CryptoChem for Encoding and Storing Information Using Chemical Structures

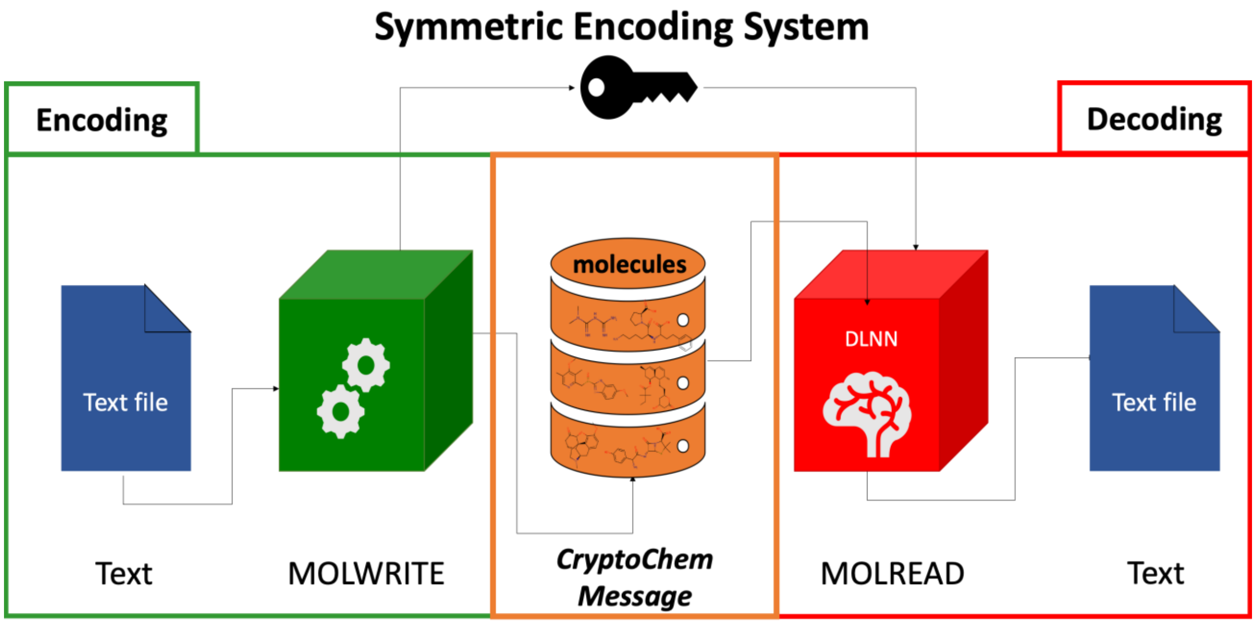
Phyo Phyo Kyaw Zin[1,2]†, Xinhao Li[1,2]†, Dhoha Triki[1,2], and Denis Fourches[1,2]*

(† equal contribution)

[1] Department of Chemistry, North Carolina State University, Raleigh, NC, USA.

[2] Bioinformatics Research Center, North Carolina State University, Raleigh, NC, USA.

* To whom correspondence should be sent. Email: dfourch@ncsu.edu



ABSTRACT

This study presents CryptoChem, a new method and associated software to securely store and transfer information using chemicals. Relying on the concept of Big Chemical Data, molecular descriptors and machine learning techniques, CryptoChem offers a highly complex and robust system with multiple layers of security for transmitting confidential information. This revolutionary technology adds fully untapped layers of complexity and is thus of relevance for different types of applications and users. The algorithm directly uses chemical structures and their properties as the central element of the secured storage. QSDR (Quantitative Structure-Data Relationship) models are used as private keys to encode and decode the data. Herein, we validate the software with a series of five datasets consisting of numerical and textual information with increasing size and complexity. We discuss *(i)* the initial concept and current features of CryptoChem, *(ii)* the associated MOLREAD and MOLWRITE programs which encode messages as series of molecules and decodes them with an ensemble of QSDR machine learning models, *(iii)* the Analogue Retriever and Label Swapper methods, which enforce additional layers of security, *(iv)* the results of encoding and decoding the five datasets using CryptoChem, and (v) the comparison of CryptoChem to contemporary encryption methods. CryptoChem is freely available for testing at https://github.com/XinhaoLi74/CryptoChem

1. Introduction

As the amount of data produced worldwide is growing exponentially, the need to reliably, durably, and securely store and transfer that data has never been so critical. However, data storage is at a pivotal crossroad. Physical storage devices such as optical drives and tapes are not only reaching their technical limit in terms of capacity and density of storage but also in terms of durability. Also on the rise is the need for disrupting encryption methods based on advanced technologies requiring highly technical skills and/or appropriate equipment. To solve these problems, next-generation DNA storage[1,2] has been developed to encode and store enormous amounts of information on DNA molecules[3]; but this technology suffers many practical challenges, especially when it comes to obfuscation protocols[4]. Meanwhile, the chemical space is estimated to be filled with 10^{60} unique small molecules that can be characterized by properties directly computed from their two-, three-, or even four-dimensional structures and conformations[5,6]. There has been a few attempts to use different subsets of chemicals for establishing novel steganography and/or cryptography methods, especially with dyes[7]. But as of today, to the best of our knowledge, there is no available technology capable of exploiting the chemical space to directly store information in a secured, reproducible, and efficient way.

Herein, we propose to use chemical structures and their properties as the central element for a completely novel data encoding method. It is based on the high complexity and uniqueness of the thousands of structural characteristics and properties that can be computed for every single molecule of the chemical universe. As the chemical universe is estimated to be in the order of 10^{60} molecules[5], it is nearly impossible to enumerate all possible molecules and their properties using a brute force algorithm. Therefore, storing information using chemicals could simultaneously offer very high levels of storage density[8] and security. Ultimately, this approach could also be used to physically store encryption keys or any other information in physical storage technology (*e.g.*, direct encoding by chemicals packaged within a DNA storage cell[9]).

Moreover, for the past decade, robust quantitative structure-activity relationship (QSAR) models[10–13] have been built to predict very specific physical, chemical, and biological endpoints of compounds. Those QSAR models are based on the hypothesis that similar compounds have similar properties. But chemicals first need to be encoded into numerical data being fully amenable to computer-based calculations. To do so, we use molecular descriptors that are well-defined, reproducible, interpretable, and numerical parameters directly and solely computed from the chemical structure of a compound. For a given chemical, thousands of descriptors can be computed solely based on its two-dimensional structure, thousands more based on its three-dimensional structure (*i.e.,* one 3D conformation of that compound). Importantly, those 3D conformation-dependent descriptors can also be computed for thousands of 3D conformations of that same particular compound (*e.g.,* time-dependent descriptors computed from molecular dynamics trajectories[6,14]). Overall, every chemical can be characterized by tens of thousands of numerical descriptors directly computed *in silico*.

For this study, we developed *Quantitative Structure-Data Relationships* (QSDR) models that use machine learning to establish non-linear, quantified links between computed molecular descriptors and numerical/textual data. A collection of QSDR models was built using several types of machine learning techniques (*e.g.,* deep learning neural networks, random forests) and families of 2D, 3D structural descriptors and fingerprints directly computed from various series of molecules. In this CryptoChem project, we designed, implemented, and validated two *proof-of-concept* programs, MOLWRITE and MOLREAD, enabling the encoding and retrieval of information using series of molecular structures. MOLWRITE program stores and encodes a given message in the molecular CryptoChem format, whereas the MOLREAD program decodes a given CryptoChem message using the right QSDR model. The feasibility and capabilities of this innovative technology have been tested on a series of five different datasets consisting of numerical and textual information. To have a better understanding of CryptoChem in relation to other popular encoding algorithms, we also compared it to contemporary methods such as block and stream ciphers on the basis of features, weaknesses and strengths.

