




## School of Computing Technologies

# INTE2627 Blockchain Technology Fundamentals

## Assignment 2

	<b>Assessment Type:</b> Group assignment. A maximum of 2 people are allowed in a group. Submit online via Canvas → Assignments → Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	<b>Due date:</b> Week 8, Friday the 2 <sup>nd</sup> May 2025 11:59 pm Deadlines will not be advanced, but they may be extended. Please check Canvas → Syllabus or via Canvas → Assignments → Assignment 2 for the most up-to-date information. As this is a major assignment in which you demonstrate your understanding, a university standard late penalty of 10% per each working day applies for up to 5 working days late unless special consideration has been granted.
	<b>Weighting:</b> 50 marks (Contributes 50% of the total Grade)

### 1. Overview

The objective of Assignment 2 is to evaluate your understanding of the topics covered in Lectures 1-7. These topics include the basic building blocks of Blockchain, building a Blockchain platform with these blocks, and querying on Blockchain. Assignment 2 will focus on developing your ability to implement and integrate advanced security mechanisms in Distributed Ledger Technology (DLT).

Assignment 2 contains one comprehensive problem: the continuation and implementation of Assignment 1 with added requirements. You are required to prepare the solutions with a step-by-step description in a single PDF file, including the necessary code implementations.

Develop the solution for this assignment in an iterative fashion (rather than completing it in one sitting). You should start preparing your answers immediately after Lecture 5 (Week 5). By the end of each week, starting from Week 4 to Week 7, you should aim to solve at least one question.

If there are questions, you must ask them via **the relevant Canvas discussion forums** in a general manner.

**Overall, you must follow the following special instructions:**

- You must fulfill the requirements in the questions.
- Create a **group** of a **maximum of 2 people** and perform the tasks as a group.
- For the questions that require implementation, you must implement the functionalities stated in the questions. Any change in a user interface is acceptable if the functionality is there.
- Your work **must be demonstrated to your tutor**. The date and time will be announced separately.  
**No demo, no marks.**
- **You must submit the solutions as a report on CANVAS.** In your solution, you must show all the steps with the necessary code segments and screenshots for each question.
- Upload your solution as a single **PDF or Word document** in CANVAS. Also, **upload codes as a single ZIP file in the CANVAS.**
- **Do not put the PDF within the ZIP file.**

## 2. Assessment Criteria

This assessment will determine your ability to:

- Follow the requirements provided in this document and in the lessons.
- Work in a group or individually to solve a problem by using concepts taught over the first seven weeks of the course.
- Meeting deadlines.

## 3. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

- **CLO 1:** Explain the fundamental concepts of blockchain technology, such as its structure, applications, and data immutability.
- **CLO 2:** Compare and contrast different types of blockchains, including public, private, and permissioned, and analyze their use in various industries.
- **CLO 3:** Demonstrate a deep understanding of decentralized systems, hash functions, cryptography, and consensus algorithms and their role in blockchain technology.
- **CLO 4:** Analyse and evaluate advanced aspects of the technology, such as data retrieval (e.g., query) on the blockchain, etc.

## 4. Assessment details

Please ensure that you have read **Sections 1 to 3** of this document before going further. Assessment details are provided on the [next page](#).

## Demonstrating a Secure DLT-Based Inventory Management System

This assignment builds upon Assignment 1 by expanding the requirements to implement and demonstrate key cryptographic and consensus mechanisms through real code.

By the end of **Assignment 1**, we established that there are **four different inventories**, each maintaining **four records stored in their respective databases**. In **Assignment 2**, the following scenario outlines the secure operations that your system must implement:

### 1. New Record Submission (Part 1, Task 1)

- a. When an inventory node wants to add a new record to the distributed inventory system, it must first be authenticated by the other nodes to ensure:
  - i. The request originates from a legitimate inventory node (not an external adversary).
  - ii. The record was not tampered with during transmission.

### 2. Consensus Protocol Execution: (Part 1, Task 2)

- a. A consensus protocol is executed to validate and confirm the new record across the network.
  - i. This ensures immutability and global agreement among all inventory nodes.
  - ii. Only when consensus is reached, the record permanently stored in the blockchain ledger.

