Password Hashing

October 4, 2024

1 Password Hashing: Security and Strength

Owen Kroeger

Prof. Sluiter

ITT-305

1.1 Why Hashing is a Better Choice for Password Security than Encryption

Hashing is often preferred over encryption for password storage due to its one-way nature. While encryption can be reversed (decrypted) with the correct key, hashing is a one-way function. Once a password is hashed, it cannot be directly reverted to its original form. This makes it more secure for password storage because even if the hashed passwords are stolen, the actual passwords cannot easily be retrieved. Additionally, modern password storage methods use **salting**, where random data is added to the password before hashing, making it much harder for attackers to use precomputed tables (rainbow tables) to reverse-engineer passwords.

1.2 Other Than Passwords, How is Hashing Used to Improve Data Security?

Beyond password security, hashing is widely used in various data security contexts:

- **Data Integrity**: Hashing is used to verify that data has not been altered during transmission or storage. By comparing the hash of a file before and after transmission, we can detect whether the data has been tampered with.
- **Digital Signatures**: Hashes are used in digital signatures to make sure that a document has not been changed since it was signed. The hash of the document is encrypted with the sender's private key, and any modification to the document would result in a different hash.
- **Blockchain**: Hashing is fundamental in blockchain technology. Each block in the blockchain contains a hash of the previous block, creating a secure chain. Altering one block would break the chain due to hash mismatch, making it evident that tampering has occurred.

1.3 What Insights Did You Learn About Password Strength?

An insight I gained through this was that using a long string of words as a password is still quite insecure. I always wondered why the default (generated) passwords for things were complicated strings of numbers, letters, and characters, but now I understand how dictionary attacks can be easily used to crack word passwords. I also now understand why increasing the size of the library used in a brute force attack (via using symbols) greatly increases the security of the password.

1.4 Password Cracking Time Estimate Table

Below is the password cracking time estimate table based on the hashing speed of 5,000,000 guesses per second (based on my CPU's average run):

1.4.1 Password Cracking Time Estimates (Based on 5,000,000 guesses/second)

Password Length	Alphabet Size	Total Combinations	Time to Crack
5	26	26^5 = 11881376	2.38 seconds
6	26	26^6 = 308915776	1.03 minutes
7	26	26^7 = 8031810176	26.77 minutes
8	26	26^8 = 208827064576	11.60 hours
9	26	26^9 = 5429503678976	12.57 days
10	26	26^10 = 141167095653376	326.78 days
11	26	26^11 = 3670344486987776	23.27 years
12	26	26^12 = 95428956661682176	605.21 years
8	52	52^8 = 53459728531456	123.75 days
8	72	72^8 = 722204136308736	4.58 years

1.4.2 Findings and Best Practices for Creating Passwords

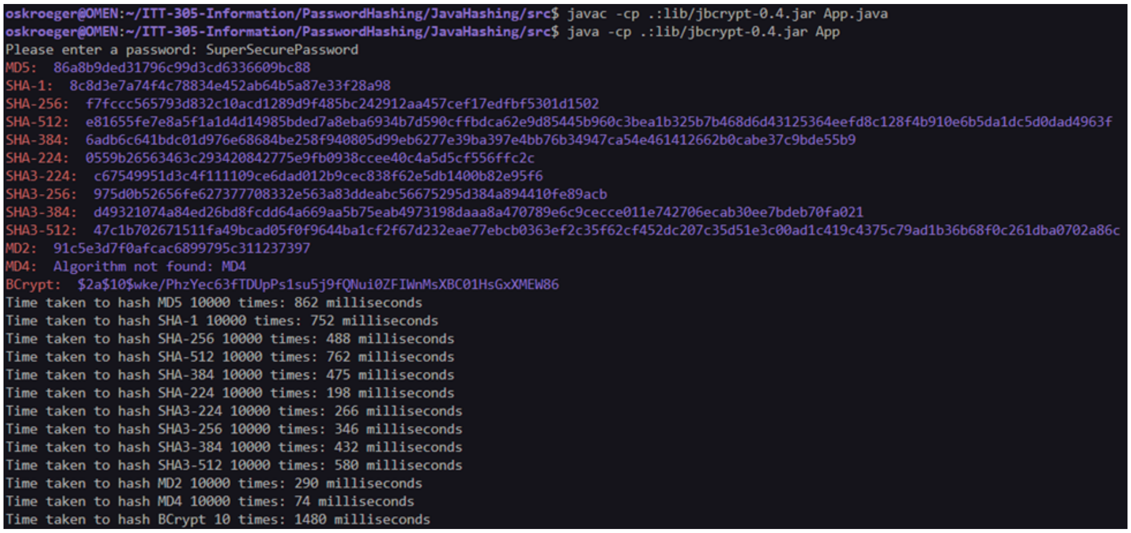
From the password cracking time estimate table, we can draw several conclusions:

1. **Longer Passwords are More Secure**: Increasing the password length significantly increases the number of combinations an attacker must guess. For example, a password of 8 characters using the 26-letter alphabet takes over 11 hours to crack, while a 12-character password takes over 600 years.

2. **Increase Alphabet Size**: Using more characters, such as adding numbers, symbols, and uppercase/lowercase letters, further increases password strength. In my example, an 8-character password using 52 characters (letters plus upper and lowercase) takes just under 124 days to crack, while using 72 characters extends that to over 4.5 years.

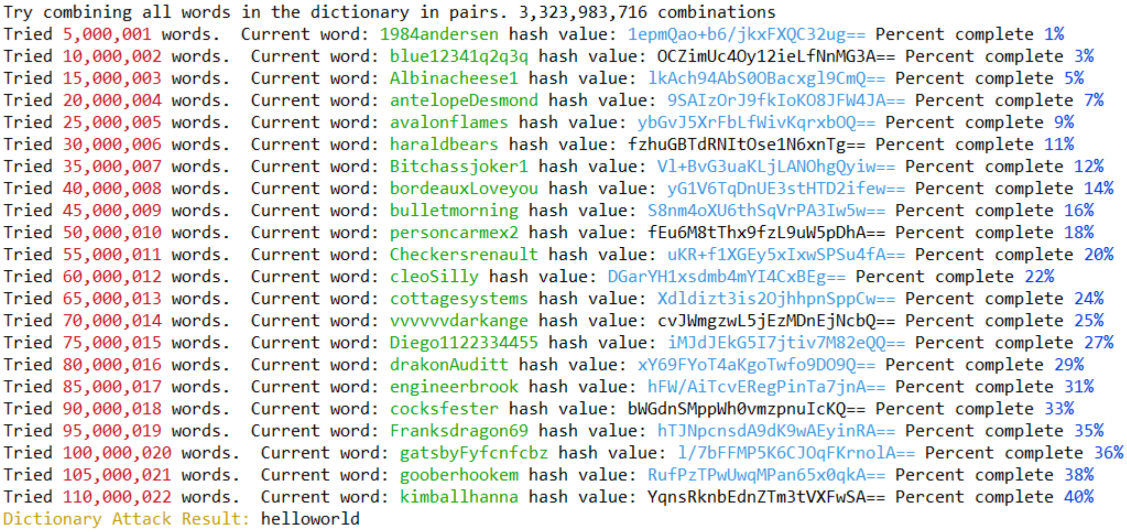
3. **Best Practices**: Based on these findings, it is clear that we should:

   - Use passwords of at least 12 characters.
   - Include a mix of letters, numbers, and symbols.
   - Avoid easily guessable passwords such as common words or phrases.
   - Consider using a password manager to generate and store complex, unique passwords for each site or service.



