2024/12/03 @WIFS 2024

Smart Contract Kill Switch for Security in a Private Blockchain-based Software Transaction System

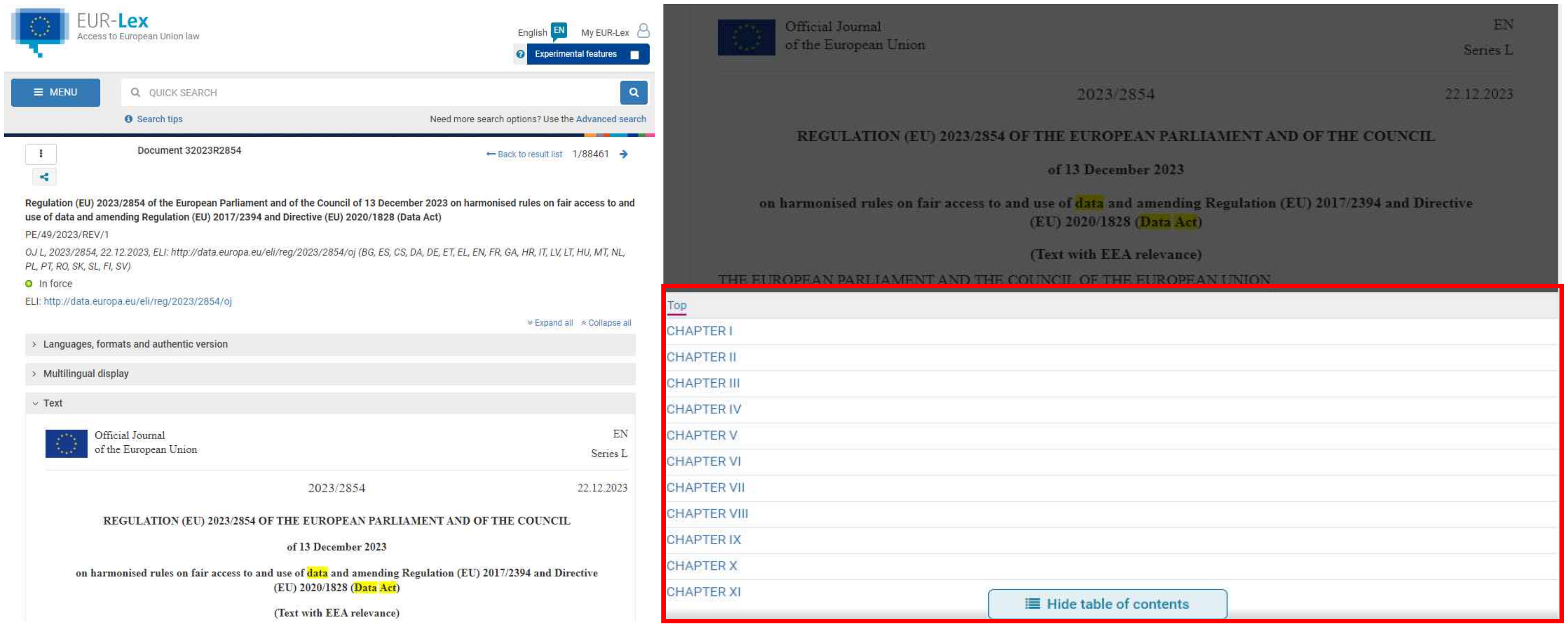
Soonhong Kwon, Wooyoung Son, Jong-Hyouk Lee
({soonhong, wooyoung, jonghyouk}@pel.sejong.ac.kr)
Protocol Engineering Lab., Sejong University

Contents

Overview (1/3)

EU – Enforcement of the Amendment to the Data Act

- In March 2023, the EU Commission passed an amendment to the EU Data Act, which officially takes effect on January 11, 2024
- The amendment allows a transition period of 20 months and is planned to be implemented across all EU member states starting September 2025



Document 32023R2854

Regulation (EU) 2023/2854 of the European Parliament and of the Council of 13 December 2023 on harmonised rules on fair access to and use of data and amending Regulation (EU) 2017/2394 and Directive (EU) 2020/1828 (Data Act)

