ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022

CLOAKCRYPT: SECURING SECRETS USING FALSE BOTTOM ENCRYPTION

Dr. G. Murali [1], L. Bhavya[2]

[1]Assistant Professor and Head of the Department, Department of Computer Science and Engineering, JNTUA College of Engineering, Pulivendula

[2]Assistant Professor, Department of Computer Science and Engineering, JNTUA College of Engineering, Pulivendula

ABSTRACT

This paper presents a novel approach to deniable encryption using secret sharing techniques, distinct from honey encryption. Unlike honey encryption, which relies on a preprocessing step to reshape plaintext distribution, our method avoids computational assumptions and data preprocessing. This ensures deniability against attackers capable of forced decryption or brute-force attacks. Leveraging the concept of plausible deniability, multiple decryption keys can reveal different plaintexts from a single ciphertext. The symmetric scheme we propose is lightweight, efficient, and does not rely on computational intractability. This paper introduces False Bottom Encryption, a symmetric encryption scheme that combines aspects of honey encryption and deniable encryption without relying on computational intractability. Unlike honey encryption, which reshapes plaintext distribution through preprocessing, our approach achieves deniability by allowing multiple decryption keys to reveal different plaintexts from a single ciphertext. We provide numeric examples and Jupyter notebook implementations to validate the method's effectiveness.

INDEX TERMS: Deniable encryption, honey encryption, secret sharing

I. INTRODUCTION

In the context of secure communication, establishing safety often necessitates the exchange of information between senders and receivers. Encryption, combined with robust password protection, serves as a prevalent strategy for safeguarding data. Although these encryption techniques effectively thwart eavesdropping attempts, they may fall short in addressing coercion threats. In scenarios where attackers intercept ciphertext without access to the decryption key, both sender and receiver may be coerced into decrypting the message.

As fixes for this vulnerability, deniable encryption and non-committing encryption have been suggested. While the publicly known faking algorithm creates phony messages, the sender's encryption method encrypts messages using a secret key. On the other hand, it is computationally demanding to obtain the same ciphertext from two different techniques. In this research, we demonstrate a single algorithm capable of efficiently producing ciphertexts for both the genuine and fake messages, addressing this challenge while maintaining computational efficiency.

ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022

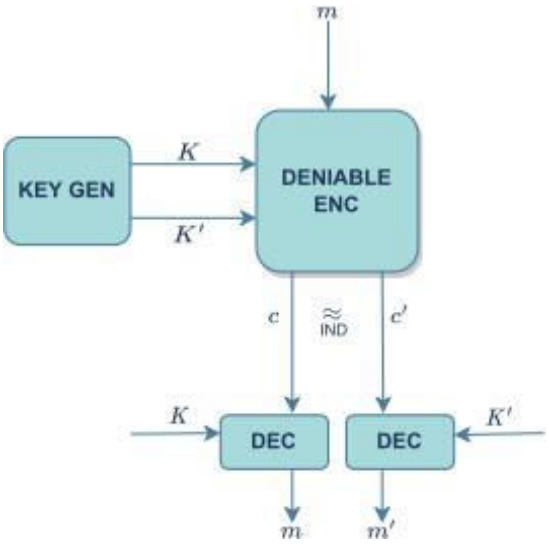
Let's say Alice wants to communicate with her friend John in secret so her mother won't find out. In this scenario, Alice can employ deniable encryption to conceal her communication. She gets ready two messages, one message which doesn't have meaning and a hidden message. Using an appropriate encryption algorithm, Alice encrypts John's message alongside Bob's message. She then encourages her pals to download the ciphertext and distributes it via a public channel. While only Bob and John possess the decryption keys to unlock the ciphertext, they each reveal different messages upon decryption: a fake message for Bob and a genuine message for John. This approach ensures that Alice can communicate securely with John while avoiding unwanted scrutiny from her mother. In the scenario where decryption is successful, both Bob and John can decipher and receive coherent messages sent by the sender, Alice. In response to her mother's questions, Alice can say that Bob was the intended recipient of the ciphertext and reveal the message Bob received. As Bob only knows the content of the message he received, he can serve as a reliable witness. Alice's mother is unable to identify which of Alice's acquaintances is the real receiver, even if she believes there is concealed information in the ciphertext. As a result, Alice does not need to step in to defend John because her mother cannot identify John as a suspect unless she has misgivings about all of Alice's acquaintances. This situation demonstrates how effective deniable encryption is at protecting private communications and preventing prying eyes.