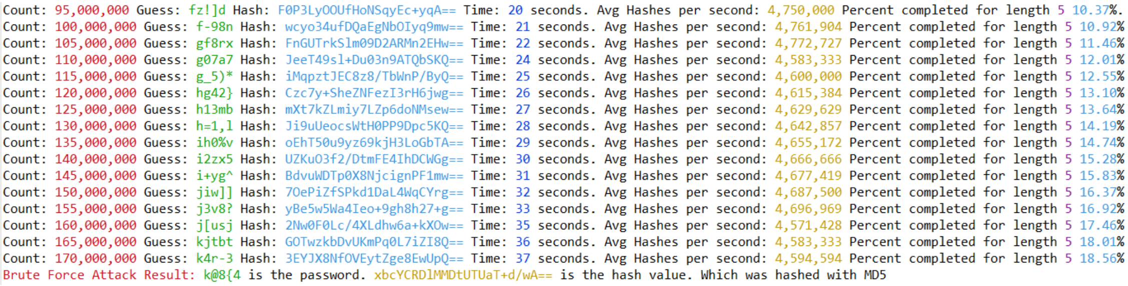
```
oskroeger@OMEN:~/ITT-305-Information/PasswordHashing/JavaHashing/src$ javac -cp .:lib/jbcrypt-0.4.jar App.java
oskroeger@OMEN:~/ITT-305-Information/PasswordHashing/JavaHashing/src$ java -cp .:lib/jbcrypt-0.4.jar App
Please enter a password: SuperSecurePassword
MD5:  86a8b9ded31796c99d3cd6336609bc88
SHA-1:  8c8d3e7a74f4c78834e452ab64b5a87e33f28a98
SHA-256:  f7fccc565793d832c10acd1289d9f485bc242912aa457cef17edfbf5301d1502
SHA-512:  e81655fe7e8a5f1a1d4d14985bded7a8eba6934b7d590cffbdca62e9d85445b960c3bea1b325b7b468d6d43125364eefd8c128f4b910e6b5da1dc5d0dad4963f
SHA-384:  6adb6c641bdc01d976e68684be258f940805d99eb6277e39ba397e4bb76b34947ca54e461412662b0cabe37c9bde55b9
SHA-224:  0559b26563463c293420842775e9fb0938ccee40c4a5d5cf556ffc2c
SHA3-224:  c67549951d3c4f111109ce6dad012b9cec838f62e5db1400b82e95f6
SHA3-256:  975d0b52656fe627377708332e563a83ddeabc56675295d384a894410fe89acb
SHA3-384:  d49321074a84ed26bd8fcdd64a669aa5b75eab4973198daaa8a470789e6c9cecce011e742706ecab30ee7bdeb70fa021
SHA3-512:  47c1b702671511fa49bcad05f0f9644ba1cf2f67d232eae77ebcb0363ef2c35f62cf452dc207c35d51e3c00ad1c419c4375c79ad1b36b68f0c261dba0702a86c
MD2:  91c5e3d7f0afcac6899795c311237397
MD4:  Algorithm not found: MD4
BCrypt:  $2a$10$wke/PhzYec63fTDUpPs1su5j9fQNui0ZFIWnMsXBC01HsGxXMEW86
Time taken to hash MD5 10000 times: 862 milliseconds
Time taken to hash SHA-1 10000 times: 752 milliseconds
Time taken to hash SHA-256 10000 times: 488 milliseconds
Time taken to hash SHA-512 10000 times: 762 milliseconds
Time taken to hash SHA-384 10000 times: 475 milliseconds
Time taken to hash SHA-224 10000 times: 198 milliseconds
Time taken to hash SHA3-224 10000 times: 266 milliseconds
Time taken to hash SHA3-256 10000 times: 346 milliseconds
Time taken to hash SHA3-384 10000 times: 432 milliseconds
Time taken to hash SHA3-512 10000 times: 580 milliseconds
Time taken to hash MD2 10000 times: 290 milliseconds
Time taken to hash MD4 10000 times: 74 milliseconds
Time taken to hash BCrypt 10 times: 1480 milliseconds
```

Caption: ↑ This picture shows the result of Part 2 where we ran the same password through different hashing algorithms 10000 times each and compared the run times against each other.



