Maintaining security while sending sensitive company information through text messages is a major concern. An encryption system must be simple enough to create ciphertext in a form that can be typed on a standard computer or mobile device, but it must also be complicated enough that the confidential information being sent will be kept safe. Our proposal seeks for security through obscurity. Each character of the plaintext will be hidden in a group of four other randomly chosen characters to create the ciphertext. The encryption algorithm tells the user which position of these five characters is the important one—the one that holds a character of the plaintext. All other positions within the block of five will be filled randomly.

**Here is the encryption algorithm:**

*Step one:* The following characters are supported in this algorithm: letters, numbers, and spaces. We must remove all other characters from the plaintext before encryption.

*Step two:* We choose a 10-digit non-square number. This will be the encryption key.

*Step three:* We take the square root of this number in a program that allows us to see it to arbitrary precision. (The number of digits after the decimal point must be equivalent to the number of characters in the plaintext.) Each digit after the decimal point determines the position in each successive block of five where the plaintext character is located. The first position is indexed 0, the second position is indexed 1, and so on. The digit from the square root is taken modulo 5 (i.e. the remainder when divided by 5) to find the index.

For example, using \( \sqrt{234567890} \approx 35136.418286 \ldots \), the digits after the decimal point are 4, 1, 8, 2, 8, 6, … and so the chosen positions have indices 4, 1, 3 (because 8 \( \equiv 3 \) (mod 5)), 2, 3, 1, …

**NOTE:** The modulo 5 is important because it allows us to break the plaintext into 5 character blocks, with each position in the block being represented by only one number.

*Step four:* We place the plaintext characters in the appropriate position designated by step three.

For example, if the current character of the plaintext is X and the chosen position has index 2, then that block would be _ _ X _ _ at this point (recall the first position has index 0, and so the third position has index 2), where “_” denotes a random character.
Step five: We fill in all other positions with random letters, numbers, and spaces.

Step six: We send the ciphertext.

Here is the decryption algorithm:
Step one: We locate the encryption key.

Step two: We take the square root of this number in a program that will allow us to see it to arbitrary precision. The number of digits following the decimal point of our square root must be equivalent to the number of characters in the plaintext. As every plaintext character is represented by 5 ciphertext characters, this will be the length of the ciphertext divided by five, so we should be able to determine this easily.

Step three: Each digit modulo 5 following the decimal point represents the index location of the plaintext character within the five character block (starting with the first five). See explanation in the encryption algorithm, step three. We then simply mark the important characters in order to see the plaintext.

Explanation
Although the encryption method may seem simple, the system will be effective due to the complexity of the key. Assuming that any adversary knows that our method of encryption involves hiding the plaintext characters within blocks of random characters, it is important to have a large number of possible keys. The number chosen for the key is a 10-digit number, meaning there are nearly 10^10 possible keys (excluding perfect squares). Because the square root of the number used for the key will be irrational, the digits behind the decimal point won’t repeat within the 140 characters of a text message (or ever, for that matter…). The chance of an outsider determining the key being used is quite low, so the information being sent will be kept safe.