II. RELATED WORK

According to [3], honey encryption is an innovative way to enhance standard encryption techniques by making sure that decryptions made with the incorrect decryption key seem reasonable. It can be accomplished after converting plaintext into an output that closely resembles a predefined distribution that is specified and corresponds to the distribution of the decrypted result obtained with an alternative key. Whichever key is used to decrypt the ciphertext, the ensuing plaintexts with real and fake messages show about the same allocation in either case which one is used. Since the distributions of the real and fake plaintexts are identical, this approach can be used to fool attackers who are trying to decipher the ciphertext without the right key. The capacity of the party encrypting data to determine the likelihood that an attacker will successfully identify a plaintext as authentic during the encryption process forms the basis of honey encryption security. In contrast to our technique and deniable encryption, honey encryption does not encrypt a second plaintext. Rather, it preserves one plaintext inside the ciphertext, similar to traditional encryption, but it tries to lessen the visibility of incorrect decryptions. This approach aims to maintain security even in the face of easily guessed keys or plaintexts with low min-entropy.

Canetti et al. [1] introduced deniable encryption, encompassing both deniable shared key and deniable public key schemes. This concept allows encryptors to assert alternative interpretations of encrypted messages. For instance, in the one-time pad, an encryptor can claim a different message was sent, supported by a different decryption key. Canetti et al. [2] expanded on this, presenting techniques for generating falsified messages with robust justificatio

ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022



They introduced translucent sets, which obscure membership without trapdoor information, enhancing deniability in encrypted communication. As per the sender-deniable encryption method proposed by Canetti et al., encrypting a bit b entails either emitting a string for b = 1 from the transparent set T, or a random string for b = 0. Because determining whether the string that is released is random or from T requires trapdoor information for membership verification in T, this technique makes sure that the plaintext bit is deniable even under duress. Thus, the sender can plausibly deny the authenticity of the message by claiming either possibility. Canetti et al. further extended this scheme to support receiver-deniability through an interactive approach, enabling the receiver to show evidence that the messages were forged.

The Canetti et al. method was improved by Klonowski et al. [6] to support messages of any depth by presenting alternative deniable encryption solutions that do not automatically recommend their use. The possibility that attackers could identify false communications in encryption methods that use the same algorithm for dishonest and honest encryptions was first noted in [4]. To solve this issue, the research presented two instances of public-key bit encryption techniques. In alignment with this, our work introduces a symmetric scheme that relies on entropy considerations for security, yet retains the use of a single algorithm for both real and fake messages. Our contribution represents a step towards providing formal evidence that security remains achievable even without the typical assumption that attackers can discern the right message out of a range of potential plaintexts. Non-committing encryption is closely related to deniable encryption but differs in its approach to generating fake ciphertexts and corresponding randomizers, making them indistinguishable from legitimate encryptions of true information. In non-committing encryption, the randomness (often represented by a key) used for generating fake ciphertexts is not under the user's control, as opposed to deniable encryption, which allows the user to produce appropriate randomness to decode the desired false message. Deniable encryption inherently encompasses non-committing encryption, but the reverse is not true, as highlighted in [1] with a relevant example.