PE/49/2023/REV/1

OJ L, 2023/2854, 22.12.2023, ELI: http://data.europa.eu/eli/reg/2023/2854/oj (BG, ES, CS, DA, DE, ET, EL, EN, FR, GA, HR, IT, LV, LT, HU, MT, NL, PL, PT, RO, SK, SL, FI, SV)

In force

ELI: http://data.europa.eu/eli/reg/2023/2854/oj

Official Journal of the European Union — EN Series L

2023/2854 22.12.2023

REGULATION (EU) 2023/2854 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL

of 13 December 2023

on harmonised rules on fair access to and use of data and amending Regulation (EU) 2017/2394 and Directive (EU) 2020/1828 (Data Act)

(Text with EEA relevance)

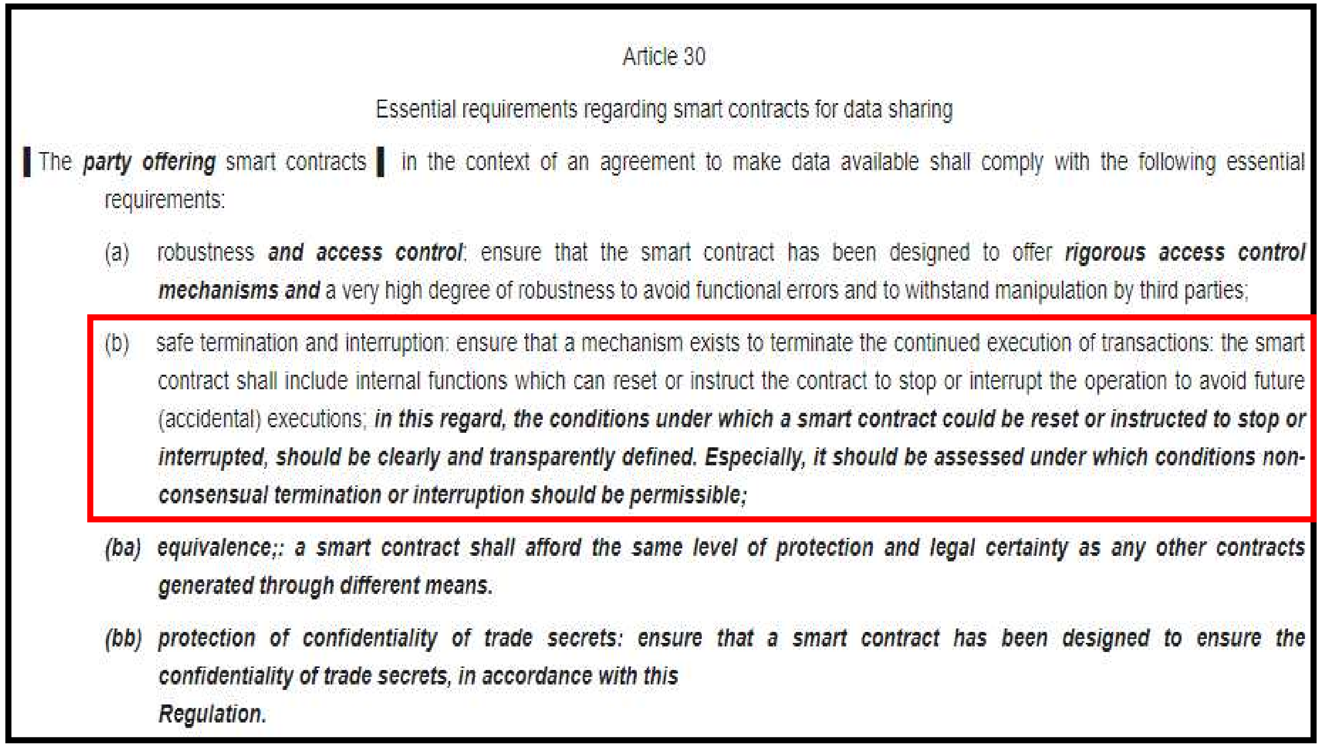
THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION

Figure 1. Overview of the EU Data Act Amendment

Overview (2/3)

