Security in Computing and Information Technology

COSC2550 (Semester 2, 2025)

Assignment 2

Group SIC_A1_77 (s4076955 - Calvier, s4015983 - Aiza)

Part A - Privacy Preserving Secure Models

1. (Calvier)

Given: $p = 107,\ q = 61,\ g = 7019$

The voting authority will generate the public key and private key:

$n = pq = 107 \cdot 61 = 6527$

$n^2 = 6527^2 = 42601729$

$\lambda = lcm(p-1, q-1) = lcm(106, 60) = 3180$

$u = g^{\lambda}\ mod\ n^2$

$u = 33568362$

$L(u) = \frac{u-1}{n} = \frac{33568362-1}{6527} = 5143$

$\mu = L(u)^{-1} mod\ n = (5143)^{-1} mod\ 6527 = 6145$

Public key: $(n,\ g) = (6527,\ 7019)$

Private key: $(\lambda, \mu) = (3180, 6145)$

The voting authority will send the public key to all voting booths and keep the private key.

In the voting booths, vote encoding:

ALICE: $0001\ 0000\ 0000 = 2^8 = 256$

BOB: $0000\ 0001\ 0000 = 2^4 = 16$

EVE: $0000\ 0000\ 0001 = 2^0 = 1$

Based on the assignment, 4 voters choose ALICE, 3 voters choose BOB and 3 voters choose EVE.

Paillier encryption for message m with randomness r:

$c \equiv g^m \cdot r^n \ (mod\ n^2)$

Voter No.	Voting message,m	Voter's Private Number, r	c
1	256	71	22787030
2	256	72	6155676
3	256	73	26435665
4	256	74	5713316
5	16	75	35730009
6	16	76	22037180
7	16	77	30448834
8	1	78	35591
9	1	79	31639803
10	1	90	28502453

The voting booth will send all the encrypted votes to the voting server to compute the votes.

The voting server will utilize Paillier's Additively homomorphic property, and multiplies all the cipher-texts to tally the votes:

C_{agg} = (22787030 * 6155676 * 26435665 * 5713316 * 35730009 * 22037180 * 30448834 * 35591 * 31639803 * 28502453) mod 42601729 = 30566829

The C_{agg} will be sent to the voting authority. The voting authority will decrypt it with the following formula:

$$m = L(C_{agg}^{\lambda} \bmod n^2) \cdot \mu \bmod n$$

$$m = L(30566829^{3180} \bmod 6527^2) \cdot 6145 \bmod 6527$$

$$m = L(2323613) \cdot 6145 \bmod 6527$$

$$m = 356 \cdot 6145 \bmod 6527$$

$$m = 1075$$

So the decrypted tally value is m =1075.

Now convert 1075 to 12-bit binary because voters are 12-bit one-hot:

$$1075 = 0100\ 0011\ 0011_2$$

First 4 bits (bits 8–11) → ALICE: 0100 → 4 votes

Next 4 bits (bits 4–7) → BOB: 0011 → 3 votes

Last 4 bits (bits 0–3) → EVE: 0011 → 3 votes

Result: ALICE =4=4, BOB =3=3, EVE =3=3 → Alice is the winner.

Part C - Analytical

Modernization of Hotel Check-In System with Blockchain

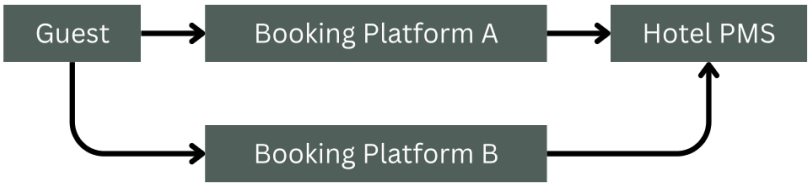
1. **Overview**

   Today, most hotels in Australia still use traditional property management systems (PMS) and centralised booking platforms. When a guest arrives, they must present their booking reference, passport or driver's license, and credit card. The hotel staff then re-enters these details into the PMS. This process raises several challenges:

   - Privacy risks: Guests hand over the same personal information repeatedly, and copies are stored across multiple hotel databases. Each database is a potential point of data breach.
   - Fraud and forgery: Fake bookings or duplicated confirmations can be presented by malicious actors, since there is no single trusted record.
   - Audit gaps: Regulators such as tourism boards, tax offices have limited visibility, relying on hotels to provide reports that could be incomplete or altered.