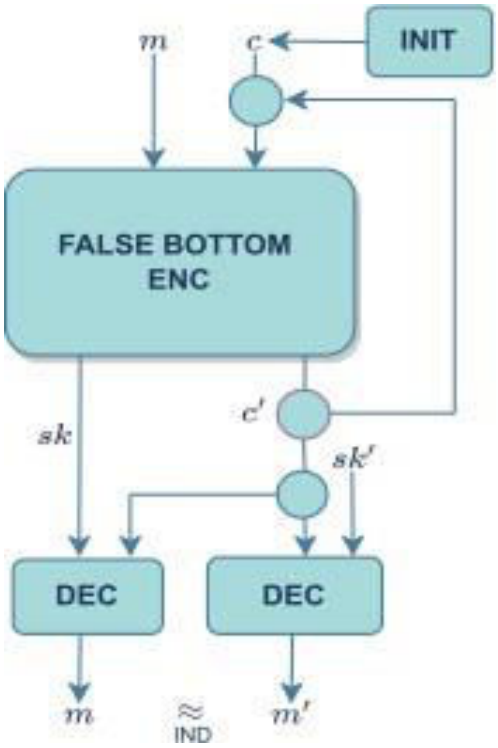
ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022

A. OUR CONTRIBUTION

This work introduces the following novel contribution:

1. It encrypts the plaintext which can be the combination of alphabetical, numerical, special characters and not limited to the encryption of numerical characters.

The False Bottom Encryption scheme, derived from secret sharing, is introduced as a symmetric and plausibly deniable encryption technique. Its name draws analogy from stage magicians' equipment featuring boxes with hidden compartments beneath false bottoms. Similar to this concept, the goal of False Bottom encryption is to make encrypted data sets—such ciphertextskept remotely, as in the cloud—appear empty or hidden, all the while keeping the user's secret key in their hands. If coercion is used, With modified copies of the secret key, the user can execute the decryption algorithm. As seen in Figure, this will result in the ciphertext decrypting into either the actual message or a string of fictitious plaintexts with different keys, decrypting the ciphertext using the original key yields the authentic message, while decryption using alternative keys generates different messages, thereby preserving deniability and enhancing security against coercion.



III. CONCEPTUAL IDEA

False-Bottom Encryption conceals confidential binary string secret within existing data blocks by representing it as a weighted sum in a finite field. This representation, akin to multivariate polynomial secret sharing, allows for efficient mapping of the secret by splitting it into smaller blocks. By fixing a set of constant *r*-values, termed the 'key-base', additional secrets can be represented similarly by reusing r- and α-values from the original representation. This approach minimizes the expansion of data while facilitating the representation of multiple secrets. The ciphertext comprises α-values, with all but the last being relevant for both the original and additional secrets. Only the last α-value is exclusively computed for each additional secret, ensuring concealment. This strategy allows for the symmetric deniable encryption of fake messages in instances of coercion, offering robust security measures. False-Bottom Encryption is a technique designed to conceal a secret within an array of existing data blocks. Assuming the secret is a binary string, It can be divided into smaller pieces and invertibly mapped to the constituents of a finite field F. Using a weighted sum to represent the secret, random values r1,...,rn and variables α1,...,αn−1 are selected, and the value that remains (αn) is solved for, allows multivariate polynomial secret sharing. By fixing a set of constant *r*-values, termed the 'key-base', additional secrets can be represented similarly by reusing *r*- and *α*-values from the original representation. Although representing the secret as a linear combination may seem inefficient, as it expands the secret into multiple values, it facilitates the representation of multiple secrets using the same set of values. For instance, another secret (*m′*) can be represented similarly to the original secret (*m*) by reusing the *r*- and *α*-values, albeit in a different order and number. This approach ensures that both original and additional secrets remain concealed within the data blocks, providing a robust method of encryption.