### 3. Multi-Signature-Based Query Verification: (Part 2)

When an authorised user (e.g., Procurement Officer) queries the distributed inventory system for a specific item's quantity, all inventory nodes must jointly verify the queried record using the Harn identity-based multi-signature scheme.

- i. This guarantees that the returned data is accurate, untampered, and collectively approved.
- ii. It prevents malicious or faulty nodes from providing inconsistent or forged results.

### 4. Data Encryption Before Response Transmission:(Part 2)

- a. After the queried data has been jointly verified and approved by all inventory nodes, it must be encrypted before being sent to the authorised user.
  - i. The data is encrypted using the query user's public key, ensuring that only the intended recipient can decrypt and access the information.
  - ii. This protects sensitive inventory data during transmission and prevents eavesdropping or data leakage on the network.

## Part 1

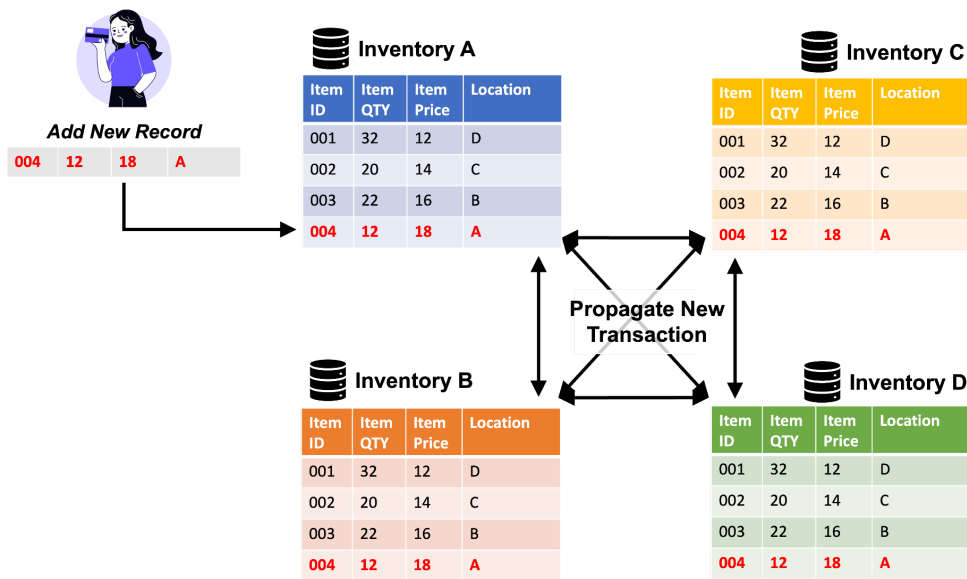


Figure 1. Scenario Overview for Part 1

### Task 1: RSA Digital Signature Implementation (10 Marks)

Inventory A/B/C/D has purchased 32/20/22/12 units of item with ID 001/002/003/004, priced at 12/14/16/18. Inventory A/B/C/D wants to add this new record to the blockchain-based inventory system (see the Diagram above). Before broadcasting the record, Inventory A/B/C/D must digitally sign it to ensure authenticity and integrity.

**What you need to do:**

- Key setup:** Use the variables **p**, **q**, and **e** (as provided in the “List of Keys” document) to generate the RSA key pair {PK, SK} for Inventory A, B, C, and D. These values must be hardcoded in your program.
- Signing:** Inventory A/B/C/D signs the new record.
- Verification:** Other inventories verify the signature.

### Task 2: Consensus Protocol Integration (10 Marks)

To ensure that all inventories agree on updates and to prevent unauthorized modifications, a consensus protocol must be integrated into the system.

**What you need to do:**

- Consensus Protocol Justification:**
  - Choose a suitable consensus protocol depending on the scenario given (e.g., PoW, PoS, PoA, etc.).
  - Explain why it fits this current scenario. The justification needs to be written in the report.
- Implementation:**
  - Implement the chosen consensus protocol to reach an agreement on whether a new record should be accepted.