Data Act – Issues related to the Smart Contract Kill Switch Provision

- Article 30 of the Data Act specifies the mandatory inclusion of a “kill switch” function in smart contracts
- Within the cryptocurrency community, the requirement for a mandatory “kill switch” in smart contracts has sparked significant concerns and ongoing debates



Article 30

Essential requirements regarding smart contracts for data sharing

The ***party offering*** smart contracts in the context of an agreement to make data available shall comply with the following essential requirements:

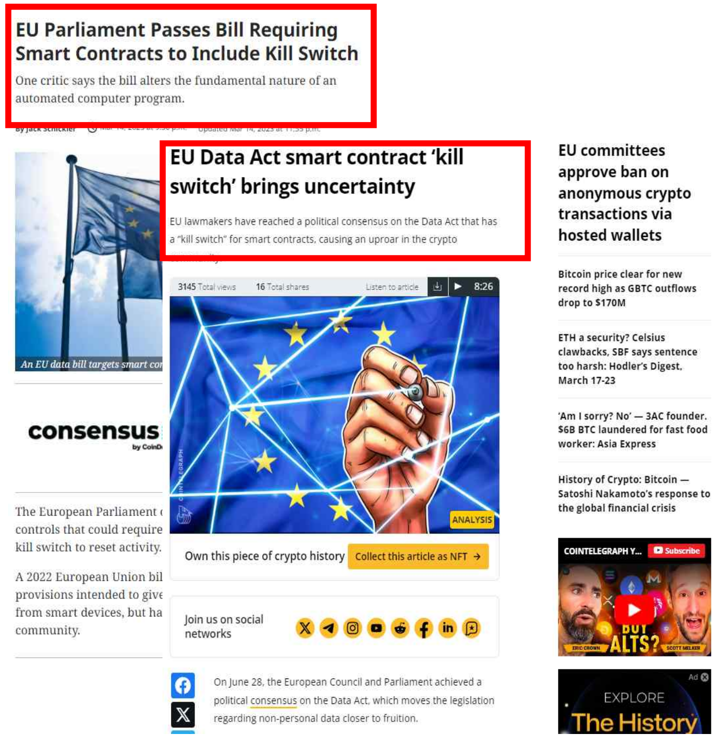
(a) robustness ***and access control***: ensure that the smart contract has been designed to offer ***rigorous access control mechanisms and*** a very high degree of robustness to avoid functional errors and to withstand manipulation by third parties;

(b) safe termination and interruption: ensure that a mechanism exists to terminate the continued execution of transactions: the smart contract shall include internal functions which can reset or instruct the contract to stop or interrupt the operation to avoid future (accidental) executions; ***in this regard, the conditions under which a smart contract could be reset or instructed to stop or interrupted, should be clearly and transparently defined. Especially, it should be assessed under which conditions non-consensual termination or interruption should be permissible;***

(ba) equivalence;: a smart contract shall afford the same level of protection and legal certainty as any other contracts generated through different means.

(bb) protection of confidentiality of trade secrets: ensure that a smart contract has been designed to ensure the confidentiality of trade secrets, in accordance with this Regulation.

Figure 2. EU – Provision on the Mandatory Kill Switch for Smart Contracts in the Data Act Amendment



EU Parliament Passes Bill Requiring Smart Contracts to Include Kill Switch

One critic says the bill alters the fundamental nature of an automated computer program.

An EU data bill targets smart co

The European Parliament
controls that could require
kill switch to reset activity.

A 2022 European Union bil
provisions intended to give
from smart devices, but ha
community.

EU Data Act smart contract ‘kill switch’ brings uncertainty

EU lawmakers have reached a political consensus on the Data Act that has a “kill switch” for smart contracts, causing an uproar in the crypto

On June 28, the European Council and Parliament achieved a political consensus on the Data Act, which moves the legislation regarding non-personal data closer to fruition.