Scenario 1 - Duplicate Booking

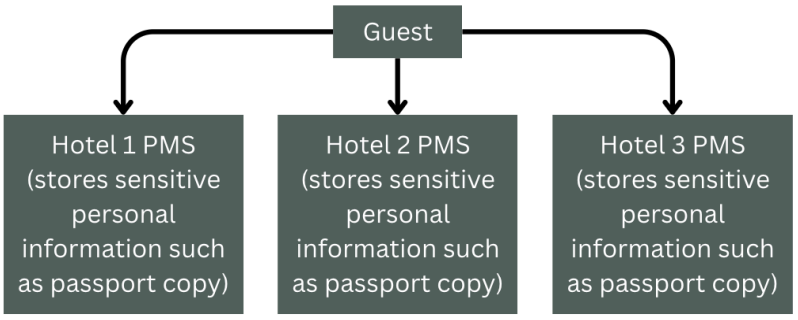
A guest books a hotel room through two different online platforms using slightly varied details. When they arrive, the hotel PMS sees both as valid because there is no shared ledger across platforms. This can lead to overbooking and revenue disputes.



Each PMS stores guest info separately.
No common record, privacy risks & duplicate bookings possible.

Scenario 2 - Identity Privacy Risk

A traveller checks into three different hotels during a trip. Each time, they hand over their passport and credit card, which are scanned and stored separately by each hotel. If any one of these hotel databases is hacked, the guest's identity is at risk.



Sensitive personal data is duplicated across multiple databases, increasing risk of identity theft.

These scenarios highlight the significance of introducing privacy-preserving techniques and a tamper-proof shared ledger. A blockchain-based approach ensures that only the minimum necessary data is shared, duplicate bookings cannot slip through, and regulators can independently verify check-in/out records without hotels having to hand over raw customer data.

2. Background (Existing approaches)

We know that legacy PMS databases are primarily centralised, hotel-controlled, and vulnerable to breaches. In this day and age, customers tend to use different third-party booking platforms to book rooms. This causes the trust gap between these platforms and hotels, as they hold sensitive personal and payment data. App check-ins are convenient but usually still backed by central databases with weak cross-hotel interoperability. These improve convenience but do not solve multi-party trust, privacy, or tamper-evident logging.

The well-known weaknesses of the existing approach:

- Single points of failure: If the central database is hacked or unavailable, the entire operation is affected.
- Poor interoperability: Different PMS vendors use proprietary data formats, making cross-chain or cross-brand verification difficult.
- Limited transparency: Regulators and auditors cannot easily confirm the accuracy of reported occupancy of taxation data.
- Privacy exposure: Data travels through multiple streams such as booking engines, payment gateways, and marketing tools, multiplying the risk of leakage.

Therefore, even with digital transformation, the hospitality sector remains largely centralised and vulnerable.