In the False-Bottom Encryption scheme, reusing the *r*- and *α*-values from the original secret representation allows for the efficient inclusion of additional secrets. This reutilization ensures that only a single new element, αn′, needs to be computed afresh for each new secret, minimizing data expansion. The ciphertext comprises the set of α-values, including all values relevant for both the original secret (*m*) and the additional secret (*m′*), with only $\alpha_{n'}$ being calculated only for m′. As such, Unaware of the r- and α-values being utilized for each message, both the original and additional secrets remain hidden. 'Secret decryption key' encompasses information regarding the necessary *r*-values, *α*-values, and their multiplication and summation order to recover the intended message. Additional messages may incorporate using the similar approach, incrementally extending the information. In cases of coercion, any one of fraudulent messages (m′) can be revealed rather than the actual message (m), facilitating symmetric deniable encryption. This process is further elucidated to ensure robust implementation.

IV. METHODOLOGY

Lowercase characters like t and k stand for scalars in the discussion that follows, whereas c and r are bold which vectors. Random variables and sets are indicated by uppercase letters in a regular

font. Notably, X ⊂R M denotes the choice of a subset within M in which each member From M, x ∈ X is uniformly and independently distributed. Similarly, x ∈R M represents an x- draw from the finite set M that is uniformly random. When M is converted to a vector, a selection of coordinates expressed as a sequence is shown. On the other hand, dim(c) shows how many coordinates or dimensions a vector c has,

|M| indicates the size or cardinality of a finite collection. Using array notation, which is widely used in programming languages, i-th member in the c vector is represented by c[i]. F represents an arbitrary but fixed finite field and is the symbol

for the subgroup of multiplicative units, F∗ = F

\{0}. The random variable X with the distribution F is represented by the notation X ∼ F, and the Shannon entropy of the random source X is indicated by H(X).

SECRET SHARING BASICS

This research builds upon the concept of secret sharing, a cryptographic technique where a secret value m is distributed among multiple players as shares, ensuring that no individual player can deduce m solely from their share. Collaboration among a designated subset of players is necessary to reconstruct the original secret. Shamir's scheme, for instance, employs polynomial interpolation to distribute shares. In order to satisfy p(0) = m, a degree n random polynomial having coefficients randomly must be generated. The shares of each participant are then allotted in accordance with the values of the polynomial at particular points, such as p(1), p(2),..., p(n + 1). If n + 1 or more values are combined, the polynomial p(x) can be uniquely determined; otherwise, it remains uncertain if less than n players combine their shares. In some situations, the interpolation issue collapses to an underdetermined system of equations, which prevents an attacker from knowing enough to figure out the secret. Similar ideas underlie Blakley's secret sharing, except each participant is given a single linear equation in n variables, which essentially describes an n-dimensional hyperplane. Both Shamir's and Blakley's secret sharing schemes rely on creating underdetermined systems of equations to protect the secrecy of the shared value. In these schemes, if fewer than the required number of players collaborate, the resulting system of equations remains underdetermined, preventing the reconstruction of the secret. This protection mechanism hinges on ensuring that there are not enough equations to uniquely determine the secret unless a specific threshold of information is collectively available. Similarly, our deniable encryption scheme will leverage this concept by utilizing the "degrees of freedom" inherent in underdetermined systems of equations as secret decryption keys, ensuring that the secret remains concealed unless the requisite conditions for decryption are met.

A. ENCRYPTION

The three algorithms that make up an encryption system are Dec, Enc, and KeyGen. The Bit Lengths of blocks and keys are frequently determined by the KeyGen algorithm, a probabilistic function that initializes structures and algorithms depending on security factors. The encryption and decryption keys are provided by the pair (sk, sk') that it outputs. As symmetric schemes are

the main focus, one can assume that sk = sk', which reduces the nomenclature to simply "secret key," in contrast to public-key cryptography. With a message m and a secret key sk, the Enc algorithm—which might alternatively be probabilistic—produces a ciphertext c. The Dec algorithm, which is deterministic in nature, uses the secret key sk to decode the ciphertext c and guarantees that there is a one in the likelihood of recovering the original message m.