EU committees approve ban on anonymous crypto transactions via hosted wallets

Bitcoin price clear for new record high as GBTC outflows drop to $170M

ETH a security? Celsius clawbacks, SBF says sentence too harsh: Hodler’s Digest, March 17-23

‘Am I sorry? No’ — 3AC founder. $6B BTC laundered for fast food worker: Asia Express

History of Crypto: Bitcoin — Satoshi Nakamoto’s response to the global financial crisis

Figure 3. Issues and Debates surrounding the mandatory kill switch provision for smart contracts

Overview (3/3)

Key Perspectives on the Smart Contract Kill Switch Provision

- By September 2025, when the law is applied across all EU member states, existing deployed smart contracts may risk being deemed illegal; This is expected to have a particularly detrimental impact on DeFi, which is fundamentally rooted in transparency and decentralization
- However, with financial losses continually arising from smart contract security issues, the implementation of a smart contract kill switch offers the advantage of minimizing damage caused by security incidents

Related Work

Software Transaction Revenue Settlement/Distribution Process

1. Service providers such as Apple or Google develop and upload a Software Development Kit (SDK) to the software sales server
2. Developers or development companies use the SDK to develop software and register it on the software sales server
3. When a buyer purchases the software, the software sales server deducts a certain fee and settles and distributes the sales revenue accordingly

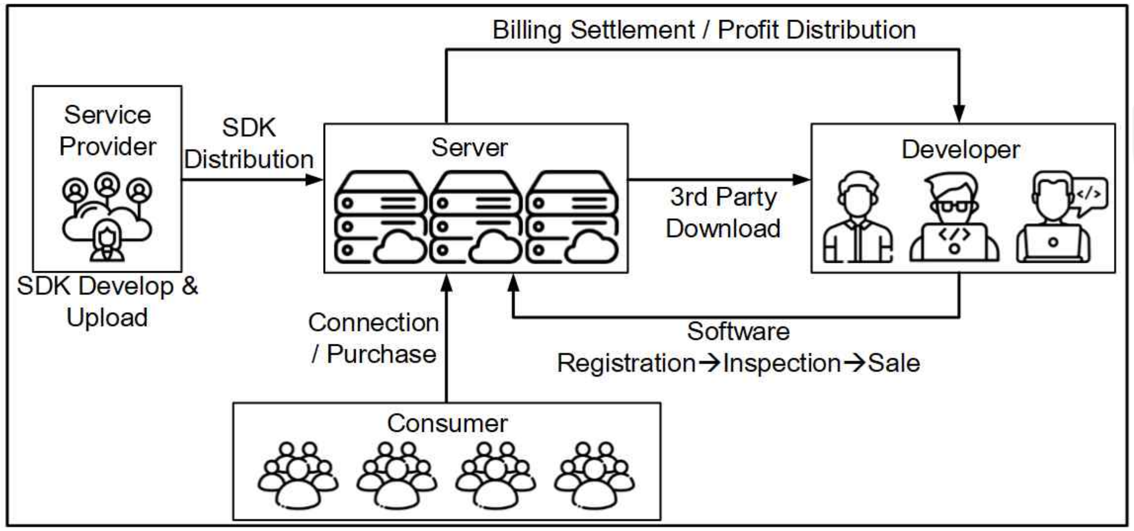


Figure 4. The Existing Process for Settling and Distributing Software Transaction Revenue

Table 1. Limitations of the Software Transaction Revenue Settlement and Distribution Process

Limitation	Description
Centralized Method for Revenue Settlement and Distribution	• Developers or development companies cannot transparently verify the revenue settlement and distribution efforts for their software due to centralized sales servers
Issues with Unclear Revenue Settlement and Distribution	• Development companies may provide incentives to developers from the additional revenue generated by the software, but developers cannot verify the evidence of settlement and distribution

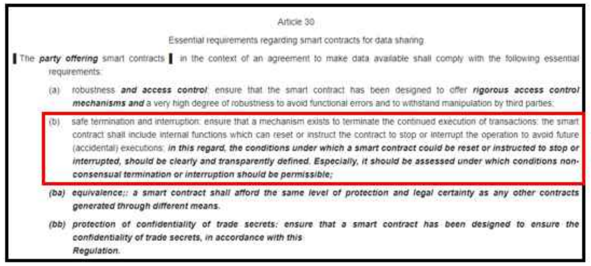
Proposed Framework (1/8)

