectively

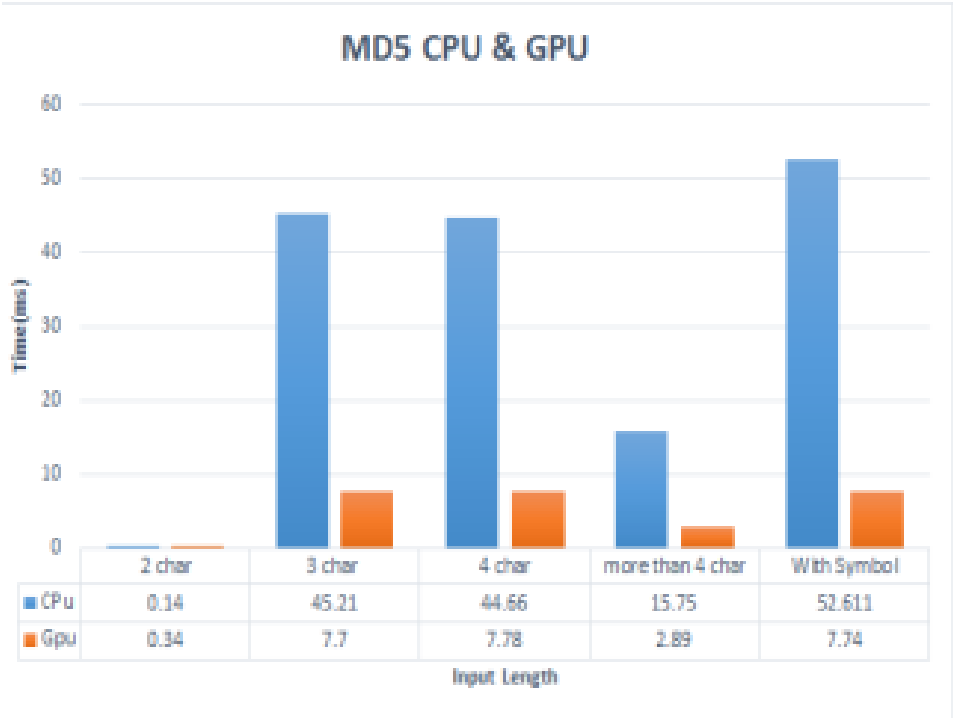

Figure 2 - MD5 Algorithm Results

3. Hash algorithms work

Since the overall contents of a server that can validate passwords are necessarily sufficient to indeed validate passwords, an attacker who obtained a read-only snapshot of the server is in position to make an offline dictionary attack: he tries potential passwords until a match is found. This is unavoidable. So we want to make that kind of attack as hard as possible. Our tools are the following:

Cryptographic hash functions: these are fascinating mathematical objects which everybody can compute efficiently, and yet nobody knows how to invert them. This looks good for our problem - the server could store a hash of a password; when presented with a putative password, the server just has to hash it to see if it gets the same value; and yet, knowing the hash does not reveal the password itself.

Salts: among the advantages of the attacker over the defender is parallelism. The attacker usually grabs a whole list of hashed passwords, and is interested in breaking as many of them as possible. He may try to attack several in parallels. For instance, the attacker may consider one potential password, hash it, and then compare the value with 100 hashed passwords; this means that the attacker shares the cost of hashing over several attacked passwords. A similar optimisation is precomputed tables, including rainbow tables; this is still parallelism, with a space-time change of coordinates.

The common characteristic of all attacks which use parallelism is that they work over several passwords which were processed with the exact same hash function. Salting is about using not one hash function, but a lot of distinct hash functions; ideally, each instance of password hashing should use its own hash function. A salt is a way to select a specific hash function among a big family of hash functions. Properly applied salts will completely thwart parallel attacks (including rainbow tables).

Slowness: computers become faster over time (Gordon Moore, co-founder of Intel, theorized it in his famous law). Human brains do not. This means that attackers can "try" more and more potential passwords as years pass, while users cannot remember more and more complex passwords (or flatly refuse to). To counter that trend, we can make hashing inherently slow by defining the hash function to use a lot of internal iterations (thousands, possibly millions).