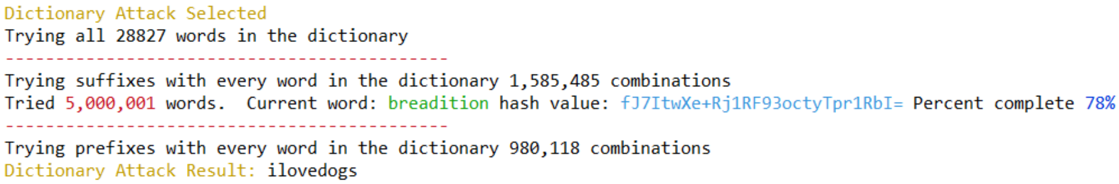
```
Try combining all words in the dictionary in pairs. 3,323,983,716 combinations
Tried 5,000,001 words.  Current word: 1984andersen hash value: 1epmQao+b6/jkxFXQC32ug== Percent complete 1%
Tried 10,000,002 words.  Current word: blue12341q2q3q hash value: OCZimUc4Oy12ieLfNnMG3A== Percent complete 3%
Tried 15,000,003 words.  Current word: Albinacheese1 hash value: lkAch94AbS0OBacxgl9CmQ== Percent complete 5%
Tried 20,000,004 words.  Current word: antelopeDesmond hash value: 9SAIzOrJ9fkIoKO8JFW4JA== Percent complete 7%
Tried 25,000,005 words.  Current word: avalonflames hash value: ybGvJ5XrFbLfWivKqrxbOQ== Percent complete 9%
Tried 30,000,006 words.  Current word: haraldbears hash value: fzhuGBTdRNItOse1N6xnTg== Percent complete 11%
Tried 35,000,007 words.  Current word: Bitchassjoker1 hash value: Vl+BvG3uaKLjLANOhgQyiw== Percent complete 12%
Tried 40,000,008 words.  Current word: bordeauxLoveyou hash value: yG1V6TqDnUE3stHTD2ifew== Percent complete 14%
Tried 45,000,009 words.  Current word: bulletmorning hash value: S8nm4oXU6thSqVrPA3Iw5w== Percent complete 16%
Tried 50,000,010 words.  Current word: personcarmex2 hash value: fEu6M8tThx9fzL9uW5pDhA== Percent complete 18%
Tried 55,000,011 words.  Current word: Checkersrenault hash value: uKR+f1XGEy5xIxwSPSu4fA== Percent complete 20%
Tried 60,000,012 words.  Current word: cleoSilly hash value: DGarYH1xsdmb4mYI4CxBEg== Percent complete 22%
Tried 65,000,013 words.  Current word: cottagesystems hash value: Xdldizt3is2OjhhpnSppCw== Percent complete 24%
Tried 70,000,014 words.  Current word: vvvvvvdarkange hash value: cvJWmgzwL5jEzMDnEjNcbQ== Percent complete 25%
Tried 75,000,015 words.  Current word: Diego1122334455 hash value: iMJdJEkG5I7jtiv7M82eQQ== Percent complete 27%
Tried 80,000,016 words.  Current word: drakonAuditt hash value: xY69FYoT4aKgoTwfo9DO9Q== Percent complete 29%
Tried 85,000,017 words.  Current word: engineerbrook hash value: hFW/AiTcvERegPinTa7jnA== Percent complete 31%
Tried 90,000,018 words.  Current word: cocksfester hash value: bWGdnSMppWh0vmzpnuIcKQ== Percent complete 33%
Tried 95,000,019 words.  Current word: Franksdragon69 hash value: hTJNpcnsdA9dK9wAEyinRA== Percent complete 35%
Tried 100,000,020 words.  Current word: gatsbyFyfcnfcbz hash value: l/7bFFMP5K6CJOqFKrnolA== Percent complete 36%
Tried 105,000,021 words.  Current word: gooberhookem hash value: RufPzTPwUwqMPan65x0qkA== Percent complete 38%
Tried 110,000,022 words.  Current word: kimballhanna hash value: YqnsRknbEdnZTm3tVXFwSA== Percent complete 40%
Dictionary Attack Result: helloworld
```

Caption: ↑ This is the result of using the MD5 algorithm with a Dictionary Attack to crack the password **helloworld**.