Motivation

- The ongoing disputes and issues surrounding the smart contract kill switch stem from the lack of clear guidelines and the undefined scope of its application in the legislation

AS – IS

Lack of Scope and Clear Guidelines for Smart Contract Kill Switch Implementation



Article 30

Essential requirements regarding smart contracts for data sharing

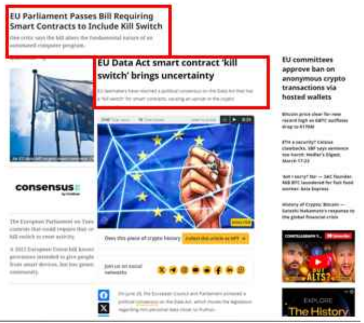
▌The ***party offering*** smart contracts ▌ in the context of an agreement to make data available shall comply with the following essential requirements:

(a) robustness ***and access control***: ensure that the smart contract has been designed to offer ***rigorous access control mechanisms and*** a very high degree of robustness to avoid functional errors and to withstand manipulation by third parties;

(b) safe termination and interruption: ensure that a mechanism exists to terminate the continued execution of transactions: the smart contract shall include internal functions which can reset or instruct the contract to stop or interrupt the operation to avoid future (accidental) executions; ***in this regard, the conditions under which a smart contract could be reset or instructed to stop or interrupted, should be clearly and transparently defined. Especially, it should be assessed under which conditions non-consensual termination or interruption should be permissible;***

(ba) equivalence;: a smart contract shall afford the same level of protection and legal certainty as any other contracts generated through different means.

(bb) protection of confidentiality of trade secrets: ensure that a smart contract has been designed to ensure the confidentiality of trade secrets, in accordance with this Regulation.



EU Parliament Passes Bill Requiring Smart Contracts to Include Kill Switch

EU Data Act smart contract 'kill switch' brings uncertainty

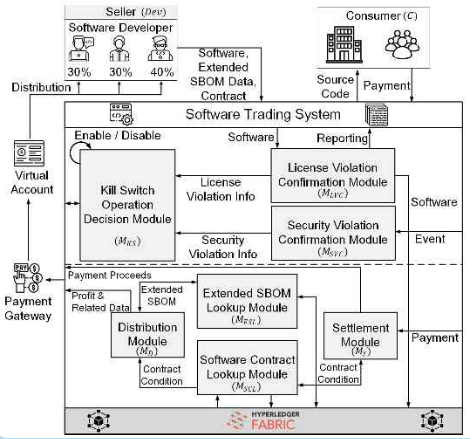
EU committees approve ban on anonymous crypto transactions via hosted wallets

Concerns Over Existing Deployed Smart Contracts Being Deemed Illegal Due to Mandatory Smart Contract Kill Switch Enforcement

Lack of Scope and Guidelines for Smart Contract Kill Switch Implementation in the Data Act Amendment

TO – BE

Proposed System



Providing Buyers with Software Deliverables Free from License Issues

Settlement and Distribution of Revenue Based on Developer Contributions Evaluated by Peer Review

Kill Switch Functionality to Address Security and License Violation Issues

Activation of the Smart Contract Kill Switch Based on Stakeholder Consensus

Preventing Unfair Revenue Settlement and Distribution Issues Caused by the Closed Structure of Traditional Software Transaction Systems

Minimizing Financial Losses in the Event of Security Incidents or License Violations

Proposed Framework (2/8)

Overview

- In response to the ongoing issues caused by centralized software sales servers, a decentralized and transparent software transaction system is proposed
- To address potential license issues and security threats arising from smart contract vulnerabilities in the software transaction system, a smart contract kill switch is implemented