We have a few standard cryptographic hash functions; the most famous are MD5 and the SHA family. Building a secure hash function out of elementary operations is far from easy. When cryptographers want to do that, they think hard, then harder, and organize a tournament where the functions Figureht each other fiercely. When hundreds of cryptographers gnawed and scraped and punched at a function for several years and found nothing bad to say about it, then they begin to admit that maybe that specific function could be considered as more or less secure. This is just what happened in the SHA-3 competition. We have to use this way of designing hash function because we know no better way. Mathematically, we do not know if secure hash functions actually exist; we just have "candidates" (that's the difference between "it cannot be broken" and "nobody in the world knows how to break it").

A basic hash function, even if secure as a hash function, is not appropriate for password hashing, because:

- it is unsalted, allowing for parallel attacks (rainbow tables for MD5 or SHA-1 can be obtained for free, you do not even need to recompute them yourself);

- it is way too fast, and gets faster with technological advances. With a recent GPU (i.e. off-the-shelf consumer product which everybody can buy), hashing rate is counted in billions of passwords per second.

So we need something better. It so happens that slapping together a hash function and a salt, and iterating it, is not easier to do than designing a hash function -- at least, if you want the result to be secure. There again, you have to rely on standard constructions which have survived the continuous onslaught of vindicative cryptographers.

System Input Form Design and Sample Input This is the dialog box that enables users to encode their files and thus specify how data are entered into the system. Figure 3. shows the input dialog form.

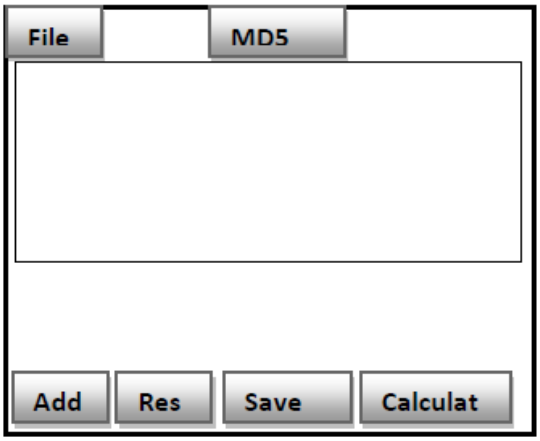

Figure 3 - Input Form

The input process entails the user to browse for the file to be hash and to also verify if it is the actual file selected. The user simply click “Browse for the Hash file” and then click “verify” as shown in Figure 4.

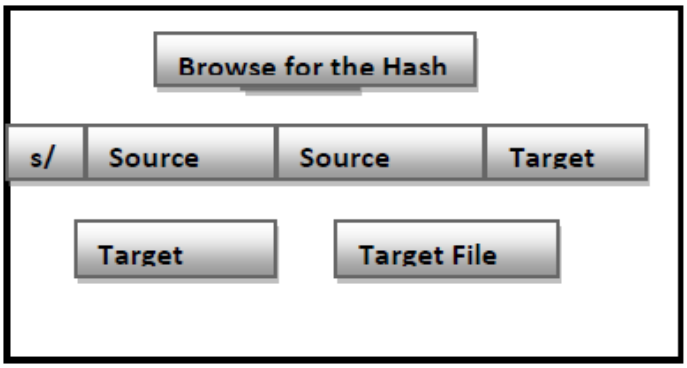

Figure 4- Sample System Input

The System Output Design and Sample Form Similarly, another interface is design to display the result of the verification exercise, which comprises of the name of the source file, source/initial hashes, targeted/new hashes and status of the verification (Figure1) The output generated by the system based on the input of Figure 1 is shown in Figure 5.