2. Methods

2.1. Overview of CryptoChem

CryptoChem is comprised of two major components: (1) MOLWRITE and (2) MOLREAD. The former encodes textual/numerical information into *in silico* molecules, and the later decodes the molecules back into the original textual/numerical message. The overview of CryptoChem encryption and decryption algorithm is provided in **Figure 1**.

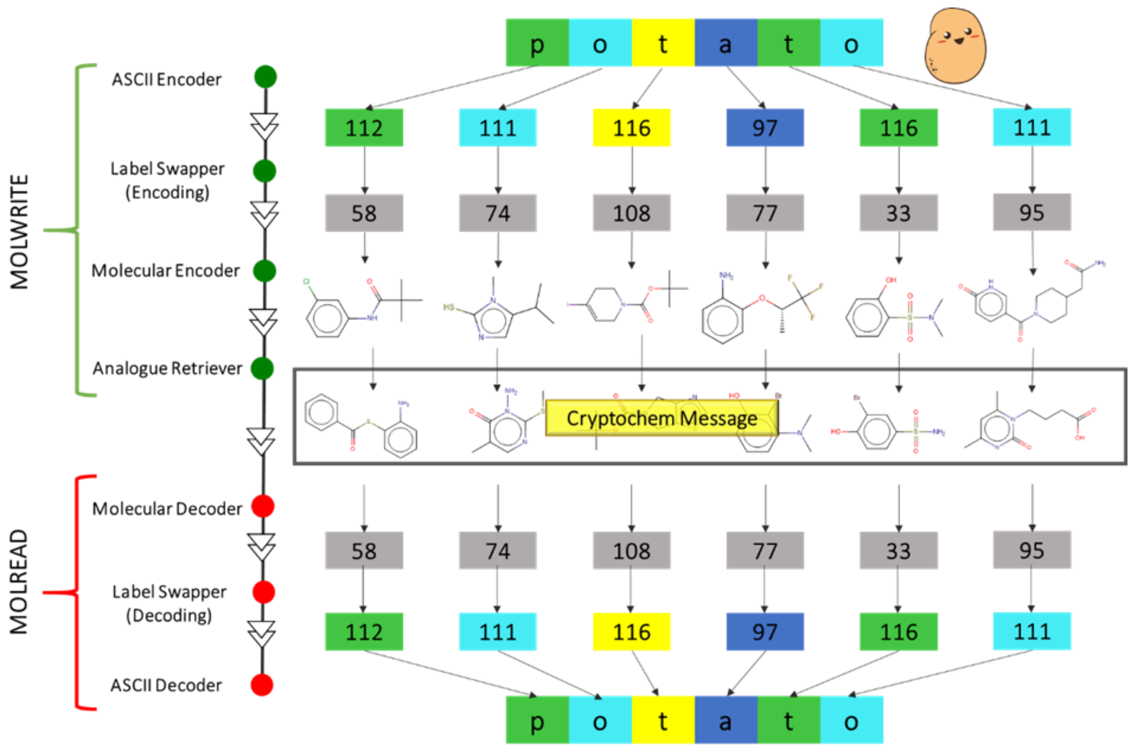


Figure 1. Simplified workflow of the CryptoChem Algorithm.

MOLWRITE (the encryption part) is composed of four major functions: ASCII Encoder, Label Swapper (LS) Encoder, Molecular Encoder (ME), and Analogue Retriever (AR). First, the input message is encoded into digits using ASCII Encoder (ection 2.1. ASCII Encoder). In the current version of the method/software, the algorithm can encode the first 128 characters in the standard ASCII table[15]. To introduce an extra layer of protection in the algorithm, the digits are then transformed into new digits using Label Swapper (LS, see section 2.2. Label Swapper). Next, Molecular Encoder (ME, see section 2.3. Molecular Encoder) is applied to replace each value by a virtual reference chemical which has been tagged with labels. Then, an additional layer of security and confusion is added into the algorithm with the use of Analogue Retriever (AR, see section 2.4. Analogue Retriever). AR replaces the previously *tagged* reference molecules with new chemical analogues which have no explicit tags and/or labels associated with them (also defined as the reference set). These new molecular analogues are used to generate the encoded message, also known as CryptoChem message that carries the original textual/numerical information in encoded molecular format (SMILES). The Simplified Molecular-Input Line-Entry System (SMILES) encodes the molecular structures as strings of text.[16]

MOLREAD (the decryption part) is comprised of three major functions: Molecular Decoder, Label Swapper (LS, Decoder) and ASCII Decoder. Molecular Decoder converts the molecules from CryptoChem message into digits. Then, LS (Decoder) is applied to transform these digits back into the original digits. These digits are then decoded into characters from the initial message using ASCII Decoder.

In the following sections, we will explain each function accordingly.

2.1. ASCII Encoder

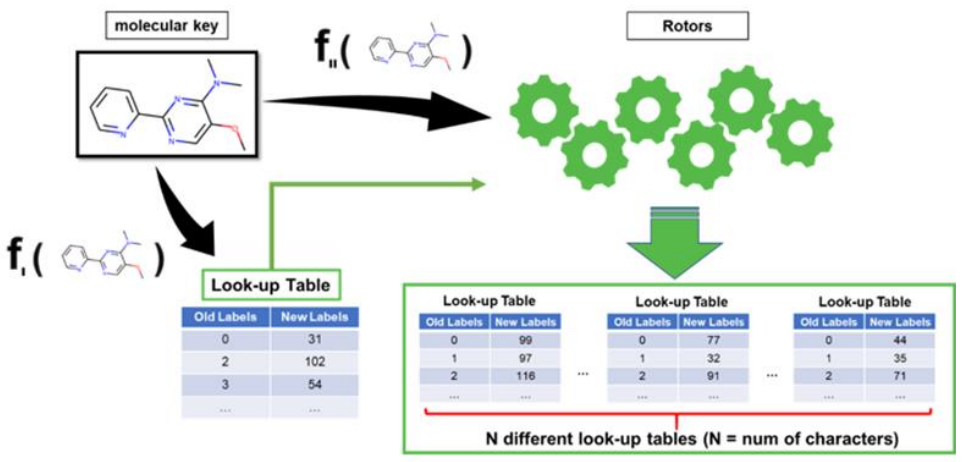
ASCII stands for American Standard Code for Information Interchange. ASCII encodes a numeric value to different characters and symbols. For example, the ASCII value of the character "d" is 100. The complete ASCII table can be found at https://theasciicode.com.ar/. In this study, the original character "d" is converted to the ASCII value of 100. Label Swapper is then applied to switch 100 to a different number (*e.g.*, 91). The new number is then encoded with a chemical molecule from the reference set. For the current version of the method, the algorithm can encode the first 128 characters in the standard ASCII table.