V. IMPLEMENTATION

Our cipher's size of block is determined by a natural number parameter called t, and block count required the ciphertext's initialization is indicated by another parameter called n. Our hidden key represented by sk for m message can have a uniform size determined by a third parameter, $k \in N$. For differentiation between symbols of message and to link the appropriate keys, we'll employ an index. The numbers k, n, and t are selected independently of each other, although they are all of individually appropriate sizes for efficiency; that is, t will start out big, but n and k may start out tiny.

A. INITIALIZATION

We design a field F = Zp[X]/(f (X)), where p is denoted by prime and f is a non-reducible polynomial having degree d which belongs to N, selected so $\lceil \log_2(p) \rceil \cdot d \geq t$, given the security parameters t and n. Additionally, an element can function as a singular block. As a result, an element from F can encode the entire message m. This ordered collection will never change and will be used as a starting point for drawing fresh plaintexts into an expanding set of ciphertexts.

B. STRING ENCODING

Let's denote the message string as m with a length of n, and let A be the array to store the ASCII values. We convert the message into ASCII values as a function *f* where $f(M,A)=\{A1,A2,A3,\ldots,An\}$ where m is the message as a string, A is the array storing the ASCII values and Ai represents the i^{th} element in the array A containing the ASCII value of i^{th} character in the string S.

This step, String encoding helps in encrypting plain text i.e., messages of string datatype not limiting the encryption to numerical data. Each ascii value is encrypted as we go through further steps in encryption process. It generates different ciphertext for each different character in the message which can be stored in the form of matrix.

C. INCLUDING MESSAGES TO THE CURRENT CIPHERTEXT

Since we expect to add several messages (m1, m2, m3,...) to the ciphertext $c = (\alpha 1,\ldots,\alpha \dim(c))$ in the future, we will now allow each message to have an index (i), that will create a one-to-one mapping with the next set of decryption keys.

Thus, let us assume that we want to call Enc with a list which doesn't have any elements consisting of secret keys in order to include the message to

c. This starts the following processes: After randomly choosing a number ni∈{2,...,k}, we extract the ni−1 indices (ji,1,...,ji,ni−1)∈{1,...,dim(c)}, where αi,j are set to c[j] for any j that falls inside ji,1,...,ji,ni−1. After that, the modified ciphertext is returned as c′ → c‖α, where ‖ denotes that the newly computed value is simply appended to the list of values in c.

The information about which indices in the vector (list) c′ and which elements of the key base r were utilized to represent mi is all that is contained in the secret key returned to recover mi from c′. The secret key shares are returned as a list

$sk_{new}:=((j_{i,1},\rho_{i,1}),...,(j_{i,n_i-1},\rho_{i,n_i-1}),(dim(\mathbf{c})+1,\rho_{i,n_i}))$.

Since we are simply appending the value to c, the last index in the ciphertext is always the ciphertext length +1. But take notice that this present last index won't be the last element when more messages are added to c, possibly before a forced decryption occurs.

Because we only store indexes instead of the actual message values, the key size stays relatively small. It grows much slower than the message itself, making it manageable in practical applications: (1) involves a maximum of k terms, with every ρ element requiring O(1) bits. Since adding more plaintexts would expand the index's range, the resulting key size will be O(k · log dim c) = O(log dim(c)) bits.

D. DECRYPTING *i*-TH MESSAGE *mi*

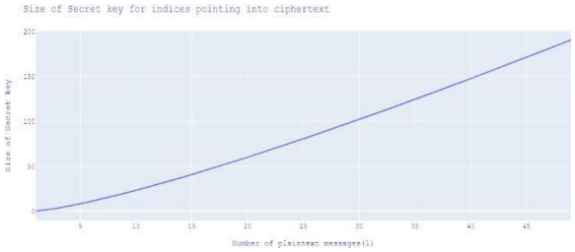
The "current" ciphertext, denoted by c, could have come from a prior call to Enc (see above). We retrieve the corresponding indices from secret keys and use (1) to recover mi in order to decrypt the desired message. While its correctness is insignificant, the security merits more investigation. But first, let's also talk about the options for editing and removing messages from C.