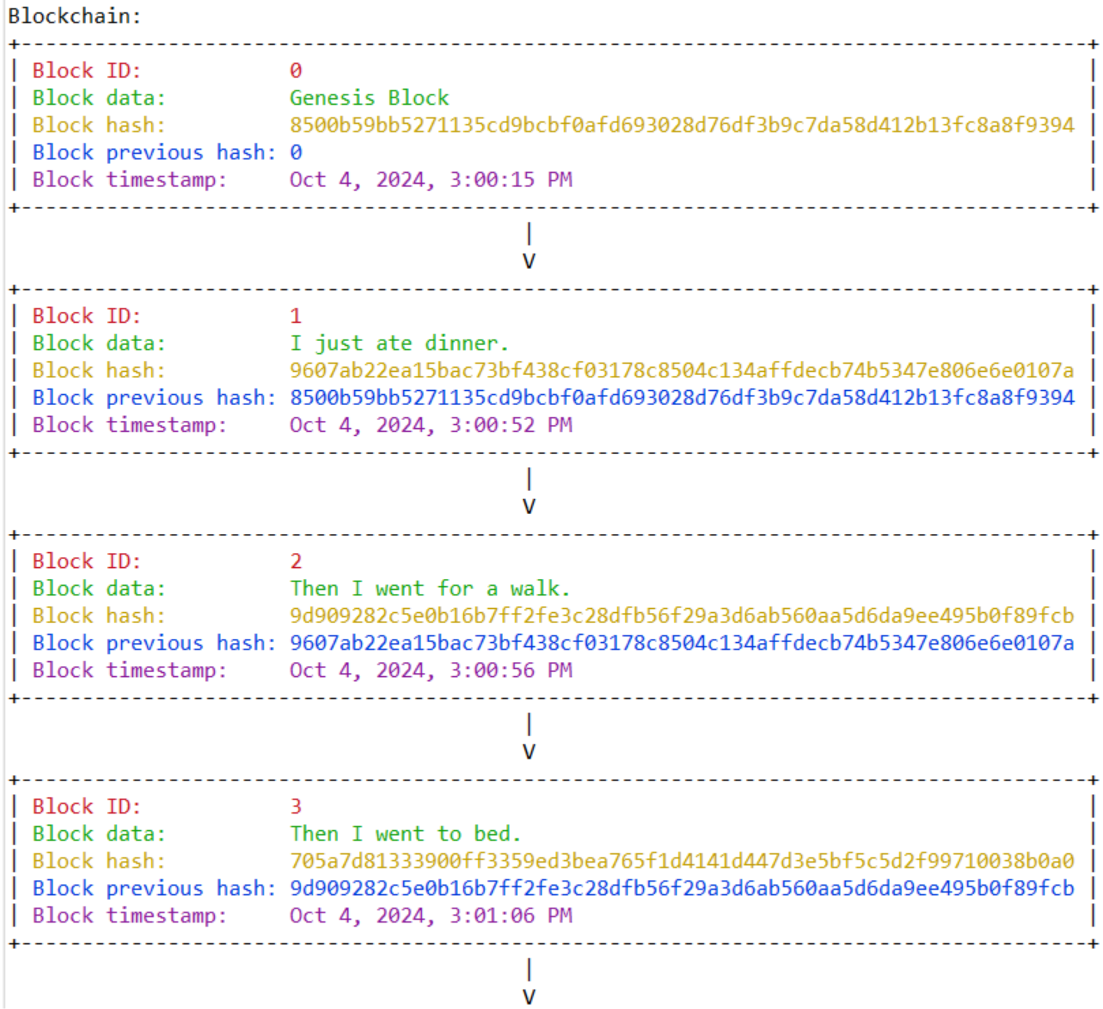
```
Count: 95,000,000 Guess: fz!]d Hash: F0P3LyOOUfHoNSqyEc+yqA== Time: 20 seconds. Avg Hashes per second: 4,750,000 Percent completed for length 5 10.37%.
Count: 100,000,000 Guess: f-98n Hash: wcyo34ufDQaEgNbOIyq9mw== Time: 21 seconds. Avg Hashes per second: 4,761,904 Percent completed for length 5 10.92%
Count: 105,000,000 Guess: gf8rx Hash: FnGUTrkSlm09D2ARMn2EHw== Time: 22 seconds. Avg Hashes per second: 4,772,727 Percent completed for length 5 11.46%
Count: 110,000,000 Guess: g07a7 Hash: JeeT49sl+Du03n9ATQbSKQ== Time: 24 seconds. Avg Hashes per second: 4,583,333 Percent completed for length 5 12.01%
Count: 115,000,000 Guess: g_5)* Hash: iMqpztJEC8z8/TbWnP/ByQ== Time: 25 seconds. Avg Hashes per second: 4,600,000 Percent completed for length 5 12.55%
Count: 120,000,000 Guess: hg42} Hash: Czc7y+SheZNFezI3rH6jwg== Time: 26 seconds. Avg Hashes per second: 4,615,384 Percent completed for length 5 13.10%
Count: 125,000,000 Guess: h13mb Hash: mXt7kZLmiy7LZp6doNMsew== Time: 27 seconds. Avg Hashes per second: 4,629,629 Percent completed for length 5 13.64%
Count: 130,000,000 Guess: h=1,l Hash: Ji9uUeocsWtH0PP9Dpc5KQ== Time: 28 seconds. Avg Hashes per second: 4,642,857 Percent completed for length 5 14.19%
Count: 135,000,000 Guess: ih0%v Hash: oEhT50u9yz69kjH3LoGbTA== Time: 29 seconds. Avg Hashes per second: 4,655,172 Percent completed for length 5 14.74%
Count: 140,000,000 Guess: i2zx5 Hash: UZKuO3f2/DtmFE4IhDCWGg== Time: 30 seconds. Avg Hashes per second: 4,666,666 Percent completed for length 5 15.28%
Count: 145,000,000 Guess: i+yg^ Hash: BdvuWDTp0X8NjcignPF1mw== Time: 31 seconds. Avg Hashes per second: 4,677,419 Percent completed for length 5 15.83%
Count: 150,000,000 Guess: jiw]] Hash: 7OePiZfSPkd1DaL4WqCYrg== Time: 32 seconds. Avg Hashes per second: 4,687,500 Percent completed for length 5 16.37%
Count: 155,000,000 Guess: j3v8? Hash: yBe5w5Wa4Ieo+9gh8h27+g== Time: 33 seconds. Avg Hashes per second: 4,696,969 Percent completed for length 5 16.92%
Count: 160,000,000 Guess: j[usj Hash: 2Nw0F0Lc/4XLdhw6a+kXOw== Time: 35 seconds. Avg Hashes per second: 4,571,428 Percent completed for length 5 17.46%
Count: 165,000,000 Guess: kjtbt Hash: GOTwzkbDvUKmPq0L7iZI8Q== Time: 36 seconds. Avg Hashes per second: 4,583,333 Percent completed for length 5 18.01%
Count: 170,000,000 Guess: k4r-3 Hash: 3EYJX8NfOVEytZge8EwUpQ== Time: 37 seconds. Avg Hashes per second: 4,594,594 Percent completed for length 5 18.56%
Brute Force Attack Result: k@8{4 is the password. xbcYCRDlMMDtUTUaT+d/wA== is the hash value. Which was hashed with MD5
```