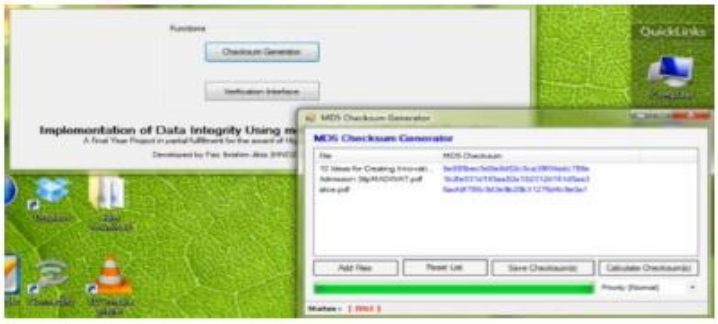

Figure 5 - Output Form

Sample System Output Development tools Utilized The data integrity checking system was developed using several technologies which are technically sufficient to build a reliable and well-maintained platform. Below are the tools that were utilized:

- Microsoft visual studio2010: is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and

- web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver light.

System Requirements The major requirements for the successful implementation and operation of the system are listed below:

- Pentium 4 processor or higher
- 1 gb RAM or higher
- .Net Framework 3.5 or higher
- USB 2.0 port or higher User Operational Manual for the Data integrity checking system guides the user on how he/she will go about using the Software. Details Once the software is launched from desktop The Main/Welcome Screen appears on screen. The Graphical User Interface (GUI) of the main screen has the following: [1] Checksum Generator Button – when clicked, it enables the user to insert the file(s) that will be hashed [2] Verification interface Button – On clicking this button, the verification interface is open, where two button are made available for the user, the browse button and verify button. on clicking the browse button, a window is open for the user to choose the desired hash file for comparison. After selecting the hash files, verify button is clicked which execute the comparison and display the result. [3] Exit Button – This button exit the interface once clicked.

Ensuring the integrity of files transferred from one department/unit to the other has been a major concern in most organizations, institutions, businesses and industries today. Hard copies of documents are carried manually from one department to another within the same organisation. Even with the advent of electronic mails, security of document is left in the hands of the owner of such document. In most situation, the security provided by the network architecture in cases of electronic mails may not be enough to ensure a secured transmission of document. It is even worst when the medium of manual. MD5 is one among various algorithms that can be used to implement a system that will ensure the integrity of document being transferred over a connection from one department/unit to the other.This research therefore focuses on the design of software that utilises the concept of the MD5 algorithm to overcome the problems of document management of the above institution of higher learning. The system is composed of two modules with two different functionalities. The first module is called the Hash Generation module, which is responsible for hashing a file (generate the checksums/hashes of the file) before sending it. The second module, which is called the Verification Module, receives the encrypted file, performs the verification function; and make comparison between the source file and the target file and a status report is generated at the end of the comparison, which determine whether what was actually sent has been received without any alteration or modification. In no small way, the system has actually be helpful to the institution since its implementation.

4 USED TECHNOLOGIES

Enciphering Algorithms was built on Microsoft Visual Studio 2013 Express as a console application in C++ Programming language.

4.1 Microsoft Visual Studio 2013 Express

Microsoft Visual Studio 2013 Express is Integrated Development Environment (IDE) for C++/C++ programming language. The key features of Microsoft Visual Studio 2013 is the work with widget elements as the Qt (IDE) from Nokia which works only with C++ Programming language, Microsoft Visual Studio 2013 Express supports more Programming languages and build options (PC, Windows Phone, Tablets) all corresponding widgets can work both on mobile devices and PC's which reduces the time needed for porting application on another platform. And what most important is that end user will not need to spend time on knowing new interface, widgets as application will be look graphically similarly as on PC. Also the Microsoft Visual Studio 2013 allows to quickly port console to form application with minimum time spent on Code changes. The running instance of program can be deployed on any CPU (if the work conducted on multi core CPU), optimized debugger (which allows to see the state of all register, their contents at the time of interrupt occurs, with out coding additional function for returning state of all variables) due to this debugger it is possible to quickly reveal and destroy occurred error. The running instance of program runs in the same window as the code there is no needs to write separate comments for compiling the instance of program. Execution of program starts in the same window as the code. This features allows to concentrate on code. Also the inside benchmarking tools can analyse the running time of program and to see how much of memory is being allocated to running instance. The most important advantage of Microsoft Visual Studio IDE is the improved emphasizing options in code editor which is allowing to enclose unneeded functions, classes for faster and convenient navigation. And the major advantage is that Microsoft Visual Studio Express IDE is free. Such flexibility was the core factor in selecting IDE for code.