2.2. Label Swapper

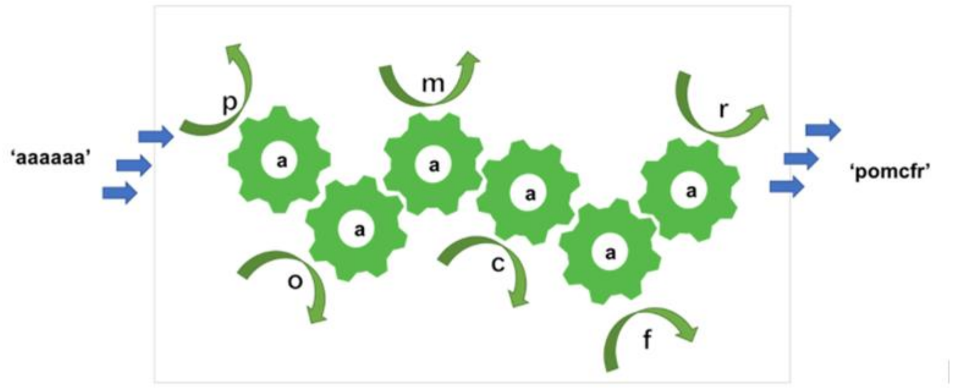
Label swapper (LS) employs a permutation key (also known as the *molecular key*) and 128 (pre-shared with message sender and receiver) *neighbor* molecules, to switch the ASCII digits to new digits. The algorithm of LS is directly inspired by the famous *Enigma* machine that was used by the Germans during World War II[17]. Instead of using the real keyboard and rotors to encrypt the information, LS uses the permutation key and the *neighbor* molecules to generate a virtual look-up table for each ASCII digit to change from one digit to another. For each neighbor molecule, we assigned a digit (0-127) to it. During label swapping, the distances between the permutation key and neighbor molecules are computed, and neighbor molecules are ranked according to the computed distances. The virtual look-up table is formulated based on the originally assigned digits and the computed distance ranks of the neighbor molecules. The Label swapper is an important component to ensure a higher security level in CryptoChem.

Essentially, the native CryptoChem system (without Label Swapper) can be seen as a form of substitution encryption. The machine learning model acts as a substitution key. The assumption of QSDR is that 'similar' molecules (characterized by the specific molecular descriptors used) will be used to represent the same ACSII code. The trained QSDR machine learning models learn the patterns of the text-molecule relationship so that it can map the molecules to the text.

LS is designed to resist several major cryptanalysis techniques against the substitution encryption part in case the machine learning model itself is compromised by the adversarial. The concept of LS is shown in **Figure 2**. The central part of Label Swapper is the molecular key. It is of high importance to underline that it can be any molecule chosen from the whole molecular universe (~10^{60}). Based on same molecular properties/descriptors of the molecular key, an initial look-up table and some rotors are generated by one or several mathematical functions (**Figure 2a**). The look-up table maps each ACSII code to another. Every time encoding a new text, the rotors will generate a new look-up table. For example, the text 'aaaaaa' would be changed to 'pomcfr'. In CryptoChem system, the molecular key acts as a permutation key. In the encoding process (MOLWRITE), the original text is changed to a 'new' text before translating to the molecules. In the decoding process (MOLREAD), the text decoded by the machine learning model will be further translated back to the original text by the Label Swapper. To fully "crack" the Label Swapper, the adversarial would thus need to know (1) the exact molecular key; (2) the exact set of molecular properties/descriptors used for computing the initial look-up table and rotors; (3) the exact set of functions to compute the initial look-up table and rotors; and (4) how the rotors work.



(a)



(b)

Figure 2. Graphical Illustration of Label Swapper.

2.3. Molecular Encoder

The *reference set* contains 128 clusters of molecules with tagged digits. Molecular Encoder (ME) takes the output from the previous step of LS (see Figure 1); which are numerical digits. Based on these digits, ME randomly samples a chemical cluster from the *reference set* and picks a molecule from that cluster. Hence, in this step, digits from LS are encoded into chemical molecules.

2.4. Analogue Retriever

As the name suggests, Analogue Retriever (AR) aims at identifying and retrieving "analogues" from a very large external set (millions randomly chosen and/or generated among 10^{60} chemicals) of molecules and replaces a target reference molecule. It is incorporated into MOLWRITE to add an additional layer of security in encoding the CryptoChem messages. In this disclosed case study, we split it into two datasets: *reference set* (a small set of chemicals with tagged labels) and *analogue set* (a large chemical space containing multiple clusters of chemicals). *Reference set* contains molecules with associated cluster labels, and *analogue set* contains multiple clusters (*e.g.,* thousands) of molecules without any labels attached to them. AR is an essential part of MOLWRITE in improving the security level by obscuring the relationships between labels and molecules even more. Without AR in MOLWRITE, chemicals could be potentially mapped with labels.

When MOLWRITE program encodes texts into molecules, all these molecules are taken from the *reference set*. Cryptochem messages encoded with molecules directly retrieved from *reference set* could be vulnerable to security breach since having access to *reference set* enables not only encoding but also decoding CryptoChem messages through SMILE matching and identifying their cluster labels. Thus, it is important to devise a strategy to ensure that chemicals used in the CryptoChem messages are not easily traceable to the original labels associated with them. Therefore we developed and implemented Analogue Retriever (AR) which made it impossible to directly decode the message as the same compound is never used twice to encode the same character in the same message or in difference messages.

In this method, the molecules with classified labels in the *reference set* are not actually used in encoding CryptoChem messages; instead, they are only used as *reference* molecules for picking compounds from an extremely large external set which, in our case, is called the *Analogue*

Set. The highlight of applying DNN models (see next sections) in CryptoChem is that well-trained DNN models can, however, predict the right clusters for these analogues.

2.5. Implementation of Analogue Retriever

We incorporated multistage sampling (clustering, stratified sampling) into designing AR to make the process of searching and selecting analogues efficient. There is certainly ample room for optimizing the algorithm and integrating parallel computing on GPUs to further improve the efficiency and runtime. Currently, the software can run on multiple CPUs of a standard desktop computer. The scheme of AR is provided in **Figure 3**.

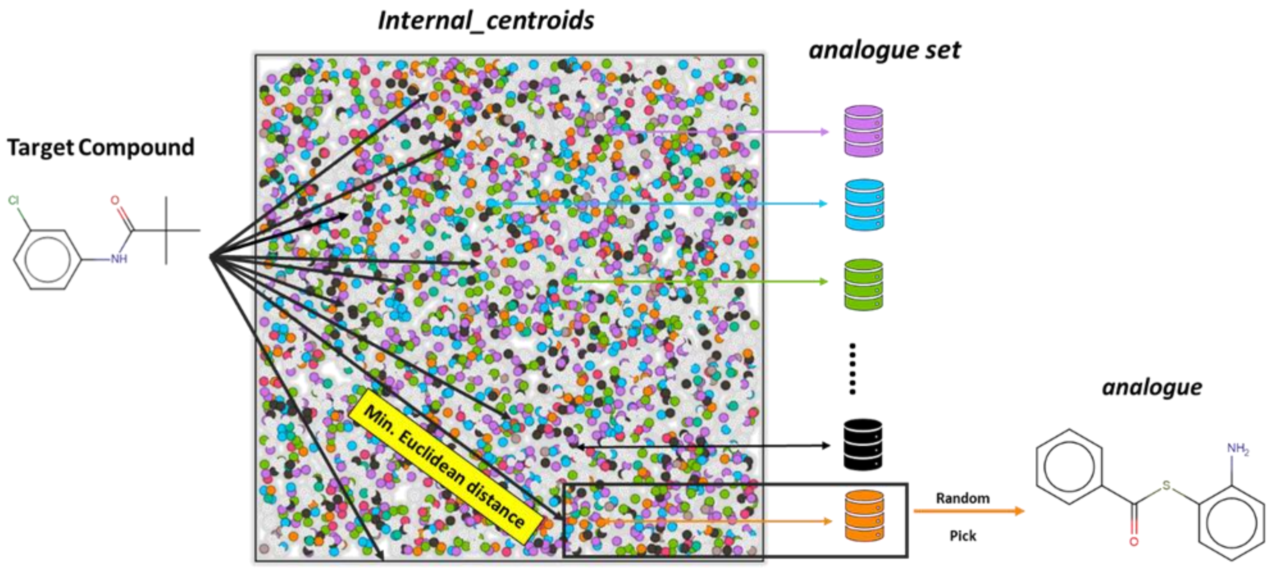


Figure 3. Analogue Retriever scheme for replacing a reference molecule with an analogue from the Analogue Set.