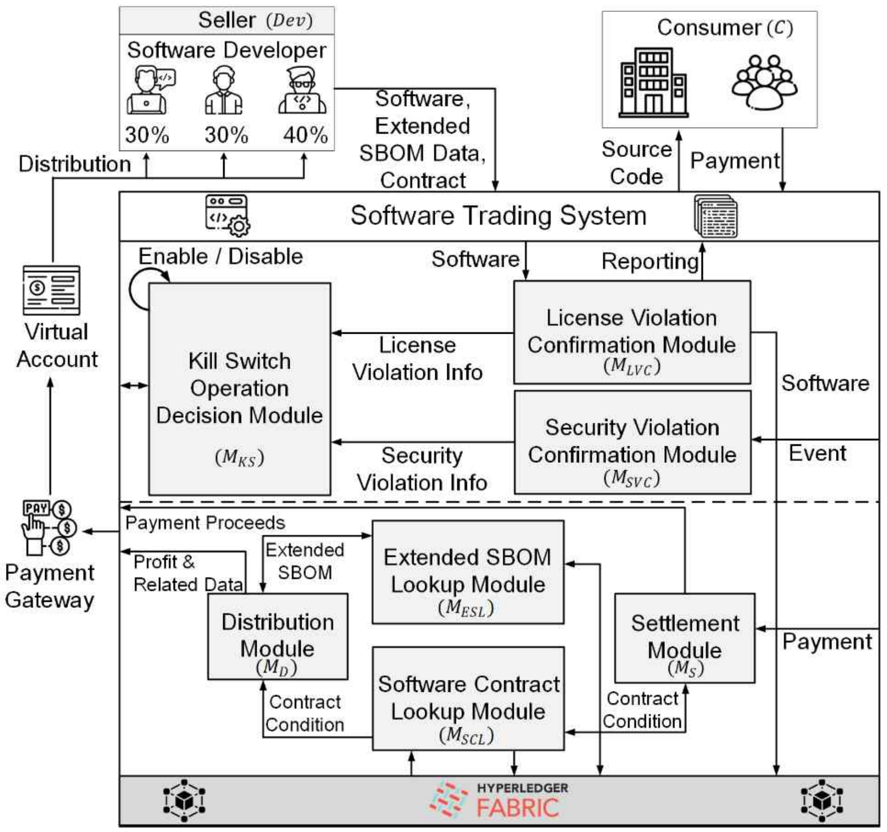
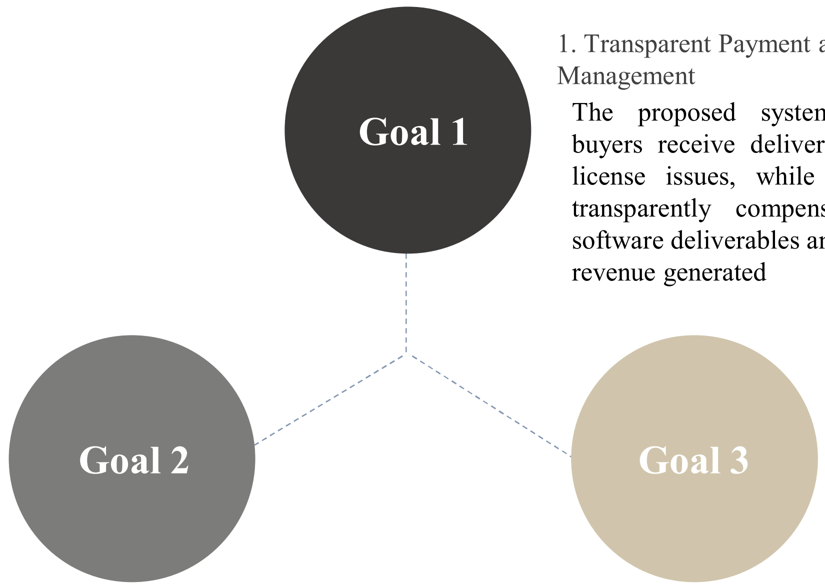


Figure 5. Proposed System



1. Transparent Payment and License Management

The proposed system ensures that buyers receive deliverables free from license issues, while developers are transparently compensated for their software deliverables and any additional revenue generated

2. Addressing Security and License-Related Issues

In the proposed system, if security issues or license violations occur during operation, the smart contract kill switch functionality is utilized to effectively respond to such incidents