- b. Once the consensus process is finished, each inventory stores the records in its corresponding database.
- c. For simplicity, you may exclude Merkle tree computations and are not required to add the verified block to the blockchain.

**Part 1 Note:** Both Task 1 and Task 2 codes need to be integrated into one single system. Ensure that you use separate files for the inventory records and for storing all parameters required for the calculation. Any values specified in the “List of Keys” document must be hardcoded into your code.

## Part 2

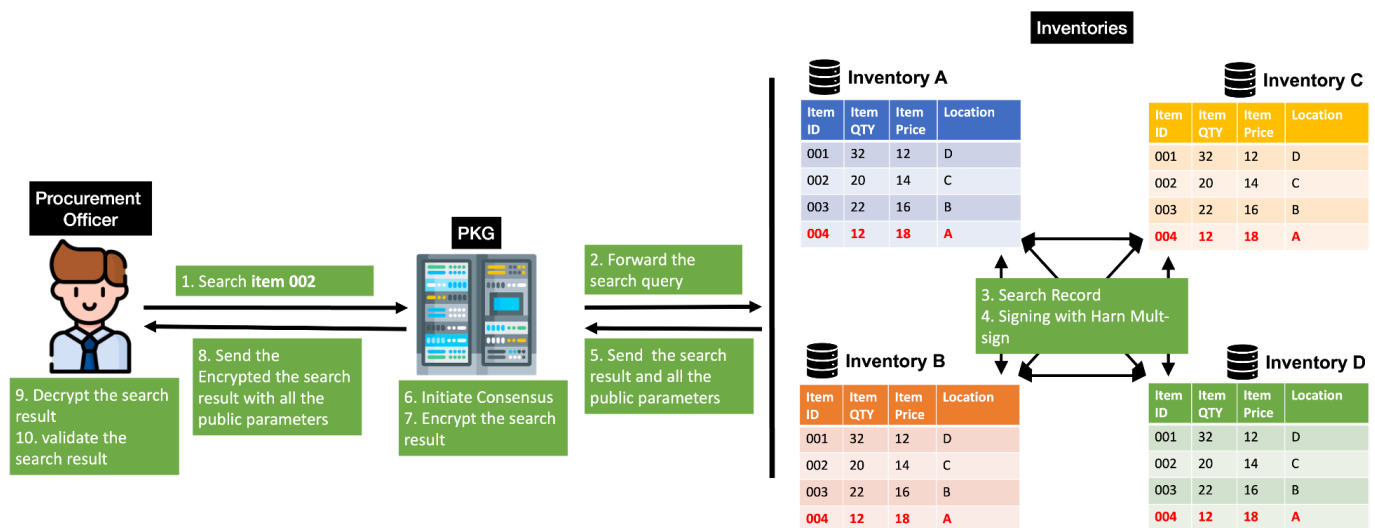


Figure 2. Scenario Overview for Part 2

### Task 3: Multi-Signature Query Verification & Secure Delivery (30 Marks)

When an authorised external user (e.g., Procurement Officer) queries the distributed inventory system for the quantity of a specific item ID (e.g., 004), all inventory nodes must jointly verify and approve the result using the Harn identity-based multi-signature scheme. After the result is verified, it must be encrypted and securely delivered to the user.

**What you need to do:**

1. **Key Setup:**
  - a. The PKG should compute each required key using the provided values of  $p$ ,  $q$ , and  $e$  (as specified in the 'List of Keys' document), based on Harn's identity-based multi-signature algorithm.
2. **User Query Submission:**
  - a. The Procurement Office sends a request to retrieve the quantity of a specific item (e.g., enter an Item ID 002) from the distributed inventory system.
3. **Inventory Verification & Multi-Signature Authentication:**
  - a. Each inventory node searches its local database for the requested item record.
  - b. Each inventory generates partial signatures and combines their signatures with the Harn identity-based multi-signature algorithm.

- c. A consensus process is required to ensure that all parties obtain the same aggregated signature, which will be coordinated by the PKG.