AR takes the output molecules from Molecular Encoder (**Figure 1**) generated initially with the *reference set*. For readers' convenience, we will refer to them as target molecules from now on. These *reference* molecules are later replaced with new molecules from the *analogue set*. Then, AR can compute the Euclidean distance (among other metrics) between each target molecule and multiple centroids (hundreds or thousands) from the list of *internal_centroids*. The closest centroid is chosen based on the smallest Euclidean distance (and/or other metrics). To make this process efficient, we have already extracted all the centroids from all selected chemical clusters and assigned codenames to each centroid beforehand. Once the closest centroid is identified, the code name can be extracted. Using the code name, the chemical cluster is identified, and an *analogue* is randomly selected from that chemical cluster to replace the target molecule. Finally, the molecules from the original CryptoChem message generated initially with *reference set* are thus, one by one, entirely replaced by molecules from the *analogue set*.

2.6. Molecular Data Preparation

In this disclosed case study, 1 million molecules were randomly selected from the purchasable 'drug-like' molecules in the ZINC15 library[18]. The selected molecules (called V1 library) were used as the training set to develop the deep learning neural network DNN model (*molecular decoder*). The full V1 library was grouped into 128 clusters using k-means algorithm in Scikit-learn package[19] in Python based on 166-bit MACCS keys (but any custom-pool of descriptors could be used for a given user/application). We then extracted a few molecules from each of 128 clusters, assigned them labels 0 to 127, and exported them to the *reference set*. Our machine learning model was trained to associate the molecular MACCS keys with the labels. After removing the *reference set*, several internal clusters within each of these aforementioned 128 clusters were then generated (see **Figure S1**), resulting in thousands of chemical clusters. These chemical clusters, also known as the *analogue set*, have no explicit label or digit associated with them. The centroids from all the internal clusters were assigned code names and saved as *internal_centroids* which is an integral part of AR (see section 2.5. Implementation of Analogue Retriever).

2.7. Model Development

We used the V1 Library as training set to develop the model for our molecular decoder. The disclosed Decoder model was trained using Keras 2.2.2 functional API, TensorFlow backend with GPU acceleration, NVIDIA CuDNN libraries [20,21]. We used RMSprop algorithm to train for 1,000 epochs for all molecules in one batch with default learning parameters. It is composed of seven hidden dense layers activated with RELU. Since it is an integer classification (128 classes) task, we used multilabel classification model with 128 nodes in the output layer with SoftMax activation function, sparse categorical crossentropy loss function, and accuracy as the evaluation metric. We then evaluated it by predicting the clusters for the same dataset, and the accuracy of our model is above ~99.7%. A few outliers (0.3%, ~140 compounds) which were predicted to have wrong cluster labels were simply discarded to ensure the 100% accuracy for *chemical-to-character* and *character-to-chemical* QSDR-based recognition.

2.8. MOLWRITE software

MOLWRITE program takes an input text file and encode it into molecules (SMILES). The workflow is summarized in Figure 1. The input message is first translated to ASCII values (a string of digits). For each digit in the ASCII values, LS encoder (section 2.2. Label Swapper) is then applied to change the original digits to new ones. Based on the new digits, molecules from the *reference set* are randomly picked (*e.g.*, for digit ‘12’, a molecule tagged with label 12 is picked) using ME (section 2.3. Molecular Encoder). Then, AR (section 2.4. Analogue Retriever) replaces the picked molecules with an unlabeled *analogue* molecules from the *analogue set*, generating CryptoChem messages as output. That CryptoChem message can be further treated with modern encryption algorithm (*e.g.*, AES).

2.9. MOLREAD software

MOLREAD program takes a CryptoChem message as input file, which contains molecules (SMILES). Available set of descriptors developed in Python using the RDKit modules are reported in **Table S1**. Chemical descriptors (both the sender and the receiver must pre-share and/or know the type and exact number of descriptors applied) are then computed on these molecules and fed into the machine learning model that was initially trained with V1 library (substitution key; it could be any model trained with any chemical library). The model then predicts and decodes cluster labels for each molecule based on its chemical descriptors. The generated cluster labels are stored in a string of digits in the same linear order as molecules. LS decoder (section 2.2. Label Swapper) transforms these digits back into the original labels. These digits are translated into characters using ASCII decoder, resulting in the decoded original message.

3. Results

3.1 Assessment of CryptoChem with five datasets

To demonstrate the feasibility and capabilities of CryptoChem, we validated the QSDR and the Molecular Encoder approach by encoding and decoding a series of five datasets consisting of numerical and textual information with increasing size and complexity (**Table 1**). Often considered as benchmark for storage and encoding techniques, these datasets have different sizes and complexity increasing from dataset 1 to 5.

Table 1. Description of the five datasets to be encoded in this project.

ID	Datasets to be stored using the Molecular Informatics technique	Size	Complexity
1	"123456789"	+	+
2	"abcdefghijklmnopqrstuvwxyz"	+	+
3	"Operation start at 11:00PM"	+	++
4	Latitude, Longitude, and Corresponding Time Zones for Major Cities. *Source: https://www.infoplease.com/world/world-geography/major-cities-latitude-longitude-and-corresponding-time-zones*	+++	++
5	Full text of the Declaration of Independence *Source: http://www.ushistory.org/declaration/document/*	+++++	+++++

MOLWRITE directly uses text files (*e.g.*, ".txt", ".csv") as input and stores the encoded molecules (SMILES) in CryptoChem messages. MOLREAD takes the generated CryptoChem messages as input and decodes the molecules into original textual message. All the five datasets were saved as plain texts in txt files.

For each of the five datasets, we used MOLWRITE to encode the original numerical and/or textual information with and without analogue retriever. The encoded messages were then decoded by MOLREAD. The results are summarized in **Table 2**.

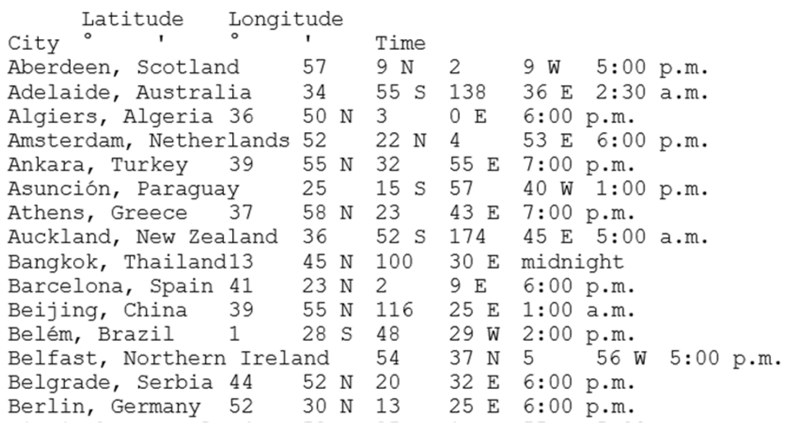
The first and the second datasets consists of Arabic numerals and English alphabet, respectively, which are considered as the basic elements of English texts. The results of the first and the second datasets showed that the MOLWRITE and MOLREAD programs can fast encode and decode the Arabic numerals and English alphabets with perfect accuracy. The result of the third dataset demonstrated the ability of MOLWRITE and MOLREAD programs to encode and decode simple sentence.

Table 2. Results of encoding and decoding the five datasets using MOLWRITE and MOLREAD programs (with programs performing in single CPU mode).

Dataset	**No. of characters** [a]	**No. of compounds**	**Accuracy of original molecules set**	**Encoding Time**	**Decoding Time**
1	10	10	100%	1s	1s
2	26	26	100%	1s	1s
3	26	26	100%	1s	1s
4	5,163	5,163	100%	144 s	10s
5	8,685	8,685	100%	247s	16s

[a] with spaces.