Caption: ↑ This picture shows a brute force attack with MD5 finding the password **k@8{4**.



```
Dictionary Attack Selected
Trying all 28827 words in the dictionary
--------------------------------------------
Trying suffixes with every word in the dictionary 1,585,485 combinations
Tried 5,000,001 words.  Current word: breadition hash value: fJ7ItwXe+Rj1RF93octyTpr1RbI= Percent complete 78%
--------------------------------------------
Trying prefixes with every word in the dictionary 980,118 combinations
Dictionary Attack Result: ilovedogs
```

Caption: ↑ Here I tried to use a prefix **ilove** with the password **ilovedogs**, and it got cracked immediately by a dictionary attack.



```
Blockchain:
+---------------------------------------------------------------------------------------+
| Block ID:            0                                                                |
| Block data:          Genesis Block                                                    |
| Block hash:          8500b59bb5271135cd9bcbf0afd693028d76df3b9c7da58d412b13fc8a8f9394 |
| Block previous hash: 0                                                                |
| Block timestamp:     Oct 4, 2024, 3:00:15 PM                                          |
+---------------------------------------------------------------------------------------+
                                          |
                                          V
+---------------------------------------------------------------------------------------+
| Block ID:            1                                                                |
| Block data:          I just ate dinner.                                               |
| Block hash:          9607ab22ea15bac73bf438cf03178c8504c134affdecb74b5347e806e6e0107a |
| Block previous hash: 8500b59bb5271135cd9bcbf0afd693028d76df3b9c7da58d412b13fc8a8f9394 |
| Block timestamp:     Oct 4, 2024, 3:00:52 PM                                          |
+---------------------------------------------------------------------------------------+
                                          |
                                          V
+---------------------------------------------------------------------------------------+
| Block ID:            2                                                                |
| Block data:          Then I went for a walk.                                          |
| Block hash:          9d909282c5e0b16b7ff2fe3c28dfb56f29a3d6ab560aa5d6da9ee495b0f89fcb |
| Block previous hash: 9607ab22ea15bac73bf438cf03178c8504c134affdecb74b5347e806e6e0107a |
| Block timestamp:     Oct 4, 2024, 3:00:56 PM                                          |
+---------------------------------------------------------------------------------------+
                                          |
                                          V
+---------------------------------------------------------------------------------------+
| Block ID:            3                                                                |
| Block data:          Then I went to bed.                                              |
| Block hash:          705a7d81333900ff3359ed3bea765f1d4141d447d3e5bf5c5d2f99710038b0a0 |
| Block previous hash: 9d909282c5e0b16b7ff2fe3c28dfb56f29a3d6ab560aa5d6da9ee495b0f89fcb |
| Block timestamp:     Oct 4, 2024, 3:01:06 PM                                          |
+---------------------------------------------------------------------------------------+
                                          |
                                          V
```

Caption: ↑ This picture shows the result of me testing out the blockchain code with three simple blocks.