Poster: Statistical coding scheme for the protection of cryptographic systems against brute-force attack

Hyun-Ju Jo (Student) and Ji Won Yoon (Faculty)
Gradudate School of Information Security, Korea University, South Korea
Email: {hyunju870, jiwon_yoon}@korea.ac.kr

***Abstract*—A new algorithm for secure communication, with statistical encoders and decoders is introduced to protect cryptographic algorithms from brute-force attacks. In our proposed approach, even incorrect plain texts decrypted with wrong keys can have semantically/synthetically fine meaning. That is, every keys can be used as a key in cryptanalysis. Because of this, malicious people cannot distinguish correct plain texts from many incorrect plain texts.**

I. Introduction

A brute-force attack is the simplest but still effective attack for the cryptographic algorithms. An underlying assumption of the brute-force attack is that the complete keyspace was used to generate keys so longer key size is believed to be able to provide higher security level. Barker and Roginsky's work shows the recommendation for transitioning the use of cryptographic algorithms and key lengths [1] against modern threats including brute-force attacks.

However, there are still some concerns in security although the length of the key is increased to obtain such higher security level because of two reasons. First, there are some reports that a number of cryptographic systems have unfortunately been cracked although they were originally thought to be impossible to crack by the brute-force attack since the keys used in the cryptographic algorithms are obtained from the pseudo random number generator. The practical keyspace to search through was found to be much smaller than theoretical one, because there exist a lack of entropy in the pseudo random number generators. For instance, Goldberg and Wagner discovered the predictable Netscape seed in Secure Socket Layer (SSL) encryption protocol with the poor pseudo random number generators [2]. Similar flaws from such lack of entropy have been discovered in several cryptographic algorithms: Debian OpenSSL[3], RSA public key factoring[4] and The Elliptic Curve Digital Signature Algorithm (ECDSA) in bitcoin [5]. Second, custom hardware attacks with graphic processing unit (GPU) and field-programmable gate array (FPGA) have proven their capability in the brute-force attack for some ciphers.

From this point of view, it may not help to increase the security level by just enlarging the key size. Of course, we can address this problem by using cryptographically secure PRNG. However, in this paper, we propose a new solution to implicitly resolve the problem by adopting a statistical coding scheme to the cryptographic algorithms.

II. Backgroud: Probabilistic Language Model

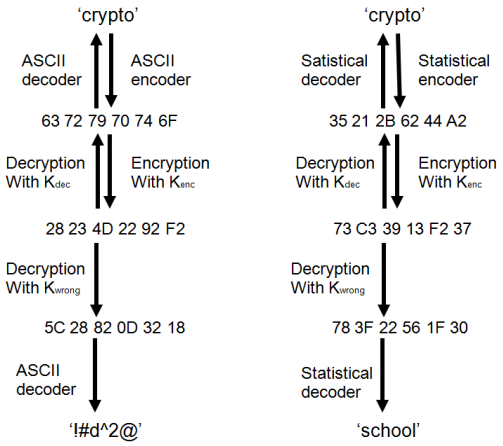
In this study, we develop a statistical code based on probabilistic language model, which is replacing a traditional ASCII code. The probabilistic language model assigns a probability to a sequence of m words/strings $\mathbf{S}$, where each symbol in the string belongs to an alphabet of words or characters. For this model, we can define a random process s with a sequence of random variables $x_0, x_1, \cdots, x_m$ that have values in a countable set $\mathbf{A}$, called the state space. Each s_i is the i-th discrete random variable which has one of N possible values where $N = card(\mathbf{A})$. The language model can be explained in discrete-time process. The full joint posterior given the k-order Markov process s with the Markov condition is $p(s_{1:m}) = \prod_{i=1}^{m} p(s_i|s_{1:i-1}) = \prod_{i=1}^{m} p(s_i|s_{i-k:i-1})$ since $p(s_i|s_{1:i-1}) = p(s_i|s_{i-k:i-1})$. In general, the language models use n-gram statistics, frequency tables of all previous sets of n consecutive words. An n-gram model is interpreted as a $(n-1)$th order Markov chain.

III. Proposed Approach

Let C, $P_{correct}$, K_{enc} and K_{dec} be the cipher text, correct plain texts, and its corresponding correct keys for decryption and encryption where $K_{dec} = K_{enc}$ for symmetric cryptosystems. In common cryptosystems like AES, DES and RSA, ones obtain syntactically meaningless plain texts P_{wrong} if the incorrect decryption key K_{wrong} is used where $K_{wrong} \neq K_{dec}$. However, if P_{wrong} looks like a real plain text both semantically and syntactically, then the security level increases since brute-force attack cannot distinguish P_{wrong} from $P_{correct}$. Thus our main idea is to make P_{wrong} to look like a semantically and syntactically meaningful plain texts with a statistical coding scheme rather than fixed ones like ASCII code. The comparison of the schemes with a simple example for both traditional approach and our proposed approach are displayed in figure 1. As shown in this figure, our proposed approach will generate semantically meaningful plain texts (P_{wrong}='$school$') although K_{wrong} is used for decryption while the decrypted plain texts (P_{wrong}='!#d^2@') are meaningless in the traditional approach. Therefore, although malicous person performs to find plain texts by brute-force, he or she cannot find actual plain texts $P_{correct}$ since $P_{correct}$ cannot be distinguished from P_{wrong} or he/she cannot determine whether $P_{correct}$ is correct.

