* Alternative Blockchains:
* Blockchains Blockchain-Outside of Currencies:
* Internet of Things,
* Government, Health, Finance, Media

**Alternative Blockchains**

**Blockchains**

* This section will give an introduction to new blockchain solutions.
* First, a new blockchain named Kadena is discussed in the following section.

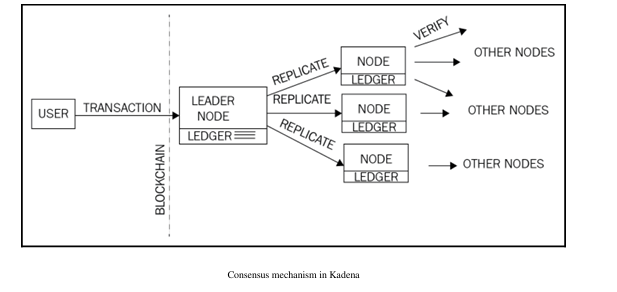
**Kadena**

* Kadena is a recently-introduced private blockchain that has successfully addressed scalability and privacy issues in blockchain systems.
* A new Turing incomplete language called Pact has also been introduced with Kadena that allows the development of smart contracts.
* A key innovation in Kadena is its Scalable BFT consensus algorithm, which has the potential to scale to thousands of nodes without performance degradation.
* Scalable BFT is based on the original Raft algorithm and is a successor of Tangaroa and Juno.
* Tangaroa, which is a name given to an implementation of Raft with fault tolerance (a BFT Raft), was developed to address the availability and safety issues that arose from the behavior of byzantine nodes in the Raft algorithm, and Juno was a fork of Tangaroa that was developed by JPMorgan.
* Consensus algorithms are discussed in Chapter 1, Blockchain 101 in more detail.
* Both of these proposals have a fundamental limitation – they cannot scale while maintaining a high level of high performance.
* As such, Juno could not gain much traction.
* Private blockchains have the more desirable property of maintaining high performance as the number of nodes increase, but the aforementioned proposals lack this feature.
* Kadena solves this issue with its proprietary Scalable BFT algorithm, which is expected to scale up to thousands of nodes without any performance degradation.
* Moreover, confidentiality is another important aspect of Kadena that enables privacy of transactions on the blockchain.
* This is achieved by using a combination of key rotation, symmetric on-chain encryption, incremental hashing, and Double Ratchet protocol.
  + Key rotation is used as a standard mechanism to ensure security of the private blockchain.
  + It is used as a best practice to thwart any attacks if the keys have been compromised, by periodically changing the encryption keys.
  + There is a native support for key rotation in Pact smart contract language.
  + Symmetric on chain encryption allows encryption of transaction data on the blockchain.
  + These transactions can be automatically decrypted by the participants of a particular private transaction.
  + Double Ratchet protocol is used to provide key management and encryption functions.

**Alternative Blockchains**

* Scalable BFT consensus protocol ensures that adequate replication and consensus has been achieved before smart contract execution.
* Consensus is achieved by following the process described below:
  1. First, a new transaction is signed by the user and broadcasted over the blockchain network, which is picked up by a leader node that adds it to its immutable log. At this point, an incremental hash is also calculated for the log. Incremental hash is a type of hash function that basically allows computation of hash messages in the scenario where, if a previous original message which is already hashed is slightly changed, then the new hash message is computed from the already existing hash. This scheme is quicker and less resource intensive compared to a conventional hash function where an altogether new hash message is required to be generated even if the original message has only changed very slightly.
  2. Once the transaction is written to the log by the leader node, it signs the replication and incremental hash and broadcasts it to other nodes.
  3. Other nodes, after receiving the transaction, verify the signature of the leader node, add the transaction into their own logs, and broadcast their own calculated incremental hashes (quorum proofs) to other nodes. Finally, the transaction is committed into the ledger permanently after an adequate number of proofs are received from other nodes.