**4. Response Encryption, Delivery, and Multi-Sign Verification:**

- a. The response message is encrypted by PKG using the corresponding RSA key.
- b. The encrypted message response, along with multi-sign parameters, is sent to the user.
- c. The user decrypts the response using the corresponding RSA key.
- d. The user verifies the multi-signature

**Part 2 Note:** For Task 3, ensure that you use separate files for the inventory records and for storing all parameters required for the calculation. Any values specified in the “List of Keys” document must be hardcoded into your code.

### **Special Requirement**

- You **MUST NOT** use any existing libraries for automatic signing and verification processes. However, you may use standard libraries for other functionalities, such as hashing, generating large prime numbers, etc.
- All tasks must include a **clear and functional UI** that demonstrates the specified functionalities. You are free to use **any programming language** for the backend/UI/front-end development.

### **Implementation Details**

- **Backend:** Implement the core logic using a suitable backend technology such as **Python (Flask/Django), PHP, Java, or any other programming language**.  
The backend should handle consensus, signature generation, and verification.
- **Frontend:** Develop a **user-friendly interface** using **HTML, CSS, JavaScript**, or any other preferred framework. The frontend should seamlessly interact with the back end to execute the required functionalities.  
The frontend must have input fields to accept any user input for adding new records (Part 1), search query (Part 2).
- **Database:** You are not required to use existing database tools (e.g., MariaDB, MySQL), but you must simulate data storage appropriate to the scenario. This can be done using files (e.g., .txt, .json, etc.) to represent your databases.

### **Demonstration**

A **live demonstration** of the complete prototype is required. You must showcase the **entire process**, including **record creation, digital signing, multi-signature authentication, consensus agreement, and final blockchain storage**. Failure to present a live demo will result in no marks being awarded.

### **Submission**

- Submit a **document (PDF format)** and a **ZIP file** containing your full source code.
- Ensure that your code is **well-documented and properly structured**. Use comments and clear organization to help reviewers understand your design choices and implementation.

### **Documentation**

Prepare a **detailed report (PDF format)** that includes:

- **Introduction:** A clear explanation of the assignment's objectives and scope.
- **Implementation Details:** Step-by-step explanations of your approach, **including key code snippets and descriptions** of how different components work. Mention the technologies, tools, and libraries used.
- **Screenshots & Workflow:** Include annotated screenshots demonstrating **each stage** of signing, verifying, and achieving consensus on inventory records. Clearly describe how your application functions.

Both the **PDF report** and the **ZIP file containing the source code** must be submitted via **Canvas** before the deadline to avoid any penalties.

## 5. Academic integrity and plagiarism (standard warning)

Academic integrity is about the honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarized, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate reference as if they were your own. RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the [University website](#).

## 6. Assessment declaration

When you submit work electronically, you agree to the [assessment declaration](#).



## 7. Rubric/assessment criteria for marking

Criteria	Ratings					Total
<b>Task 1 Digital Signature Implementation</b>	RSA key pair is correctly generated (or hard-coded) with all required parameters.	RSA key pair is correctly generated (or hard-coded) with all required parameters.	RSA key pair is correctly generated (or hard-coded) with all required parameters.	Incomplete or flawed implementation of signing/verification.	No relevant work submitted.	<b>10 pts</b>
	The new inventory record is properly signed.	The new inventory record is signed, with some minor implementation flaws.	Some components may not function as intended.	Code is largely non-functional.	No attempt or no participation in demonstration.	
	Signature verification works correctly.	Signature verification mostly works.	Little or no documentation.	During demonstration, student(s) struggles to answer basic questions or shows limited understanding during demonstration.	<b>0-2 pts</b>	
	Code is clean, functional, and well-documented.	Code has minimal or unclear documentation.	During demonstration, student(s) give partially correct answers during demonstration, but lacks clarity or misses key concept.			
	During demonstration, student(s) accurately answers all related questions, showing strong understanding.	During demonstration, student(s) answers most questions correctly, with minor misunderstanding.	<b>4-6 pts</b>	<b>2-4 pts</b>		
	<b>8-10 pts</b>	<b>6-8 pts</b>				