VI. RESULTS AND SECURITY ANALYSIS

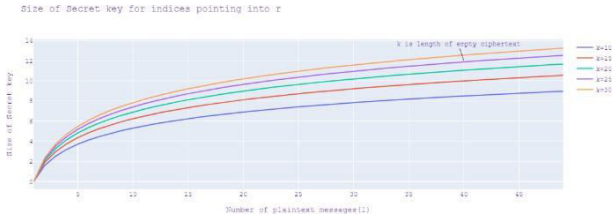
From here on, we will just employ asymptotic Landau symbols for ease in order to avoid dealing with redundant information or deciphertexts or keys. The length of the empty ciphertext is k by construction and increases by 1 element for each subsequent piece of plaintext. The bound remains constant even if messages are added, removed, or edited from c. This length is not altered—it might even go down. Assuming a block size in which each message have n bits, we calculate the relation between size of ciohertext and plaintext. The ciphertext, when represented with vector having the length of O(ℓ) in a field F, can have O(n·ℓ) bits of length, while the overall length of ℓ plaintexts is ℓ · n bits. We can represent the additional data needed for encryption with the symbol η. Compared to the total size of the original messages, this overhead is relatively small. This is because η is a constant value, and constants don't affect how the size grows as the number of messages increases. In other words, the overhead is independent of the number of messages and remains manageable (denoted by O(1) in big O notation), Thus, the plan is asymptotically efficient The secret key in False Bottom Encryption comes in two parts, each related to referencing hidden messages. The first

ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022

part, for referencing messages within a large space (denoted by c), grows slowly with the number of messages (ℓ) due to the use of indices. This growth is similar to how a list of items only needs a short code to reference each item, even as the list grows. The second part, for referencing messages within a smaller space (denoted by k), also grows slowly with the number



of messages but at a slightly different rate. Overall, the key size remains manageable even for a large number of messages.. As k is a constant, the space complexity becomes O(ℓ·log ℓ) which



encrypts all of the secret keys.

A. SECURITY ANALYSIS

In analyzing security across different scenarios, three distinct cases are considered. The first scenario involves a Forced Decryption Attack where the attacker possesses the ciphertext and can compel the legitimate user to decrypt it using a decryption key. The security model here focuses on resisting decryption under coercion. The second scenario encompasses an Offline Attack with Infinite Computational Power, assuming the attacker has unlimited computational resources to perform exhaustive be evaluated comprehensively, enabling tailored defenses against varying threat environments.

1) OFFLINE BRUTE-FORCE ATTACKS

In this case, coercion is impossible and tries to decrypt a message (mi) using a brute-force trial without Alice's help. As in earlier assessments, we normalize the attack by evaluating the attacker's residual uncertainty conditioned on all the data gathered from thorough investigation. This statement, known as Proposition 1, indicates the amount of computation needed for an exhaustive key search by stating that the brute- force complexity of decrypting every message from ciphertext (c) is given by a certain formula searches or advanced cryptanalysis.

