# CS 305 Module Five Checksum Verification Assignment

**Instructions:** Replace the bracketed text with your answers in your own words.

## Algorithm Cipher

Recommend an appropriate encryption algorithm cipher that avoids collisions.

I recommend the Rivest-Shamir-Adleman (RSA) encryption algorithm cipher to avoid collisions for the purposes of the business (which I will refer to as The Business) I am working with.

## Justification

Justify your reasoning for the recommended algorithm cipher by providing a brief, high-level overview of the encryption algorithm cipher.

Since The Business needs to have a public key, which will be distributed and shared with clients on a website, and a private key, it is necessary to use an asymmetric encryption algorithm to meet their needs. RSA is an asymmetric encryption algorithm, utilizing both keys for its encryption process. The public key is used for encryption, while the private key is used for decryption. The private key is only held by the individual(s) decrypting the data and, ideally, does not sacrifice security as it is scaled. Since it is more complex than symmetrical encryption, it is more demanding in-terms of resource requirements. As a result, it is typically used for smaller transactions (like check-sums).

RSA encryption works by utilizing prime number factorization to create numbers that are easy to compute going one-way, but difficult to reverse. Creating the key is exponentially faster than reversing the key. Once data has been encrypted by the public key, it can only be decrypted using the private key. Currently, the largest key size that has been “cracked” is 768 bits long. In theory, the amount of work it would take to crack a key of that size is immense, but still doable. The National Institute of Standards and Technology recommends a minimum key size of 2048-bits. Cracking a key of this size is technically possible, but doing so would require immense resources- essentially bad-actors on a nation state scale (Lake, 2021). As such, it is unlikely this would be a real risk to The Business.

Avoiding collisions means avoiding a scenario where two input strings produce the same hash result. Although no hash function is completely safe from this happening (there are technically infinite input possibilities to generate an output), the larger the output size of a function, the safer it is from brute force attacks. The chances of a collision are very low, but never zero. Avoiding collisions is extremely important because collisions allow bad-actors to exploit vulnerabilities. For example, an attacker could fool a system into downloading a malicious file with the same hash as the valid, safe file. The file would appear valid, and the system downloading it would have no way of identifying it was malicious based on the hash (Privacy Canada, 2021).

## Generate Checksum

Refactor the code to encrypt a text string and generate a checksum verification. You will submit your refactored code for your instructor to review in addition to this document.

## Verification

Demonstrate that a hash value has been created for the unique text string (your first and last name) by executing the Java code. Then use your web browser to connect to the RESTful API server. This should show your first and last name as the unique data string in the browser, the name of the algorithm cipher you used, and the checksum hash value. Capture a screenshot of the web browser with your unique information and insert it below.

Although I think I created the appropriate code to generate a hash value, I can’t seem to get my program to run correctly. I’ve attached the error from my console to a word document and submitted it as well!

References

*Hash Collision Attack*. Privacy Canada. (2021). https://privacycanada.net/hash-functions/hash-collision-attack/.

Lake, J. (2021, March 18). *What is RSA encryption and how does it work?* Comparitech. https://www.comparitech.com/blog/information-security/rsa-encryption/.