<b>Task 2 Explanation and Integration of Consensus Protocol</b>	<p>Clearly explains the chosen consensus protocol, including its mechanism, benefits, and why it suits this scenario.</p> <p>Shows strong understanding of latency, security, and fault tolerance trade-offs.</p> <p>Correctly implements the protocol to handle agreement across nodes.</p> <p>Code is clean, functional, and well-documented.</p> <p>During demonstration, student(s) accurately answers all related questions, showing strong understanding.</p> <p><b>8-10 pts</b></p>	<p>Explains the consensus protocol with minor gaps in mechanism or justification.</p> <p>Justifies the choice with generally relevant points.</p> <p>Implementation is mostly correct, with only small logic or integration issues.</p> <p>Code has minimal or unclear documentation.</p> <p>During demonstration, student(s) answers most questions correctly, with minor misunderstanding.</p> <p><b>6-8 pts</b></p>	<p>Provides only a basic or partially correct explanation of the consensus protocol.</p> <p>Weak or limited justification for its use in this system.</p> <p>Implementation has major flaws in logic or fails to properly coordinate consensus.</p> <p>Little or no documentation.</p> <p>During demonstration, student(s) give partially correct answers during demonstration, but lacks clarity or misses key concept.</p> <p><b>4-6 pts</b></p>	<p>Incomplete or incorrect description of the protocol.</p> <p>Little or no justification.</p> <p>Implementation fails or is mostly missing; no working consensus logic.</p> <p>Code is non-functional and undocumented.</p> <p>During demonstration, student(s) struggles to answer basic questions or shows limited understanding during demonstration.</p> <p><b>2-4 pts</b></p>	<p>No relevant work submitted.</p> <p>No attempt or no participation in demonstration.</p> <p><b>0-2 pts</b></p>	<b>10 pts</b>

<b>Task 3</b> <b>Implementation of Multi-Signature-Based Authentication and Encryption &amp; Decryption</b>	<p>Correctly implements the Harn identity-based multi-signature scheme across all inventory nodes.</p> <p>All nodes generate valid partial signatures, which are correctly aggregated and verified.</p> <p>A consensus check is correctly implemented to ensure all aggregated signatures are consistent across nodes.</p> <p>RSA encryption is correctly implemented; encrypted data can be successfully decrypted.</p> <p>Code is clean, functional, and well-documented.</p> <p>During demonstration, student(s) accurately answers all related questions, showing strong understanding.</p> <p><b>25-30 pts</b></p>	<p>Multi-signature is mostly implemented with minor flaws or inconsistencies in aggregation or verification.</p> <p>Consensus checking is present but may be partially flawed or inconsistently applied.</p> <p>RSA encryption and decryption mostly works, with small logic issues or unclear data flow.</p> <p>Code has minimal or unclear documentation.</p> <p>During demonstration, student(s) answers most questions correctly, with minor misunderstanding.</p> <p><b>20-25 pts</b></p>	<p>Basic implementation attempted but with notable issues in either signature handling or encryption logic.</p> <p>Consensus step is attempted but lacks correctness or is inconsistently used.</p> <p>Verification or decryption may fail partially; implementation lacks reliability.</p> <p>Code is partially functional but poorly structured or undocumented.</p> <p>During demonstration, student(s) give partially correct answers during demonstration, but lacks clarity or misses key concept.</p> <p><b>15-20 pts</b></p>	<p>Limited or flawed implementation of the multi-signature process or encryption.</p> <p>Consensus check is missing or incorrectly implemented.</p> <p>Signature verification or decryption mostly fails or is incomplete.</p> <p>Code is non-functional and undocumented.</p> <p>During demonstration, student(s) struggles to answer basic questions or shows limited understanding during demonstration.</p> <p><b>10-15 pts</b></p>	<p>No relevant work submitted.</p> <p>No attempt or no participation in demonstration.</p> <p><b>0-10 pts</b></p>	<p><b>30 pts</b></p>