3. Aims and Objectives

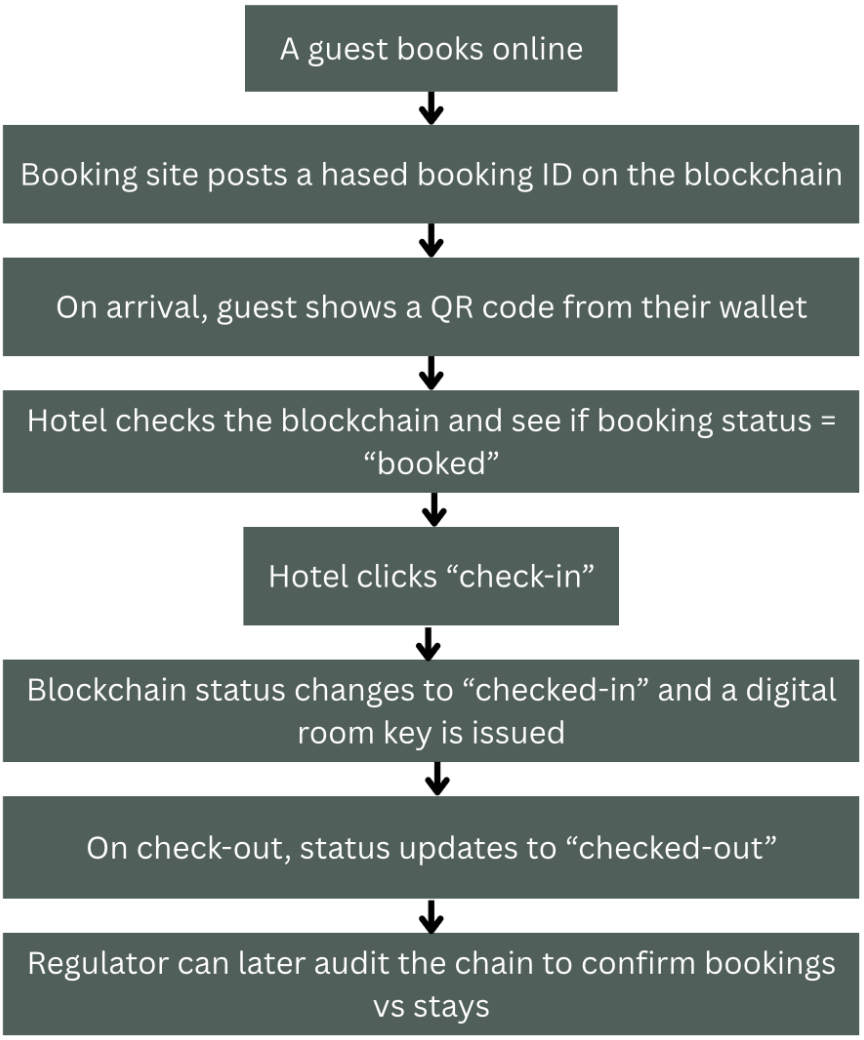
In my proposed system, I will aim to reduce unnecessary sharing of guest personal data, ensure booking authenticity to exterminate any possibility of forgery and duplicates, and create tamper-evident logs of check-ins/outs accessible to hotels and regulators.

The new features that will be introduced are permissional blockchain shared by hotel groups, booking platforms and regulators. Other than that, smart contracts to validate bookings, manage check-in/out state, and issue digital room keys. Additionally, guests are able to hold a self-sovereign digital identity wallet. For example, passports are verified once, then present Zero-Knowledge Proof (ZKP) to the front desk instead of raw ID.