The real challenges are the 4th and 5th datasets. The 4th dataset (**Figure 4**) is the latitude, longitude, and corresponding time zones for major cities. It is worth noting that this dataset is stored in a tabular format in the txt file. Impressively, the output of MOLREAD of this dataset is the same as original text. This demonstrated that our programs can not only encode and decode the content of text, but also the format of the text. The 5th dataset is the entire article of the US Declaration of Independence. We tested our programs on the entire text including 8,685 characters and got the encoded text fully recovered. This validated our programs and demonstrated the feasibility of storing complex information with molecules.



```
     Latitude   Longitude
City  °     '     °     '     Time
Aberdeen, Scotland     57    9 N   2     9 W   5:00 p.m.
Adelaide, Australia    34    55 S  138   36 E  2:30 a.m.
Algiers, Algeria 36    50 N  3     0 E   6:00 p.m.
Amsterdam, Netherlands 52    22 N  4     53 E  6:00 p.m.
Ankara, Turkey   39    55 N  32    55 E  7:00 p.m.
Asunción, Paraguay     25    15 S  57    40 W  1:00 p.m.
Athens, Greece   37    58 N  23    43 E  7:00 p.m.
Auckland, New Zealand  36    52 S  174   45 E  5:00 a.m.
Bangkok, Thailand13    45 N  100   30 E  midnight
Barcelona, Spain 41    23 N  2     9 E   6:00 p.m.
Beijing, China   39    55 N  116   25 E  1:00 a.m.
Belém, Brazil    1     28 S  48    29 W  2:00 p.m.
Belfast, Northern Ireland    54    37 N  5     56 W  5:00 p.m.
Belgrade, Serbia 44    52 N  20    32 E  6:00 p.m.
Berlin, Germany  52    30 N  13    25 E  6:00 p.m.
```

(**a**)

```
CCCCOc1cccc(NCCCO)c1.O=C(NCCC[N@H+]1CCCC[C@@H]1CO)c2ccc(O)c(Cl)c2.C[C@@H](C(=O)N1CC
Cc2cc(OC(F)(F)F)ccc21)n3cccn3.CCc1nccn1Cc2cc(C(=O)O)ccc2OC.FC(F)(F)c1ccc(C[C@@H]2CCC[NH2+]C2)
cc1.Cc1ncnc2c1ncn2c3ccc([C@H](C)O)cc3.CCOC(=O)[C@H]1C(C)=NC(=O)N[C@@H]1c2ccccc2C.COc1cc([C
@H](C)NC(=O)[C@@H]2C[C@@H](O)C[NH2+]2)ccc1OC(C)C.Cc1nc(Cl)c2c(I)n[nH]c2n1.COc1cc([C@@H]2[
NH2+]CCc3cc(O)c(O)cc32)ccc1O.COc1ccc(Cl)cc1C(=O)N2CCN([C@@H]3CCC[C@H]3O)CC2.C[C@H]1C[N
@@H+](CCc2ccccc2)[C@H](C)CC1=O.Cc1nnc(CN(C)Cc2nc(c3ccc(F)cc3)oc2C)o1.C[C@H](c1ncc(C(C)(C)C)o1
)N2CCC[C@@H](CS(N)(=O)=O)C2.CC[C@H](C)[NH2+]Cc1cccc2[nH]ccc12.OCCCn1cnc(c2ccccc2)c1c3cccc4c
3OCO4.O=C\1NC(=O)N(c2ccccc2F)C(=O)/C1=C/c3ccc(O)cc3.CCN(C(=O)COc1cc(C)ccc1C(N)=O)C2=CCCCC2.
Fc1ccc(C2=Nc3nnnn3[C@H](c4ccc(F)cc4)C2)cc1.CCOCCC[NH2+]Cc1cc(Cl)c(OCC)c(OC)c1.Cc1cc(C)c(NC(=O
)COc2ccc(F)nc2)c(C)c1.CCn1cnnc1CNC(=O)[C@H](c2ccccc2)N3CCSCC3.C#C[C@@H]1C[C@@H]2CC(=O)[C
@@H]3[C@@H]4CCC(=O)[C@@]4(C)CC[C@@H]3[C@@]2(C)C[C@H]1O.Clc1cccc(S[C@H]2CC[NH2+]C2)
c1.Cc1ccc(C(=O)C[n+]2ccccc2C)c(C)c1.NC(=O)CN1CCN(C(=O)C=C2CCCCC2)CC1.NNCc1ccccc1OCc2ccccc2.
CCOc1cccc(/C=C/2\SC(=S)NC2=O)c1.CN1/C(=C\C=C\2/SC(=S)NC2=O)/C(C)(C)c3ccccc31.CCCOc1ccc(C[NH2
+]CCOCCO)cc1Br.Cc1cc(NC(=O)C[C@H](C)c2ccccc2)no1.Cc1cccc(OC[C@H](O)C[NH2+][C@H](C)c2cncc(F)
c2)c1.CCN1C(=O)/C(=C/c2ccc(OC(C)=O)cc2)/SC1=S.CC(C)N1C(=O)/C(=C/c2ccc(N(C)C)cc2)/SC1=S.COc1ccc(
Cl)c([C@@H]2CCC[NH2+]C2)c1.Cc1ccc(c2nnc(COC(=O)c3cnccn3)o2)cc1.C[C@@H](O)C[NH2+]CCNCc1cccc
c1OCc2ccccc2Cl.O=C(CSc1nncc2ccccc12)NCc3ccccc3F.CC[C@H]1CC[NH2+][C@@H]1Cc2ccc(F)cc2.Cc1cc(C
N2CC[NH+](C[C@@H](O)COc3ccccc3)CC2)on1.
```

(**b**)

Figure 4. (**a**) The 4th dataset, and (**b**) the encoded version using MOLWRITE.

4. Discussion

CryptoChem builds on the concept of Quantitative Structure-Data Relationship (QSDR) modeling method to encode and store information using machine learning, Big Chemical Data, molecular structures and their properties. There are several key advantages of using machine learning to create those relationships: *(i)* the descriptor ⇔ data relationship is encoded non-linearly, meaning there is no direct, obvious link between a particular chemical scaffold, shape, volume and the actual character to be stored; *(ii)* different molecules can still encode the same character, meaning that character frequency analysis (traditionally used to crack methods that substitute a given alphabet by another one) is useless with this technology, *(iii)* the relationships are quantified, meaning there is an actual weight (or set of weights) associating each molecular descriptor with a character. Therefore, a CryptoChem message cannot be read and/or written without using the right QSDR model (that includes the compounds used to train the model, the type of machine learning algorithms, the set of descriptors used to characterize these compounds, and the actual parameters of the models (*e.g.,* descriptor weights, number of neurons/hidden layers if using DNN). In this disclosed study, we utilized implicit and explicit 2D molecular descriptors/fingerprints and deep learning neural networks (DNN) model to encrypt and decipher CryptoChem messages containing textual and numerical data.

We have successfully developed our first QSDR-DNN models that directly link the molecular descriptors of chemicals in V1 dataset to numerical/textual characters. It should be noted that the application of our DNN models is unique from others found in literature. In contemporary research in cheminformatics, DNN models are usually used to build the quantitative relationship between molecular structures and their properties/activities (*e.g.,* binding affinity, toxicity) in the context of drug discovery or computational toxicity. For those models, the relevance and choice of descriptor sets are crucial in the model's performance to predict these desired outputs. DNN models must be trained with the most appropriate and relevant set molecular descriptors to correctly predict the targeted endpoint. In other words, the accuracy in those predictions relies heavily on choosing the right set of features, relevance between these features and the targeted characteristics, data curation, etc. If such important criteria are not met, it is usually challenging to obtain high accuracy. It is therefore important to follow normal protocols [22] in preprocessing data, training such QSDR models and evaluating their accuracy to ensure that the model's performance will be good enough to predict values on new data.