4.2 C++ Programming languages

Unlike as Python, C++, Java C++ out of box intended to use with Byte variables. All Byte variables can be viewed as in HEX format, Decimal format without calling massive functions requiring manual conversion. The availability of Byte Converting function (Decimal to Binary, Hexadecimal to Binary, etc.) was the major fact in selecting programming language. Also C++ have built in native function for Binary operators (& AND, || OR, ^ XOR, ~ Inversion). This was important as all Enciphering Algorithms are intended to work with Byte data, C++ built in functions

for converting between data types allowed to minimize time needed for implementation of Algorithm and concentrate development time on Methods, Key logic.

5.CONCLUSION

Overall the project was a success. We were able to create an application that can recover most passwords from an MD5-hashed value with ease. Our hash generation rate is extremely close to the professional standard, and we can recover not only most passwords today, but also passwords that will be used in the future that follow well-known patterns. Even for completely random passwords, we were successfully able to come up with a brute force solution that can recover passwords up to a certain length in a reasonable amount of time. Our only regret is that we were not eligible for the iPad.

Conclusively, the developed system would solve the associated problems with the traditional/manual system. Delay and time consumption will surely be eliminated with the documents being sent over a secure network connection, and an effective & efficient integrity checking system that will guaranty the authenticity of the received document.Similarly, with the successful implementation of this system, confidentiality of messages will definitely be maintained as only authorised personnel of an organization can have access to their systems, unlike the manual system in which junior staffs are charged with the responsibility of delivering mails in the physical sense, so this makes it more prone for the message to be exposed to an outsider who does not have the authority of seen such a confidential message. The prototype of this system has been tasted with different forms of data and it is achieved that the system successfully generates hashes/checksums of a given data, stores them and also performs the verification test and finally give a status report about the message.

The main work was conducted on defining the set of most complicated and Cryptography endurance Encryption Algorithm. Analysing the speed of Algorithms, analysis of it's function and enciphering methods. Comparing their major difference and implementing them in correct order to get matching test vectors. A final work is implemented Enciphering Algorithms written on C# Programming Language in console application format, which provides output of all register states, round outputs. Optionally I suggest that by Graph Algorithms it is possible to mix Byte order in Ciphertext, so that even a 3rd party entity would get the Ciphertext, he would be not able to decipher the Ciphertext by selection method (first trying to start decipher Ciphertext by Key 0x00000000000000000000000000000000 and up to 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF until all combination have been looked. But even thought if even a right Key will be recovered the original Plaintext would be not seen, because the Byte order was mixed. In example of Serpent Non Bitslice Encryption Algorithm

Table 1.50 Enciphering test sample of Serpent Encryption Algorithm Non-bitslice

	Plaintext	Key	Ciphertext	Result
	0x0b1516f9050b0c6262798dbe5b6163bf	0x657377b5c9d2d7fb798d97bf65737779bfc0c7ca696f70776579bfc0f9fbfcff	0xa39ec6c8242efacfe769eaee6ba5ff19	Pass

We see that Ciphertext is 0xa39ec6c8242efacfe769eaee6ba5ff19, and it should be exact the same for successful Decryption. So of course we can get a random numbers to mix the Ciphertext, but in terms of the problem we can use the path (traversal path output of BFS/DFS Algorithms, Shortest path of Dijkstra Algorithm, etc.) to mix Byte order.

For example if a traversal path of BFS/DFS Algorithm is 1,5,7,8,9 and Ciphertext is 0xa39ec6c8242efacfe769eaee6ba5ff19, that means that Byte 1 will be on it's position, Byte 5 will move on 2nd position, Byte 7 will move on 3rd position, etc. In general we just swapping Bytes by its positions and finally I get this new mixed Ciphertext 0xa324facfe72ec6c89e69eaee6ba5ff19. It is

considerably differs from original Ciphertext, giving zero chance for Deciphering it by 3rd party entity.