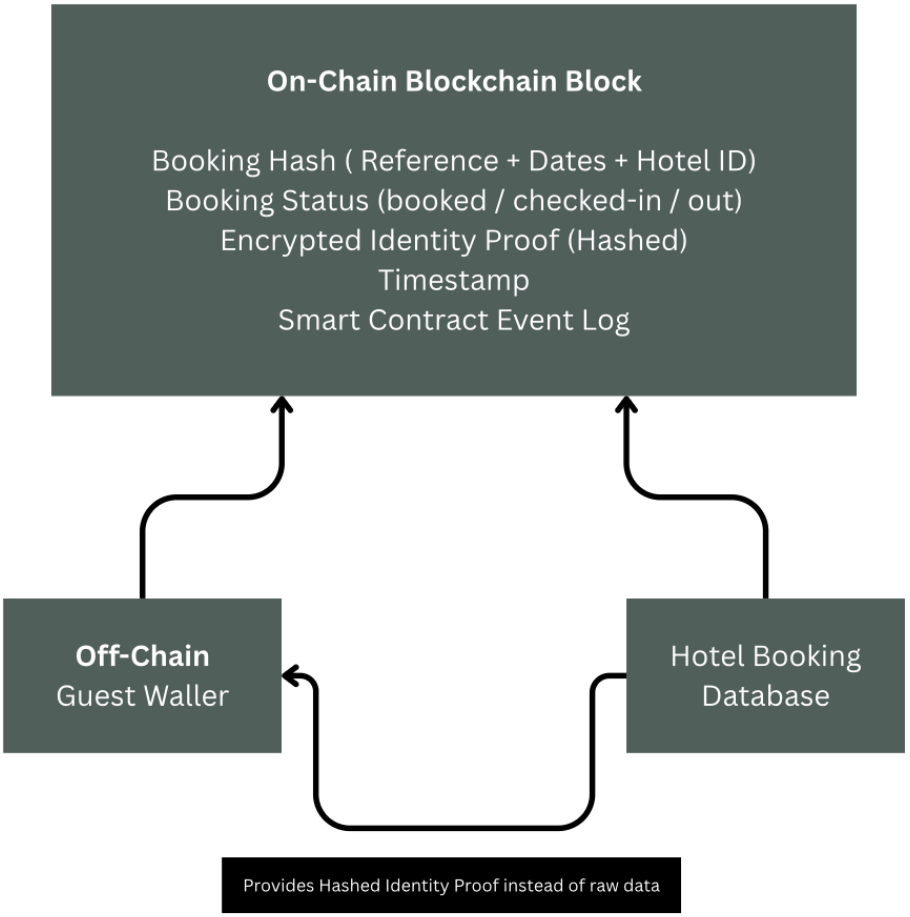
4. Proposed System

In the proposed blockchain-based hotel check-in system, all key players are hotels bookings websites, regulators and guests. They will connect through a shared permissioned blockchain. When a guest makes a booking online, the booking platform records a secure booking ID on the blockchain. This entry contains only minimal details such as the booking reference, hotel ID, and dates, while sensitive personal information such as names or payment details stays safely off-chain. On arrival, the guest simply shows a QR code stored in their phone wallet. The hotel staff can then check the blockchain to instantly verify that the booking is valid and has not already been used elsewhere. Once the guest is checked in the hotel updates the blockchain record to “checked-in”, and on departure it updates the status again to “checked-out”. Throughout this process, the blockchain provides a tamper-proof log of bookings and stays, giving hotels and booking platforms a common source of truth while also allowing regulators to audit the process without exposing private guest information.

Story Flow:


A guest books online
Booking site posts a hased booking ID on the blockchain
On arrival, guest shows a QR code from their wallet
Hotel checks the blockchain and see if booking status = “booked”
Hotel clicks “check-in”
Blockchain status changes to “checked-in” and a digital room key is issued
On check-out, status updates to “checked-out”
Regulator can later audit the chain to confirm bookings vs stays

Blockchain Diagram:



5. **Benefits**

Technological

- Integrity: Each booking and check-in/out event has a single authoritative record shared among all parties
- Security: Distributed ledger and cryptographic proofs eliminate single points of failure.
- Interoperability: A common protocol allows different hotel chains and platforms to verify each other's bookings in real time,
- Automation: Smart contracts remove human error.

Economic

- Faster settlement: Blockchain transactions can trigger automatic payment releases between booking sites and hotels once stays are confirmed.
- Reduced administrative overhead: Built-in audit logs reduce manual reporting for taxation and compliance.
- Lower fraud cost: Duplicate or forged bookings are detected instantly, preventing revenue loss.

Social

- Enhanced privacy: Guests control their own identity through digital wallets, only proofs are shared.
- Improved guest experience: Check-in becomes almost contactless and instantaneous.
- Regulatory confidence: Transparent records help authorities monitor tourism metrics accurately without invasive data collection.

6. Advantages and Limitations

Advantages:

- Tamper-proof record of all stays.
- Less personal data exposed at reception.
- Easier audits for regulators.
- Hotels and booking sites share one “source of truth.”

Limitations:

- Adoption barrier: Hotels and booking sites all need to join the system.
- Use-experience challenge: Guest wallet apps must be simple, or guests won’t use them.
- Integration complexity - Hotels still need to connect their blockchain node to existing door-lock and PMS software, requiring technical investment.

7. Conclusion

Introducing blockchain technology into hotel check-in modernises one of the most outdated yet sensitive processes in the hospitality industry. The proposed permissioned ledger eliminates duplicated bookings, strengthens privacy protection, and provides verifiable transparency for regulators.

Guests retain control of their personal data, hotels gain a shared, trustworthy record of operations, and authorities can perform audits without breaching confidentiality.

While success depends on the ability to maintain, govern, interoperability, and user adoption, this design demonstrates a practical pathway for privacy-preserving digital transformation of Australia's hospitality sector, delivering technological innovation, economic efficiency, and public trust.