However, in this study, we train our models to associate manually created labels (labeled by k-means clustering algorithm) with the molecular descriptors of compounds. The major difference between contemporary approaches and ours is that we can manipulate the descriptor sets and cluster labels as we see fit, and our model will still learn to associate certain descriptors with cluster labels. If it fails to predict the right clusters, we can simply discard those data. In essence, our models learn to distinguish molecules based on specific molecular descriptors. Our goal is to use DNN models as keys to decode information, so standard protocols used in developing machine learning models are not strictly relevant here. For a given compound, trained DNN models must assign the right clusters for our large libraries of molecules given the right set of descriptors. So, we trained and fitted our models (envisioned as substitution key) with the entire dataset, instead of splitting it into train and test sets like in other contemporary cheminformatics researches. The wrongly predicted compounds which represent a very small percentage are discarded. This way, our fitted models have learned to associate molecules in our libraries with cluster labels with 100% accuracy.

MOLWRITE and MOLREAD programs were implemented to respectively write and read CryptoChem messages encoded with chemical molecules that could only be decoded using our QSDR DNN model as substitution key and specific descriptor sets. We showed in 3.1 Assessment of CryptoChem with five datasets section the true accomplishment of our system by encoding and decoding five datasets containing textual, numerical information and tabular formats with increasing complexity with 100% accuracy not only in content but also in format. Besides the fact that it is the first time ever one actually encodes the full text of the US Declaration of Independence with chemicals, this result validates this *proof-of-concept* project and demonstrate the feasibility of storing complex information with molecules and their properties.

Additionally, we implemented Analogue Retriever (AR) to replace molecules in CryptoChem messages generated from *reference set* in MOLWRITE with the new “analogues” from *analogue set*. By introducing new molecules from an unlabeled external set, it ensures a better integrity of data even if the initial reference database and CryptoChem messages are compromised. However, our well-trained QSAR models can decipher them correctly provided the right descriptor set to use and the correct sets of parameters. We tested AR by applying it on the five datasets we previously mentioned. The molecules from these CryptoChem messages were replaced with new molecules using AR, and we encoded them using MOLREAD program. The QSDR model used in MOLREAD was able to decipher the analogue retrieved CryptoChem messages with 100% accuracy. This accomplishment showed two important things: (1) our DNN models successfully learned the intricate features and representations mapping molecular descriptors with cluster labels, and (2) AR was designed with a good understanding of how our DNN model works.

In terms of implementing the AR, we tested several different metrics such as Tanimoto, Euclidean distance, string similarity, etc. We previously assumed that we could sample a few molecules from different clusters in the external set, and selected the cluster containing a molecule with the highest Tanimoto score or highest string match score or lowest Euclidean distance to the target molecule. These approaches were tested with different sample sizes, and clustering methods (DL-model based, initial k-means based), but all failed to retrieve the original messages. None of them yielded an accuracy score more than 51%. The analogues have not been chosen simply based on Tanimoto scores, Euclidean distance or string match. More complexity is involved, and selecting the analogues relies heavily on the initial k-means clustering. Unless the centroids from the initial k-means clustering generated with the exact random seed is known, the actual decoding could be near impossible. On the other side, it showed that our model may have learned to recognize these centroids and decipher the cluster labels on its own. By successfully implementing AR, we may have gained a better understanding on the inner workings of our QSDR model.

The summary of security layers in CryptoChem is shown in **Figure 5**. In order to encode the message, one can select a particular chemical space including a set of highly specific and similar chemical compounds or a combination of various chemical scopes. The chemical universe is vast and there are many possibilities of designing and enumerating one's *in-silico* libraries. For instance, cheminformatics software such as PKS Enumerator[23] or SIME[24] can automatically build extremely large libraries of macrocycles, macrolactones or macrolides with multiple constitutional and structural constraints.

Next, the choice of descriptor sets, or fingerprints is highly relevant in establishing the QSDR relationships between chemicals and cluster labels as well. Both the sender and receiver must know the type and exact descriptor set applied in order to successfully encode and decode CryptoChem messages. For even further improved security, one can specify in-house, hand-picked chemical descriptors or use 3D descriptors for which both the sender and the receiver have to know the method to specify the conformational arrangement used for each molecule to derive the correct 3D descriptors. Going one step further, one can even use 4D/MD descriptors[14] where chemicals are docked or run through molecular dynamic simulation in a specified binding site of a protein or ribosome target of interest for a specific duration. One can then extract 4D/MD chemical descriptors from that process and use that to establish to QSDR relationships. This is however highly complex and thus very challenging and rewarding at the same time, if the proper protocol and regulations are meticulously applied.

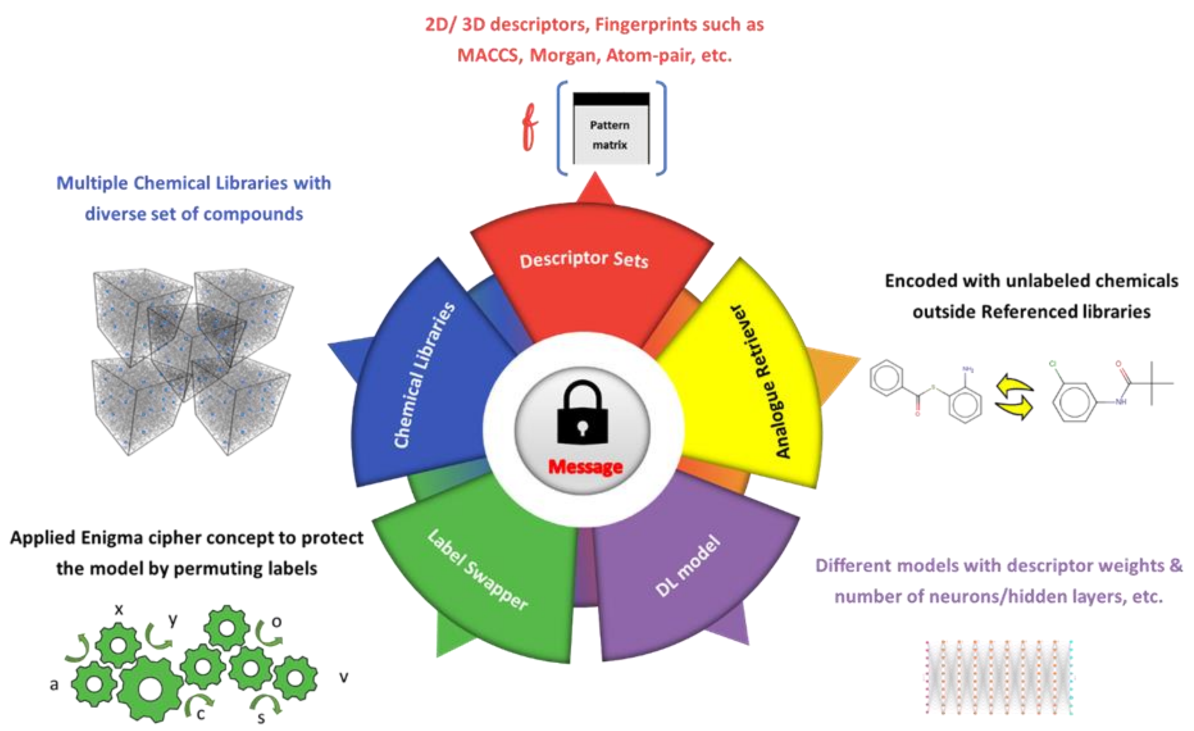

2D/ 3D descriptors, Fingerprints such as MACCS, Morgan, Atom-pair, etc.
f
Pattern matrix
Multiple Chemical Libraries with diverse set of compounds
Descriptor Sets
Chemical Libraries
Analogue Retriever
Encoded with unlabeled chemicals outside Referenced libraries
Message
Applied Enigma cipher concept to protect the model by permuting labels
Label Swapper
DL model
Different models with descriptor weights & number of neurons/hidden layers, etc.
x
y
o
a
v
c
s