A. Statistical encoding and decoding schemes

Let $p(X_1 = w)$ be the frequency of the first letter of sentences where $w \in \{a, b, , z\}$. Of course, w can have several different forms (basically it can be any formats with information.): 1) numbers like '1', 2) special characters like '?' and 3) words like '$paper$'. For example, if we want to encrypt the plain text 'the school' in our proposed scheme and



(a) Traditional approach (b) Proposed approach

Fig. 1. Comparison of traditional cryptographic systems and proposed approach with statistical coder

each character is encrypted into L bits, we find the location of the first character 't' in the cumulated mass function, $Pr(t) = p(X_1 = a) + \cdots + p(X_1 = t)$. Afterwards, we find a set of corresponding L bits binary codes to decimal number $\lfloor Pr(t) \times 2^L \rfloor$. This is an encoding process for the first character and the reverse operation is the decoding process since they are symmetric.

Let's look at the other characters. The r-th character can also be encoded in a similar way. For instance, we have the third character $(r = 3)$ 'e' of the example and it can be encoded by considering a conditional mass function $p(X_3 = e|X_2 = h, X_1 = t)$. The decoding algorithm for the r-th character is also symmetric so we can do it by using reverse operation. The decoding and decryption procedures for the actual correct key are exactly identical to those for any wrong keys. If a given cipher, C, is decrypted with a wrong key, K_{wrong}, for $K_{wrong} \neq K_{dec}$, we will obtain completely meaningless binary codes. However, the binary codes are decoded into the plain text, P_{wrong}, by using statistical decoder and then it can be a meaningful wrong and different message. Our proposed approach stores the information about probabilistic Markovian language model that represents the above cumulative mass functions. The stored information is open in public so therefore all people can access this anytime.

The next step is to construct the cumulative mass function (CMF) via a training step with any materials or documents. For instance, if we infer the language model and construct CMF with 'Bible', then we can build the CMF for the characters of the Bible. If the scientific novel is used for the training, then information about scientific novel can be used for the CMF. This provides a surprisingly important benefit to us. The wrong messages are automatically constructed and generated in a style of Bible or Scientific novel.

IV. Simulation Results

For the simulation, we have built five different cumulative mass functions from five different sources obtained from http://textfiles.com: 1) NASA documents, 2) Romeo & Juliet Novel, 3) Public Key Cryptography document, 4) Bible and 5) Classic music documents. After building the cumulative mass functions we have encrypted a message 'deniable encryption' with K_{enc} via our proposed approach. Afterwards, we decrypt the ciphers with a correct key K_{dec} and two randomly chosen different keys $K^{(1)}_{wrong}$ and $K^{(2)}_{wrong}$. We confirmed that the cipher is obviously decrypted to the actual message $P_{correct}$ when K_{dec} is used. Table I shows the decrypted messages when incorrect keys are used. Surprisingly, we could also obtain semantically meaningful incorrect messages with $K^{(1)}_{wrong}$ and $K^{(2)}_{wrong}$. Therefore, we can obtain a lot of semantically meaningful incorrect plain texts with almost all keys through whole key space so that malicious people cannot find the underlying actual plain texts $P_{correct}$ by using brute-force attack. In addition, note that the decrypted messages are highly related to the contents of the trained documents(sources) to construct the cumulative mass functions.

TABLE I. Decrypted texts with different keys and databases: NASA (16KB), Romeo & Juliet (247KB), Cryptography (340KB), Bible (4.9MB)

Dataset for DB	$K^{(1)}_{wrong} \neq K_{dec}$	$K^{(2)}_{wrong} \neq K_{dec} \neq K^{(1)}_{wrong}$
NASA	'the scout's payload ac'	'the spacecraft and the'
Romeo & Juliet	'what show the project'	'scene iii. scene iii'
Cryptography	'the secret key signatu'	'the probabilistic tech'
Bible	'and the children of th'	'and the lord was not b'

V. Conclusion

A new statistical encoding and decoding system is introduced in this paper to protect any cryptographic systems from brute-force attack. The statistical model based on probabilistic language model builds cumulative mass function (CMF). Given statistical coding scheme with CMF, plain texts decrypted with incorrect keys can be decoded to be semantically or synthetically meaningful. Therefore, malicious people cannot crack the ciphers by using brute-force attack even in a cryptosystem with lower security level since decoded incorrect plain texts cannot be distinguished from actual plain texts.

Acknowledgment

This research was mainly supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2013R1A1A1012797).

References

[1] E. B. Barker and A. L. Roginsky, "Sp 800-131a. transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," Gaithersburg, MD, United States, Tech. Rep., 2011.

[2] I. Goldberg and D. Wagner, "Randomness and the netscape browser," *Dr. Dobb's Journal*, vol. 21, no. 66, 1996.

[3] Debian Security Advisory, "DSA-1571-1 openssl – predictable random number generator," Availlable on http://www.debian.org/security/2008/dsa-1571, 2008.

[4] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter, "Ron was wrong, Whit is right." *IACR Cryptology ePrint Archive*, vol. 2012, p. 64, 2012, informal publication.

[5] The Register, "Android bug batters Bitcoin wallets," Availlable on http://www.theregister.co.uk/2013/08/12/android_bug_batters_bitcoin_wallets/, August 2013.