Reference

1] M. Barten, “Blockchain Technology and Its Uses in the Hospitality Industry,” Revfine.com, Jun. 04, 2024. [Online]. Available: https://www.revfine.com/blockchain-technology-hospitality-industry/ Revfine.com

[2] L. Hall, “How Outdated Legacy Systems Can Limit Your Hotel’s Revenue Potential,” Hotel Technology News, Feb. 22, 2024. [Online]. Available: https://hoteltechnologynews.com/2024/02/how-outdated-legacy-systems-can-limit-your-hotels-revenue-potential/ Hotel Technology News

[3] [Lecture Week 8], “Blockchain Applications in Business,” COSC2536, RMIT University, S2 2025

[4] [Lecture Week 5], “Secure E-voting” COSC2536, RMIT University, S2 2025

Question 7

Question 7.1

```
Enter your Student ID (numbers only): 4015983
Student ID: 4015983
Data Encryption by ABC tech, developed in 2000.
Value of public exponent is fixed at 65537.
Enter prime 1 (>1,000,000): 4015987
Enter prime 2 (>1,000,000): 4076981
Enter data (alphabets only): abf
Public Key:  (65537, 16373102695247)
Private Key:  (9964211134433, 16373102695247)
Cipher Text:  15403182597215
m:  6382182
Plain Text:  abf
Data written to E:\MAC-1\PartC_Q7\windows\Cipher\key.txt
Data written to E:\MAC-1\PartC_Q7\windows\Cipher\cipher.txt

Exiting!
.
Press any key to continue . . .
```

The program is encrypting plaintext using textbook RSA. It takes 2 chosen prime numbers p = 4015987 and q = 4076981 to creat key n as shown below:

$$n = p * q$$

$$n = 4015987 * 4076981$$

$$n = 16373102695247$$

With that, we calculate the inverse modulo d of the totient φ (p-1)(q-1) as shown below with chosen prime number e given already by the program:

$$\varphi = (p - 1)(q - 1)$$

$$\varphi = 4015986 * 4076980$$

$$\varphi = 16373094602280$$

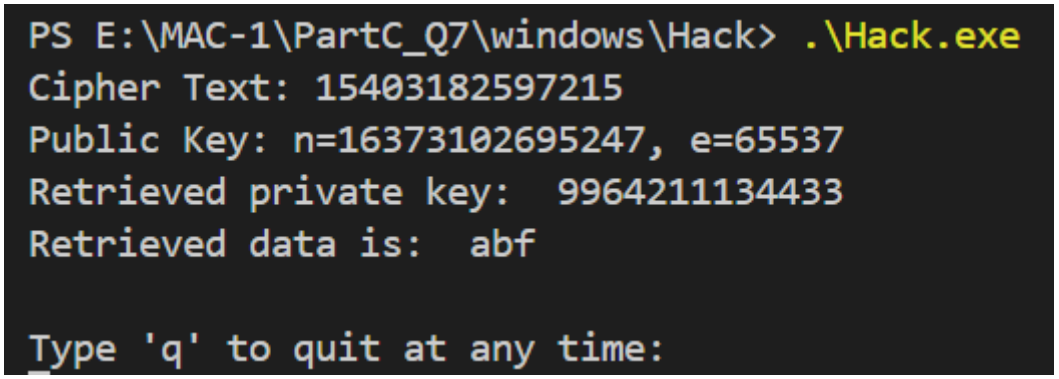
$$d = e^{-1} mod\, \varphi$$

$$d = 65537^{-1} mod\, 16373094602280$$

$$d = 9964211134433$$

Using these values, private keys p, q, and n and private key d to encrypt and decrypt the message.

When we executed Hack.exe, the results show as below:



It shows that the data available to Trudy were the private keys n and e and the ciphertext C. From that, Trudy could factor n to find values p and q. Through trial and error, Trudy would them simply need to compute $\varphi(n) = (p-1)(q-1)$. Now we can see that Trudy is simply doing the equation in reverse. She could then simply find outthe private key using the formula $d = e^{-1} mod\, \varphi$ and then use the private key d to decrypt the Ciphertext C to get message/plaintext.

This shows that the secure communication is being compromised because the method of encryption uses textbook RSA where it's weaknesses shines when prime numbers chosen are small enough for attackers to deduce the private keys just by simply factorizing n and reversing the equations the program has to go through when decrypting the message. The primes used were also predictable and had no modern padding.