A simplified version of this process is shown in the following diagram, where the leader node is recording the new transactions and then replicating them to the follower nodes:



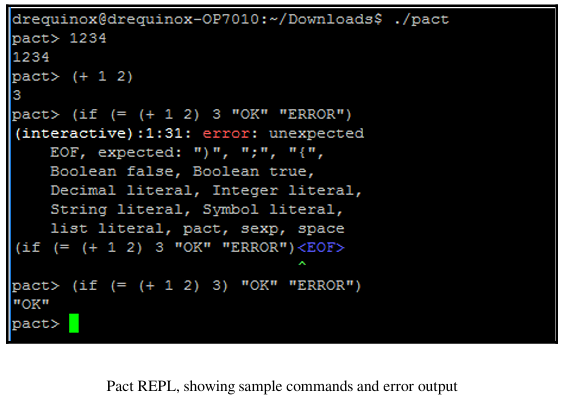
* **Consensus mechanism in Kadena**

**Alternative Blockchains**

* Once the consensus is achieved, smart contract execution can start and takes a number of steps, as follows:
  1. First, the signature of the message is verified.
  2. Pact smart contract layer takes over.
  3. Pact code is compiled.
  4. The transaction is initiated and executes any business logic embedded within the smart contract.
     + In case of any failures, an immediate rollback is initiated that reverts the state back to what it was before the execution started.
  5. Finally, the transaction completes and relevant logs are updated.

**Pact Open Source**

* Pact has been open sourced by Kadena and is available for download at <http://kadena.io/pact/downloads.html>.
* This can be downloaded as a standalone binary that provides a REPL for Pact language.
* An example is shown below where Pact is run by issuing the command ./pact in a Linux console.



**Smart Contract in Pact**

* Smart contract in Pact is generally composed of three sections:
  1. **Keysets**:
     + Defines relevant authorization schemes for tables and modules.
  2. **Modules**:
     + The smart contract code encompassing the business logic in the form of functions and Pacts.
  3. **Tables**:
     + Used for data storage.
* **Pacts within Modules**:
  1. Composed of multiple steps.
  2. Executed sequentially.

**Execution Modes in Pact**

* Pact can be used in several execution modes:
  1. **Contract Definition Mode**:
     + Allows a contract to be created on the blockchain via a single transaction message.
  2. **Transaction Execution Mode**:
     + Entails the execution of modules of smart contract code that represent business logic.
  3. **Querying Mode**:
     + Concerned with probing the contract for data.
     + Executed locally on the nodes for performance reasons.

**Pact Syntax and Features**

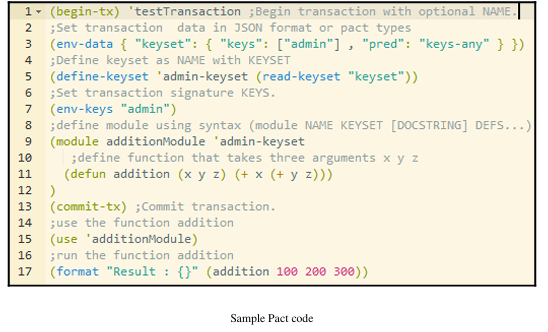
* Pact uses **LISP-like syntax**.
* Represents in the code exactly what will be executed on the blockchain.
  + Stored on the blockchain in **human-readable format**.
  + In contrast, Ethereum's EVM compiles into bytecode for execution, making it difficult to verify what code is being executed.
* Key features:
  + **Turing incomplete**: Improves safety of transaction code execution.
  + **Supports immutable variables**: Ensures stability and predictability in code.
  + **Does not allow null values**: Enhances reliability and reduces errors.

**Limitations of Chapter Coverage**

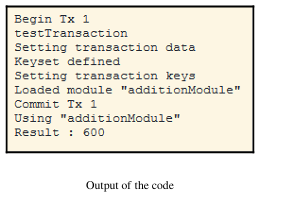
* It is not possible to cover the complete syntax and functions of Pact in this chapter.

**Example: Smart Contract in Pact**

* A small example demonstrates the general structure of a smart contract written in Pact:
  + **Module**: Defines a simple addition function.
  + **Function**: Named addition, takes three parameters, adds their values, and displays the result.
* **Development Tool**:
  + The example was developed using the online Pact compiler available at <http://kadena.io/try-pact/>.



When the code is run, it produces the output shown as follows



**Execution Transparency in Pact**

* As shown in the preceding example, the execution output matches exactly with the code layout and structure.
  + This allows for greater **transparency** and limits the possibility of **malicious code execution**.

**Kadena: A New Class of Private Blockchains**

* Kadena introduces the novel concept of **pervasive determinism**:
  + In addition to standard public/private key-based data origin security, it also provides an additional layer of **fully deterministic consensus**.
* It provides **cryptographic security** at all layers of the blockchain:
  + **Transactions**
  + **Consensus layer**

**Pact Documentation and Source Code**

* Relevant documentation and source code for Pact can be found here:
  + <https://github.com/kadena-io/pact>