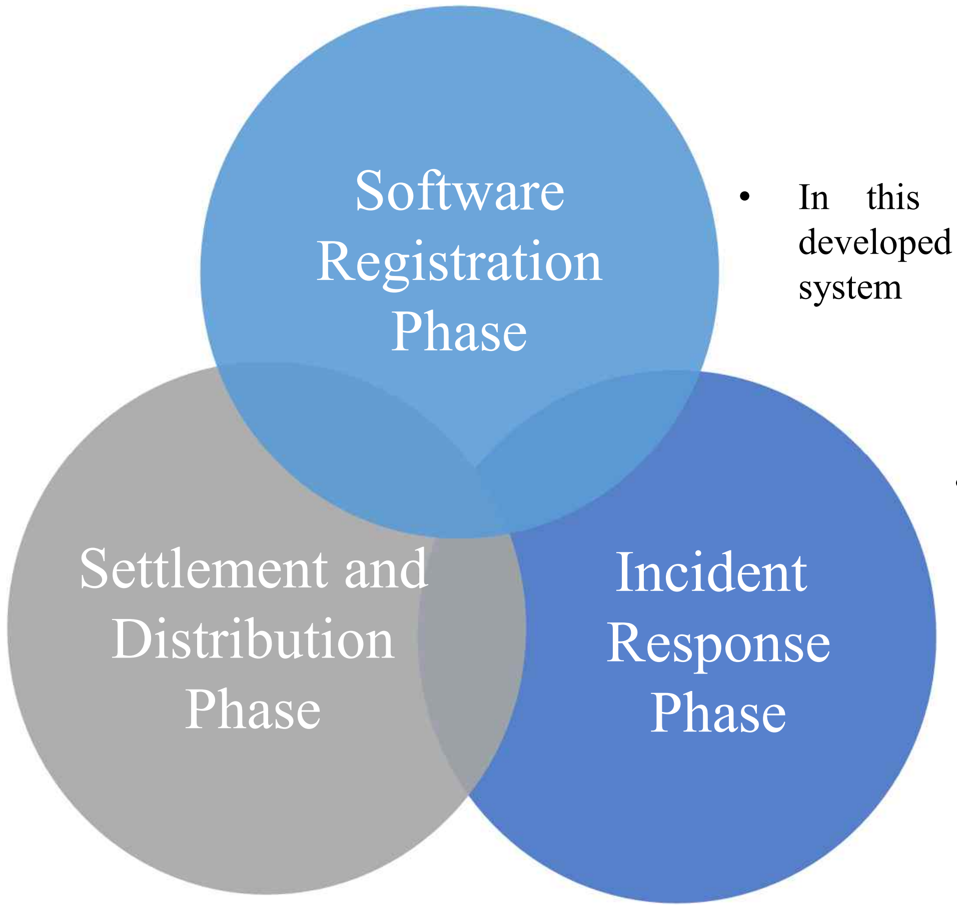
3. Stakeholder Consensus-Based Kill Switch Operation

The proposed system ensures that the activation of the smart contract kill switch is processed based on the consensus of stakeholders involved in the software transaction

Proposed Framework (3/8)

Key Phases

- The proposed system operates in three phases: ‘Software Registration Stage’, ‘Settlement and Distribution Stage’, and ‘Incident Response Stage’



Software Registration Phase
Settlement and Distribution Phase
Incident Response Phase