Figure 5. Summary of security layers in CryptoChem

Another security layer is AR which obscures and adds confusion to the link between chemical structures and digits associated with them. Since new analogues without any associated cluster labels (*analogue set*) are being used in CryptoChem messages, it makes it difficult for third parties to decode the labels by attempting to identify the chemicals even with access to *reference set*. It is also highly improbable to reproduce the same QSDR model without using the exact parameters such as number of decision trees (in RF), nodes, layers, activation functions, dropout layers, regularizations, etc. (in DNN); thus it is treated as an additional security layer in CryptoChem technology. LS is another security layer to protect CryptoChem messages. It could protect them from cryptanalysis attacks purely based on frequency analysis by generating new labels based on permutation key (a molecular key) and 128 *neighbor molecules.*

We also compared CryptoChem to contemporary encryption methods such as block cipher (e.g. AES - Advanced Encryption Standard [25]) and stream cipher (e.g. One-Time Pad [26]). A summary of encryption features among these three encryption methods are provided in

Both CryptoChem and block cipher are based on substitution and permutation concepts whereas stream cipher is based on substitution alone. In terms of cryptographic keys, CryptoChem requires both substitution and permutation keys; the former can be any virtual chemical library of user's choice and the later can be any chemical molecule that needs to be passed in a different secure channel or in a physical form. With the application of label swapper, even if the permutation key (any chemical molecule of user's choice) and the substitution key (any large virtual chemical library) are compromised, the key space for permutation key is still rather large (factorial of 128 = 128!) due to the application of Label Swapper in CryptoChem. Meanwhile, block ciphers keys are bit dependent such as 56-, 128-, 256-bits etc. and consequently the key space depends on the bits employed as well (e.g. $2^{256} = 1.2 \times 10^{77}$). Stream cipher relies on pseudorandom number generator. Block cipher is deterministic with the use of same initialization vector; an encryption system is deterministic if the same cypher text is reproducible given the same plaintext and key. Stream cipher is deterministic as well. On the contrary, CryptoChem is not deterministic; meaning that different encoded messages are generated even if the same key molecule and models are being applied to generate them.

XOR operation [27] is not applicable to CryptoChem though it is used in both block and stream ciphers. Next, we compared avalanche effect, one important trait in determining the strength of a cryptographic algorithm wherein a higher avalanche effect is desired for a stronger encryption system [28]. An avalanche effect is achieved if a substantial change in the cipher message can be triggered by a slight change in the plaintext for a fixed key [29]. CryptoChem has a high level of avalanche effect since one character can be expressed by a cluster of chemicals, and the use of

Analogue Retriever and Label Swapper switched chemicals and change labels accordingly. Consequently, very different CryptoChem message are generated even when the same message is used as input. Block cipher possesses avalanche effect whereas stream cipher does not.

The operation unit of CryptoChem is byte (character) which is converted to molecules in the encrypted message. On the other hand, block cipher uses block of bits and stream cipher uses bytes. Both CryptoChem and block cipher use both principles of confusion and diffusion; which are central to the security of conventional encryption algorithms [30] while stream cipher uses confusion principle alone. Confusion is the concealment of the relation between the secret key and the cipher text whereas diffusion is the complexity of the relationship between the plain text and the cipher text [30]. Regarding the reversibility, both CryptoChem and block cipher made it highly improbable to reverse cipher text back to original text unlike stream cipher which applies XOR that can easily reverse cipher text to the plain text. CryptoChem and stream cipher are strongly resistant to error; meaning that an error in a character will not affect the rest. However, block cipher is more susceptible to error because an error in a byte will affect the entire block.

It is important to underscore that, despite the results of this comparison, we are <u>not</u> claiming that CryptoChem can be seen as a robust, fully secure, quantum-ready encryption technique in its current state. In order to fully assess those claims, a specific and detailed cryptanalysis study would need to be conducted (far beyond the scope of this proof-of-concept study).

Table 3. Comparison with Contemporary Encryption Methods

Features	CryptoChem	Block cipher	Stream cipher
Principle	Substitution-permutation	Substitution-permutation	Substitution
cryptographic key	Substitution key + permutation key	56, 128, 256-bits, etc.	Pseudorandom number generator
deterministic	No (if using the same key molecule and model)	Yes (if using same Initialization vector)	Yes
Key space	Substitution key: any chemical library Permutation key: 128! = 4 x 10^{215}	Depend on key length 256-bit: 2^{256} = 1.2 x 10^{77}	---
XOR based	No	Yes	Yes
avalanche effect	yes	yes	no
Operation unit	Character (byte) to molecule	Block of bits	byte
Confusion and diffusion	Uses both confusion and diffusion	Uses both confusion and diffusion	Relies on confusion only
Reversibility	Reversing encrypted text is hard.	Reversing encrypted text is hard.	It uses XOR for the encryption which can be easily reversed to the plain text.
Susceptibility to error	Strong (an error in a character will not affect the rest)	Weak (an error in a byte will affect the entire block)	Strong (an error in a byte will not affect the rest)

5. Conclusions

In this study, we conceived and implemented the CryptoChem method that uses chemical structures and their properties as the central element to encode and store information using chemicals. In this proof-of-concept version, we achieved the implementation of both MOLWRITE and MOLREAD programs to write and read CryptoChem messages. Additional levels of complexity afforded by Label Swapper and Analogue Retriever significantly strengthen the security of CryptoChem. The program validation on five datasets showed the accomplishment of our goal to encode and accurately decode data/information using molecules. Additionally, the comparison of CryptoChem to contemporary encryption methods could justify launching a detailed cryptanalysis of the method to fully understand its strong characteristics as well as its weak points. That full cryptanalysis and stress tests need to be conducted in the future to further assess and potentially demonstrate the security of CryptoChem, especially with the rise of quantum computers. Moreover, one could use CryptoChem as an additional layer of protection prior to encoding a message using AES or other modern cryptographic technology.

In further phases, we plan to develop different machine learning models and create multiple libraries with varying sizes based on different sets of descriptors (including 3D, 4D/MD) and fingerprints. To encode larger texts such, we certainly need much bigger chemical libraries. There is certainly more work to be done regarding the optimization of both AR and LS in terms of method improvement and GPU-accelerated parallel computing. GPU acceleration is the key to have CryptoChem ready for real-world and commercial applications. The software is freely-available for testing and available on our GitHub.

Software availability:

The current version of CryptoChem is available at https://github.com/XinhaoLi74/CryptoChem

Author Contributions

PPKZ and XL equally contributed to the development of CryptoChem. PPKZ designed Analogue Retriever, and XL designed Label Swapper. DF conceived and designed the study. All authors edited, revised the manuscript and have given approval to the final version of the manuscript.

Funding Sources

PPKZ and XL gratefully acknowledge DARPA/ARO for financial support (grant W911NF1810315). PPKZ held an “International Fellowship from AAUW”, and Olive Ruth

Russell Fellowship from Berea College. DF thanks the NC State Chancellor's Faculty Excellence Program for funding this project.