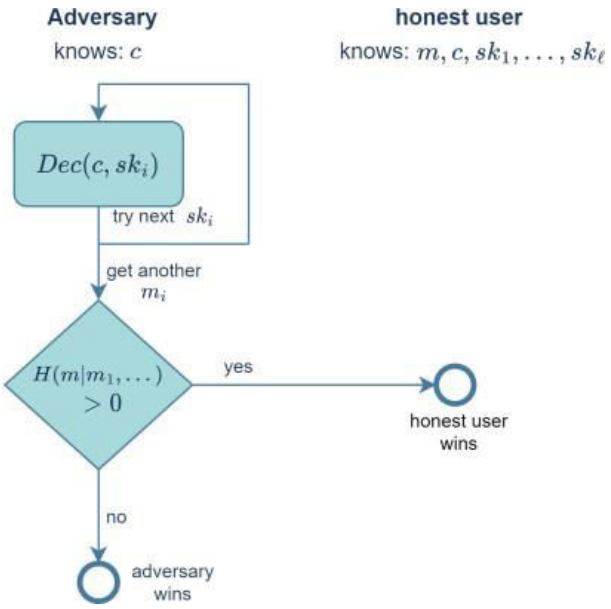
The

$$k \cdot p^d + \sum \binom{k}{} \cdot dim(\boldsymbol{c})! \in \Theta(k^{\dim(\boldsymbol{c})}) \quad (4)$$

ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022

encryption scheme in this case should withstand brute force attacks. The third scenario extends the n_i=2 n_i

ISSN PRINT 2319 1775 Online 2320 7876
 UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, 2022



$\frac{}{(\dim(c)-ni)!}$

offline attack by incorporating Background Knowledge possessed by the attacker, enhancing their capabilities beyond computational power alone. Security in this scenario demands resilience against attacks leveraging both computational power and additional knowledge. By delineating specific attacker models and corresponding security definitions for each scenario, the encryption scheme's robustness can The attacker comprehends that the message (mi) is created from a random selection of ni items from c, weighted with a corresponding selection of random values from a set F. N!/(N-ni)! is the amount of options available from c for creating a combination of ni elements out of N, both with and without replacement. Assuming the attacker first guesses the entire r for efficiency, they are

k

then left with $(_{n_i})$ choices for the indices from r

for m_i. Simple computations show that the overall count stays inside k.dim(c), where k (and all other variables) stay constant and only the dimension, or length, dim(c), changes over time

Number of Messages(k)	Brute force Complexity
1	O(128!)
5	$O(1.175494 \times 10^{34} \times 128!)$
10	$O(1 \times 10^{40} \times 128!)$
15	$O(1.54 \times 10^{45} \times 128!)$
20	$O(8.51 \times 10^{47} \times 128!)$

VII. CONCLUSION

We have demonstrated a conceptually straightforward technique for hiding data inside an already-existing string sequence, enabling deceptive decryption in the event that the decryption keys are forcibly revealed. For instance, is easy to envision in this context: let's say that we always use a pseudorandom number generator that is seeded with the password of our choice whenever we need to choose random values. In that respect, the security implications should be carefully studied, since the entire entropy regarding the secret reduces to the Shannon entropy, which is the password choosing process's min-entropy, which indicates how challenging it is to guess the password. This can be further extended to several communication processes which involves sharing of confidential information.

REFERENCES

[1] R. Canetti, U. Feige, O. Goldreich, and M. Naor, ‘‘Adaptively secure multi-party computation,’’ in Proc. 28th Annu. ACM Symp. Theory Com put., Philadelphia, PA, USA, 1996, pp. 639–648. [Online]. Available: http://portal.acm.org/citation.cfm?doid=237814.23 8015

[2] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, ‘‘Deniable encryption,’’ in Proc. 17th Annu. Int. Cryptol. Conf. (Lecture Notes in Computer Sci ence), vol. 1294. Santa Barbara, CA, USA: Springer, 1997, pp. 90–104.

[3] A. Juels and T. Ristenpart, ‘‘Honey encryption: Security beyond the brute-force bound,’’ in Advances in Cryptology—EUROCRYPT (Lec ture Notes in Computer Science), vol. 8441, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, P. Q. Nguyen, and E. Oswald, Eds. Berlin, Germany: Springer, 2014, pp. 293–310, doi: 10.1007/978-3-642-55220-5_17.

[4] M. Durmuth and D. M. Freeman, ‘‘Deniable encryption with negli gible detection probability: An interactive construction,’’ in Advances in Cryptology—EUROCRYPT (Lecture Notes in Computer Science), vol. 6632, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F.