Their implementation on C# programing language and comparison of work. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is Plaintext(usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. The project area is a Cryptography and Information security. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is Plaintext(usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. This enciphered data can be stored in database, host server and if a 3rd party entity will gain access to this data, he will not be able to recover it back to entire form, as he don't know if the data he gained access to is enciphered by hashing Algorithm, encryption Algorithm. The end result of Encryption Algorithm (how data looks after Encryption) differs from other similar Algorithms, Hashing function results due to appliance of different computation blocks, formulas. And mostly the 3rd party entity must therefore to detect what enciphering Algorithm was used on server, it requires testing of sample data by numerous enciphering Algorithms and Key's which is almost impossible. And nowadays the presence of Encryption in any computer system (application/site/server) is an important feature to distribute and use this system in real life. And currently despite of appliance of any computer system or application the embedding of Encryption Algorithm is a vital part. The aim of diploma work to specify the set of most secure Encryption Algorithm and Implement them in Microsoft Visual Studio environment on C# Programming language. Someone will say that presence of secured network, private VPN are already enough to provide security from 3rd parties entities.But what if this level of securities are already bypassed on network layers, the core importance to provide security on application layer and make information visible only by certain authorized entities. Usually the most of online attacks are intended on web servers/e-commerce sites as mostly the transmitting packets are not enciphered, it's easy to deploy not authentic links with forms for entering credit card numbers, data to collect such information. This cyber attacks intended on collection of data are costly, as in USA at 2014 the loss of E-commerce market was reached up to 20.8 million $. [1] As most of users are still uses 'dictionary' passwords, weak passwords(password that contains similar characters) the deciphering of stolen database are becoming time efficient, since the sample data are covered from dictionary and it minimize the time of hacker needed to decipher data. The target of hackers are considered mostly on credit card information, due to easy deployment of not genuine web forms for collecting password, presence of e-commerce sites with disabled enciphering software. Therefore between October and May of 2013-2014 a 350000 card information and log in information of 233 million customers of online retail sector in USA was stolen. [1] Considering all above the Implementation and further deployment of Encryption Algorithms becoming more vital and actual for every type of application or web service.

The project area is a Cryptography and Information security. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is [1] Plaintext (usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. This enciphered data can be stored in database, host server and if a 3rd party entity will gain access to this

data, he will not be able to recover it back to entire form, as he don't know if the data he gained access to is enciphered by hashing Algorithm, encryption Algorithm. The end result of Encryption Algorithm (how data looks after Encryption) differs from other similar Algorithms, Hashing function results due to appliance of different computation blocks, formulas. And mostly the 3rd party entity must therefore to detect what enciphering Algorithm was used on server, it requires testing of sample data by numerous enciphering Algorithms and Key's which is almost impossible. And nowadays the presence of Encryption in any computer system (application/site/server) is an important feature to distribute and use this system in real life. And currently despite of appliance of any computer system or application the embedding of Encryption Algorithm is a vital part.

Usually the most of online attacks are intended on web servers/e-commerce sites as mostly the transmitting packets are not enciphered, it's easy to deploy not authentic links with forms for entering credit card numbers, data to collect such information. [2]

This cyber attacks intended on collection of data are costly, as in USA at 2014 the loss of E-commerce market was reached up to 20.8 million $. In Kazakhstan, according to official sources for the year 2016, this value climbed to 12 billion 348 million tenge.

As most of users are still uses 'dictionary' passwords, weak passwords (password that contains similar characters) the deciphering of stolen database are becoming time efficient, since the sample data are covered from dictionary and it minimize the time of hacker needed to decipher data. The target of hackers are considered mostly on credit card information, due to easy deployment of not genuine web forms for collecting password, presence of E - commerce sites with disabled enciphering software Considering all above the Implementation and further deployment of Encryption Algorithms becoming more vital and actual for every type of application or web service.