Acknowledgments

We thank DARPA/ARO for funding. The goal of the DARPA-funded component of this work was to explore additional dimensions of molecular storage that would be possible using molecular descriptors. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

(1) Goldman, N.; Bertone, P.; Chen, S.; Dessimoz, C.; LeProust, E. M.; Sipos, B.; Birney, E. Towards Practical, High-Capacity, Low-Maintenance Information Storage in Synthesized DNA. *Nature* **2013**, *494* (7435), 77–80. https://doi.org/10.1038/nature11875.

(2) Adleman, L. M.; Rothemund, P. W. K.; Roweis, S.; Winfree, E. On Applying Molecular Computation to the Data Encryption Standard. *J. Comput. Biol.* **1999**, *6* (1), 53–63. https://doi.org/10.1089/cmb.1999.6.53.

(3) Malhotra, M.; . G. DNA Cryptography: A Novel Approach for Data Security Using Flower Pollination Algorithm. *SSRN Electron. J.* **2019**. https://doi.org/10.2139/ssrn.3358159.

(4) Extance, A. How DNA Could Store All the World's Data. *Nature* **2016**, *537* (7618), 22–24. https://doi.org/10.1038/537022a.

(5) Fourches, D. Cheminformatics: At the Crossroad of Eras. **2014**, 539–546. https://doi.org/10.1007/978-94-017-9257-8_16.

(6) Ash, J.; Fourches, D. Characterizing the Chemical Space of ERK2 Kinase Inhibitors Using Descriptors Computed from Molecular Dynamics Trajectories. *J. Chem. Inf. Model.* **2017**, *57* (6). https://doi.org/10.1021/acs.jcim.7b00048.

(7) Leone, L.; Pezzella, A.; Crescenzi, O.; Napolitano, A.; Barone, V.; D'Ischia, M. Trichocyanines: A Red-Hair-Inspired Modular Platform for Dye-Based One-Time-Pad Molecular Cryptography. *ChemistryOpen* **2015**, *4* (3), 370–377. https://doi.org/10.1002/open.201402164.

(8) Arcadia, C. E.; Kennedy, E.; Geiser, J.; Dombroski, A.; Oakley, K.; Chen, S.-L.; Sprague, L.; Ozmen, M.; Sello, J.; Weber, P. M.; et al. Multicomponent Molecular Memory. *Nat. Commun.* **2020**, *11* (1), 691. https://doi.org/10.1038/s41467-020-14455-1.

(9) Chen, K.; Zhu, J.; Boskovic, F.; Keyser, U. F. Secure Data Storage on DNA Hard Drives. *bioRxiv* **2019**, 857748. https://doi.org/10.1101/857748.

(10) Zhu, H.; Tropsha, A.; Fourches, D.; Varnek, A.; Papa, E.; Gramatica, P.; Oberg, T.; Dao, P.; Cherkasov, A.; Tetko, I. V. Combinatorial QSAR Modeling of Chemical Toxicants Tested against Tetrahymena Pyriformis. *J. Chem. Inf. Model.* **2008**, *48* (4), 766–784. https://doi.org/10.1021/ci700443v.

(11) Alves, V. M.; Muratov, E.; Fourches, D.; Strickland, J.; Kleinstreuer, N.; Andrade, C. H.; Tropsha, A. Predicting Chemically-Induced Skin Reactions. Part II: QSAR Models of Skin Permeability and the Relationships between Skin Permeability and Skin Sensitization. *Toxicol. Appl. Pharmacol.* **2015**, *284* (2). https://doi.org/10.1016/j.taap.2014.12.013.

(12) Kuenemann, M. A.; Fourches, D. Cheminformatics Modeling of Amine Solutions for Assessing Their CO<inf>2</Inf>Absorption Properties. *Mol. Inform.* **2017**, *36* (7). https://doi.org/10.1002/minf.201600143.

(13) Cherkasov, A.; Muratov, E. N.; Fourches, D.; Varnek, A.; Baskin, I. I.; Cronin, M.; Dearden, J.; Gramatica, P.; Martin, Y. C.; Todeschini, R.; et al. QSAR Modeling: Where Have You Been? Where Are You Going To? *J. Med. Chem.* **2014**, *57* (12). https://doi.org/10.1021/jm4004285.

(14) Kyaw Zin, P. P.; Borrel, A.; Fourches, D. Benchmarking 2D/3D/MD-QSAR Models for Imatinib Derivatives: How Far Can We Predict? *J. Chem. Inf. Model.* **2020**. https://doi.org/10.1021/acs.jcim.0c00200.

(15) Mudawwar, M. F. Multicode: A Truly Multilingual Approach to Text Encoding. *Computer (Long. Beach. Calif).* **1997**, *30* (4), 37–43. https://doi.org/10.1109/2.585152.

(16) Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Model.* **1988**, *28* (1), 31–36. https://doi.org/10.1021/ci00057a005.

(17) Keegan, J. Intelligence in War, Knowledge of the Enemy from Napoleon to Al-Qaeda by Keegan, John: (2003) | Andrew Barnes Books / Military Melbourne.

(18) Sterling, T.; Irwin, J. J. ZINC 15 - Ligand Discovery for Everyone. *J. Chem. Inf. Model.*

2015, *55* (11), 2324–2337. https://doi.org/10.1021/acs.jcim.5b00559.

(19) Géron, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*; O'Reilly Media, 2017. https://doi.org/10.3389/fninf.2014.00014.

(20) Peris, Á. *NMT-Keras Documentation*; 2018.

(21) Chollet, F. *Deep Learning with Python*.

(22) Fourches, D.; Muratov, E.; Tropsha, A. Trust, but Verify: On the Importance of Chemical Structure Curation in Cheminformatics and QSAR Modeling Research. *J. Chem. Inf. Model.* **2011**, *50* (7), 1189–1204. https://doi.org/10.1021/ci100176x.Trust.

(23) Zin, P. P. K.; Williams, G.; Fourches, D. Cheminformatics-Based Enumeration and Analysis of Large Libraries of Macrolide Scaffolds. *J. Cheminform.* **2018**, *10* (1), 53. https://doi.org/10.1186/s13321-018-0307-6.

(24) Zin, P. P. K.; Williams, G.; Fourches, D. SIME: Synthetic Insight-Based Macrolide Enumerator to Generate the V1B Library of 1 Billion Macrolides. *J. Cheminform.* **2020**, *12* (1), 23. https://doi.org/10.1186/s13321-020-00427-6.

(25) Sanchez-Avila, C.; Sanchez-Reillo, R. The Rijndael Block Cipher (AES Proposal): A Comparison with DES. In *IEEE Annual International Carnahan Conference on Security Technology, Proceedings*; 2001; pp 229–234. https://doi.org/10.1109/ccst.2001.962837.

(26) Zeng, K.; Yang, C. H.; Wei, D. Y.; Rao, T. R. N. Pseudorandom Bit Generators in Stream-Cipher Cryptography. *Computer (Long. Beach. Calif).* **1991**, *24* (2), 8–17. https://doi.org/10.1109/2.67207.

(27) Huo, F.; Gong, G. XOR Encryption Versus Phase Encryption, an In-Depth Analysis. *IEEE Trans. Electromagn. Compat.* **2015**, *57* (4), 903–911. https://doi.org/10.1109/TEMC.2015.2390229.

(28) Karuppiah, M.; Ramanujam, S.; Professor, A. *Designing an Algorithm with High Avalanche Effect*; 2011; Vol. 11.

(29) Dawson, E.; Gustafson, H.; Pettitt, A. N. *Strict Key Avalanche Criterion*.

(30) Coskun, B.; Memon, N. Confusion/Diffusion Capabilities of Some Robust Hash Functions. In *2006 IEEE Conference on Information Sciences and Systems, CISS 2006 - Proceedings*; Institute of Electrical and Electronics Engineers Inc., 2006; pp 1188–1193. https://doi.org/10.1109/CISS.2006.286645.