Question 7.2

Concatenaed ID: 40159834076955

First Prime greater than 40159834015983: 40159834015991

Second Prime greater than 40769554076955: 40769554076987

Here are the results when using these values to execute:

```
Enter your Student ID (numbers only): 40159834076955
Student ID: 40159834076955
Data Encryption by ABC tech, developed in 2000.
Value of public exponent is fixed at 65537.
Enter prime 1 (>1,000,000): 40159834015991
Enter prime 2 (>1,000,000): 40769554076987
Enter data (alphabets only): abf
Public Key:  (65537, 1637298524637767079403099117)
Private Key:  (732895761459498617526591393, 1637298524637767079403099117)
Cipher Text:  1032161288422310742854528792
m:  6382182
Plain Text:  abf
Data written to E:\MAC-1\PartC_Q7\windows\Cipher\key.txt
Data written to E:\MAC-1\PartC_Q7\windows\Cipher\cipher.txt
```

```
Cipher Text: 1032161288422310742854528792
Public Key: n=1637298524637767079403099117, e=65537
Traceback (most recent call last):
  File "cipher_breaker2.py", line 99, in <module>
  File "cipher_breaker2.py", line 96, in main
  File "asyncio\runners.py", line 194, in run
  File "asyncio\runners.py", line 62, in __exit__
  File "asyncio\runners.py", line 70, in close
  File "asyncio\runners.py", line 206, in _cancel_all_tasks
  File "asyncio\base_events.py", line 712, in run_until_complete
  File "asyncio\base_events.py", line 683, in run_forever
  File "asyncio\base_events.py", line 2050, in _run_once
  File "asyncio\events.py", line 89, in _run
  File "cipher_breaker2.py", line 90, in main_async
  File "asyncio\runners.py", line 195, in run
  File "asyncio\runners.py", line 118, in run
  File "asyncio\base_events.py", line 712, in run_until_complete
  File "asyncio\base_events.py", line 683, in run_forever
  File "asyncio\base_events.py", line 2050, in _run_once
  File "asyncio\events.py", line 89, in _run
  File "cipher_breaker2.py", line 73, in start_cracking
r2' due to unhandled exception!
Task exception was never retrieved
```

At first glance, the difference between the first ciphertext generated without concatenating the id was significantly shorter, and that Hack.exe took significantly less time to execute (comparred to the execution done in 7.1). While after concatenating the ID and choosing larger prime numbers for p and q, after waiting for 5 minutes, I decided to cancel the execution since it was not executing or needed significantly longer time to calculate the private keys and second the ciphertext. It comes to show that the larger a prime numbers chosen, the more complicated it is to calculate the private key in order to decrypt the ciphertext. This is because in order to hack, the attacker would need to factor n in order to find p and q to calculate the private key d. The prime numbers chosen create the arithmetic space in order to calculate the message (m) as shown : $0 \leq m < n$

This increases the security of the algorithm because the bigger the arithmetic space, the more calculations needed to be done in order to find the message (in this case, factorising n to find p and q). This is especially true since RSA's security fundamentally relies on how large n is. The lager it is, the more time needed to calculate d, thus increasing its security. This is shown in the second image where Hack.exe fails to compute and decrypt th emessage within the given timeframe.

Question 7.3

- Available under PartC_Q7.3.py

```
PS E:\MAC-1\PartC_Q7\windows\Hack> & C:/Users/User/AppData/Local/Programs/Pytho
hon313/python.exe e:/MAC-1/PartC_Q7.3.py
Enter data (alphabets only): abc
Generating RSA keys
Public key location: e:\MAC-1\key.txt
Ciphertext location: e:\MAC-1\cipher.txt
Doneee!
```

Uses RSA padding because the initial cipher uses textbook rsa which is less secure. It would generate random prime numbers for public and private keys instead of asking the user and only prompts the user to enter the plaintext. Will also generate key.txt and cipher.txt just like the previous cipher.

References:

1. Introduction Textbook RSA Attacks on RSA Padded RSA Textbook RSA And its insecurities. (2016). Available at: https://cs.wellesley.edu/~cs310/lectures/26_rsa_slides_handouts.pdf.
2. Tencentcloud.com. (2025). *What are the disadvantages of the RSA encryption algorithm? - Tencent Cloud*. [online] Available at: https://www.tencentcloud.com/techpedia/102501.
3. Tencentcloud.com. (2025). *What are the disadvantages of the RSA encryption algorithm? - Tencent Cloud*. [online] Available at: https://www.tencentcloud.com/techpedia/102501.
4. in (2020). *Relationship between N and $\phi(N)$ in the RSA algorithm*. [online] Cryptography Stack Exchange. Available at: https://crypto.stackexchange.com/questions/78082/relationship-between-n-and-phin-in-the-rsa-algorithm.