Robert S. Andrews  
Vice President, Security  
OCRA Creative Recursive Acronyms, Inc.

Mr. Andrews,

We have reviewed your specifications and have devised a solution to your problem that we feel meets in every regard your guidelines. We understand that security is a real concern, but also realize that the algorithm must be convenient and fast.

There are two ends to the problem at hand. One end we have complete security and on the other end we have ease-of-use. In the quest for unbreakable security, ease-of-use is often lost. However, in focusing on ease-of-use, many security concessions must be made. In this problem, we had to design an algorithm that was both easy to use and reasonably secure.

We also understood that this algorithm had to be fast on devices with limited computing power. While we have not extensively tested on mobile devices, the operations involved in encrypting and decrypting are basic computations that should be available on many if not most mobile processors. Another concern is memory. We have created a one-to-one encryption algorithm. The encrypted text has the same number of letters that the original text had. During the entire encryption process, the message is never longer than the original plaintext. The largest block of information we need store for a 140 character message is a 280 digit integer.

The algorithm relies on a numeric key, which in this case, we have limited to 10 digits. Since our encryption algorithm is similar to a substitution cipher, the output is mapped to the lower-case English alphabet and therefore easily reproduced on any mobile input device. Without the key, there is little likelihood that any intercepted messages can be deciphered in a reasonable about of time. Both encryption and decryption perform at the same speed and the time spent is proportional to the length the message.

The algorithm that we have devised is not computationally intensive and could easily be implemented on portable devices. It is safe to distribute the code to our algorithm publicly because the strength of the algorithm is in the key, not the method of encryption. We will now explain the encryption and decryption algorithm.

Let \( \text{str}(i) \) be the base ten representation of \( i \in \mathbb{Z} \). Let \( \text{len}(s) \) be the length of this representation. Let \( \text{int}(s) \) be the inverse, returning the integer. Let \( f : \text{lower case letters} \rightarrow \mathbb{Z}_{26} \) be the bijective map described on page 12 of the textbook, with \( f^{-1} \) its inverse. Consider a message \( M \), a list of lower case letters, that we wish to encrypt. Let the length of \( M \) be \( n \), and let \( M_i \) be the \( i \)th letter in the message. We first select a key \( k \) as a ten-digit integer that contains no zeros. From this key we generate a pad \( p \) of pseudorandom, two digit integers of sufficient length to cover the message. We do so by squaring the key value \( k \) until it is at least twice the length of the message. We then tokenize the resulting number \( p \) into \( n \) two digit encrypters \( e_i, 0 \leq i < n \). We let \( e_i \) be the two digit integer whose value corresponds to the \( 2i \)th and \( (2i + 1) \)th digits of \( \text{str}(p) \). For example, if the pad started 1234..., the first two encrypters would be 12 and 34. If there are more than \( 2n \) digits in \( p \), the remainder are ignored. We then construct a cipher \( C \) of lower case letters, equal in length to \( M \). We say

\[
C_i = f^{-1}(f(M_i) + e_i)
\]

We note that the function \( f^{-1} \) maps congruence classes to lower case letters, so using the chart on page 12...
of the textbook would require one to calculate the value \( (f(M_i) + e_i) \mod 26 \). To decrypt the message, we calculate the encrypters \( e_i \) in the same manner as above, using \( n = \text{len}(C) \). We calculate the original message as

\[
M_i = f^{-1}(f(C_i) - e_i)
\]

It is left as an exercise for the reader to show how this formula follows from the first. We here present the algorithm in pseudocode. Given \( M, k \) we do

\[
\begin{align*}
n & \leftarrow \text{len}(\text{str}(M)) \\
p & \leftarrow k \\
\text{while} \ \text{len}(\text{str}(p)) < 2n \ \text{do} \\
& \quad p \leftarrow p^2 \\
\text{end while} \\
\text{for} \ 0 \leq i < n \ \text{do} \\
& \quad e_i \leftarrow \text{int}(\text{str}(p)_{2i}\text{str}(p)_{2i+1}) \\
& \quad C_i \leftarrow f^{-1}(f(M_i) + e_i) \\
\text{end for} \\
\text{return} \ C
\end{align*}
\]

Only minor changes are made to decrypt rather than encrypt, as described above.

In this paper, we have addressed OCRAI’s problem of encrypting nontrivial amounts of data using a short key. As evidenced by our patented encryption cipher, we have solved this problem by using key expansion and modular arithmetic in conjunction with numerical assignments of standard alpha characters. Our key expansion method, using the properties of squared numbers, allows the original, 10-digit key to cover an entire message with relatively fast computational speed. This key expansion method is also known to derive a unique, expanded key for any given amount of plaintext, with the essential property of being unambiguous. We also are able to obfuscate and encode an entire message of any length by simply adding this key to the numeric representation of the message.

The inverse cipher is likewise able to decrypt the ciphertext with ease and without collision by simple knowledge of the 10 digit key, while also proving the inverse cipher to be strong against attack from anyone who does not possess the key. We feel that the security of this algorithm is not based upon knowledge of the algorithm itself, but upon the secrecy of the key, allowing this method to stand strong with Kerckhoffs principle. Our method will keep OCRAI’s information secure, while allowing the ciphertext and the plaintext to be typed on a standard keyboard or mobile device.

We therefore humbly submit our encryption method as a standard which OCRAI will be able to adopt with ease, a standard that keeps information security a top priority.

Securely,