1 Introduction

In any business setting, a secure method of communication is required in order for employees within the organization to collaborate and complete a project. The method must be simple enough to implement so that message is still feasible to share without alteration or corruption, while remaining secure from those for whom it is not intended. It is also vital to be able to identify, with a high confidence, where a message comes from and where it goes. The method that we present below does just that; it uses simple mathematics with large digits and prime numbers to encrypt a message in a way that only the sender and recipient are able to read the message and know that it did in fact originate from the sender and was received by the recipient. Furthermore, we outline how OCRAI's in-house engineers will be able to implement this method within the app, thus avoiding any potential human error in encryption and decryption.

2 Encryption Method

We begin by reading in the message as it is. The encryption key is a 10 digit number, where all 10 of the digits are in base 5, and the first digit is not a 0. This means there are $4(5^9)$ possible encryption keys. Furthermore, The maximum encryption key is 4,444,444,444 and our minimum encryption key is 1,000,000,000. Each employee will be assigned a unique encryption key, randomly generated from the above constraints.

We manipulate the data by simultaneously iterating over the string and the encryption key. For each character the message, a number of random ascci letters (upper and lower case letters) are added after the character, where the number of characters corresponds to the current digit in the encryption key. For example if the text message is "Alice wants to send a message to Bob" and the encryption key is 3221420140, after one iteration, the message would look something like:

AdFIlice wants to send a message to Bob

We added three random characters after "A", namely "dFI", since we are at the first character in the message (A) and the first digit in the encryption key (3). After 2 iterations, the message would look something like

AdFIleaice wants to send a message to Bob

At the 11th iteration, we would be at the character s in the word wants. Since our encryption key is only 10 digits long, we would loop back to the beginning of the key, meaning 3 letters will be added after the s. In general, if we are at the ith character in the string, we are at the i%10th digit in the encryption key.

Using the same text message and encryption key as above, the message would look something like

AdFIleaiDHcpenggW CbwaxnSqfftsGkQ eVtGXok FUSNsMRenqdIOLp aWkf
xpmyXeMsgGersxxagueXHWY tbKfoIT JIBvouKgDbsi

Note: Spaces count as a character in our data manipulation.

The engineers need to design a way for the app to decrypt the messages if they are sent to an employee. Along with a unique encryption key, each employee also has a unique prime number. In order to calculate the range of prime numbers needed, we need the equation

$$\pi(x) \approx \frac{x}{ln(x)}$$

where $\pi(x)$ is the number of primes less than x. This means we need to find an x such that

$\pi(x)$ is greater than the number of employees. For example, if you have 1,000 employees, $x = 10,000$ is a sufficient number since

$$\pi(10,000) \approx \frac{10,000}{ln(10,000)} \approx 1,085$$

So choosing primes less than 10,000 will guarantee that each employee has a unique prime.

The decryption key is fairly simple: it is the sender's encryption key multiplied by a by the sender's prime, with the sender's prime and recipient's prime appended on the end. For example, using 3221420140 as the encryption key, 53 as the sender's prime, and 13 as the recipient's prime, the decryption key is 1707352674205313 since $322142014 * 53 = 170735267420$, the sender's prime number is 53 and the recipient's prime is 13. This decryption key provides a way for the app to verify the identity of both the sender and the receiver since each prime is unique to an employee.

To decrpyt the message, the sender's prime and the recipients prime will be removed from the decrpytion key, and then what is remaining will be divided by the sender's prime. For example, our decryption key is 1707352674205313. We remove 53 and 13 to get 170735267420 and then divide 170735267420 by 53, which gives us 322142014, precisely the sender's unique encryption key. Now we use the sender's encryption key to decrypt the message by essentially working backwards, iterating over the encrypted message and the encryption key. The encrypted message is

AdFIleaiDHcpenggW CbwaxnSqfftsGkQ eVtGXok FUSNsMRenqdIOLp aWkf
xpmyXeMsgGersxxagueXHWY tbKfoIT JIBvouKgDbsi

At the first iteration, the character we are looking at is A and the digit in the encryption key is 3. Thus, we remove the 3 characters after A to get

AleaiDHcpenggW CbwaxnSqfftsGkQ eVtGXok FUSNsMRenqdIOLp aWkf
xpmyXeMsgGersxxagueXHWY tbKfoIT JIBvouKgDbsi

We continue this process until we reach the end of the message, looping through the encryption key like we did with encrypting the message.

A careful analysis shows that our method of encryption will not produce a message over 800 characters. Suppose we have the maximum number of characters allowed in the text message (160 characters) and the maximum encryption key (4,444,444,444). This means that 160 of the characters will have 4 characters added after them, so the total characters after encryption is 800.

3 Conclusion

As we can see, this method preserves the message, including punctuation, symbols, and the case of the letter, while keeping the encrypted message within the specified parameters. This method also allows a unique prime "identity" to be assigned to each employee, thus allowing sender and recipient verification, as well as a secure way for the recipient to be able to read the message. We believe that once implemented, this app will allow OCRAI's employees to communicate easily and securely, and avoid any future mishaps with leaking secure company data.