The project area is a Cryptography and Information security. As nowadays most of services becomes electronic, bank needs to store information in electronic format, provide online transaction, all online card service providers host online transaction numerous number of times. There should be a way to safely store such information and despite of granting access of 3rd parties this information should be not deciphered. Cryptographic Algorithms can solve this issues. Their main function is to convert given information which is Plaintext(usually it is the entire form of data, e.g. password) to sequence of character which are completely differs from entire data. This enciphered data can be stored in database, host server and if a 3rd party entity will gain access to this data, he will not be able to recover it back to entire form, as he don't know if the data he gained access to is enciphered by hashing Algorithm, encryption Algorithm. The end result of Encryption Algorithm (how data looks after Encryption) differs from other similar Algorithms, Hashing function results due to appliance of different computation blocks, formulas. And mostly the 3rd party entity must therefore to detect what enciphering Algorithm was used on server, it requires testing of sample data by numerous enciphering Algorithms and Key's which is almost impossible. And nowadays the presence of Encryption in any computer system (application/site/server) is an important feature to distribute and use this system in real life. And currently despite of appliance of any computer system or application the embedding of Encryption Algorithm is a vital part. The aim of diploma work to specify the set of most secure Encryption Algorithm and Implement them in Microsoft Visual Studio environment on C# Programming language. Someone will say that presence of secured network, private VPN are already enough to provide security from 3rd parties entities.But what if this level of securities are already bypassed on network layers, the core importance to provide security on application layer and make information visible only by certain authorized entities. Usually the most of online attacks are intended on web servers/e-commerce sites as mostly the transmitting packets are not enciphered, it's easy to deploy not authentic links with forms for entering credit card numbers, data to collect such information. This cyber attacks intended on collection of data are costly, as in USA at 2014 the loss of E-commerce market was reached up to 20.8 million $. [1] As most of users are still uses 'dictionary' passwords, weak passwords(password that contains similar characters) the deciphering of stolen database are becoming time efficient, since the sample data are covered from dictionary and it minimize the time of hacker needed to decipher data. The target of hackers are considered mostly on credit card

information, due to easy deployment of not genuine web forms for collecting password, presence of e-commerce sites with disabled enciphering software. Therefore between October and May of 2013-2014 a 350000 card information and log in information of 233 million customers of online retail sector in USA was stolen. [1] Considering all above the Implementation and further deployment of Encryption Algorithms becoming more vital and actual for every type of application or web service.

REFERENCES

1. The Heritage Foundation, "Cyber Attacks on U.S. Companies in 2014" By Riley Walter, (27th October 2014). URL: http://www.heritage.org/research/reports/2014/10/cyber-attacks-on-us-companies-in-2014

2. Federal Information Processing Standards Publication, "Advanced Encryption Standard", (26th November 2001). URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

3. Serpent home page, "A Candidate Block Cipher for the Advanced Encryption Standard", By Ross Anderson, Eli Biham and Lars Knudsen, (24th March 2001). URL:
http://www.cl.cam.ac.uk/~rja14/Papers/serpentcase.pdf

4. Wikipedia online encyclopedia, "Efficient implementation of Serpent Encryption Algorithm". URL:
https://ru.wikipedia.org/wiki/Serpent

5. Royal Holloway University of London, "The Cryptoanalysis of FEAL-4 with twenty choosed Plaintexts", By Sean Murphy. URL:
http://www.isg.rhul.ac.uk/~sean/feal.pdf

6. E-nigma Encryption Systems, "RSA Encryption Algorithm". URL:
http://www.e-nigma.ru/stat/rsa/

7. The University of Auckland, New Zeland, Department of Computer Science, "Dijkstra Algorithm", By John Morrison. URL:
https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html

8. Wikipedia online encyclopedia, "Kruskal Algorithm". URL:
https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

9. Wikipedia online encyclopedia, "Prima Algorithm". URL:
https://en.wikipedia.org/wiki/Prim%27s_algorithm

10. Berkeley University, Electrical Engineering & Computer Sciences, "Disjoint sets", (16th April 2014). URL:
https://www.cs.berkeley.edu/~jrs/61b/lec/33

11. UCI, Donald Bern, School of Information & Computer Sciences, "Breadth first search and Depth first search", (15th February 1996). URL:
https://www.ics.uci.edu/~eppstein/161/960215.html

12. Kapro, Trust and Safe Electrics, "Example of light level computation", URL: http://www.kapro.ua/articles/15/

13. National Security Agency of the United States of America, "Suite B Cryptography", (25th September 2014). URL:
https://www.nsa.gov/ia/programs/suiteb_cryptography/

14. Karazhat consulting, "Social tax, contributions". URL:
http://www.karazhat.kz/articles/sotsialniy-nalog