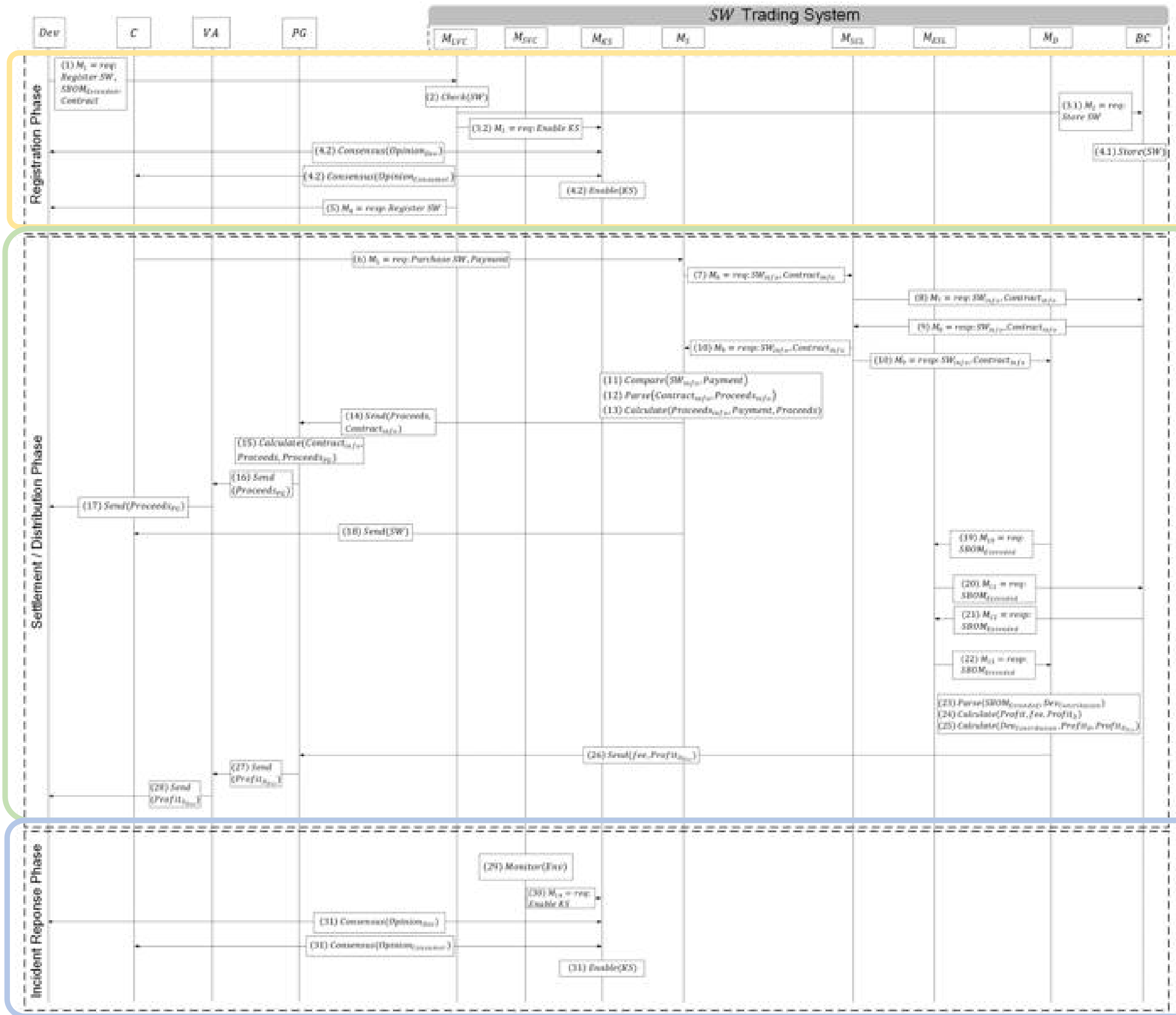
- In this phase, developers register their developed software in the software transaction system

- Clients pay for software that meets their needs, and the software transaction system performs settlement and distribution based on the transaction information

- If a security issue or license violation arises within the software transaction system, the smart contract kill switch is activated through consensus to address the issue immediately

Proposed Framework (4/8)

- Detailed Operational Process


SW Trading System
Dev
C
VA
PG
M_{LFC}
M_{SVC}
M_{KS}
M_T
M_{SCL}
M_{EVL}
M_D
BC
Registration Phase
Settlement / Distribution Phase
Incident Reponse Phase
(1) M_1 = req: Register SW, $SBOM_{Extended}$, Contract
(2) Check(SW)
(3.1) M_2 = req: Store SW
(3.2) M_3 = req: Enable KS
(4.1) Store(SW)
(4.2) Consensus(Opinion$_{Dev}$)
(4.2) Consensus(Opinion$_{Consumer}$)
(4.2) Enable(KS)
(5) M_4 = resp: Register SW
(6) M_5 = req: Purchase SW, Payment
(7) M_6 = req: SW_{info}, $Contract_{info}$
(8) M_7 = req: SW_{info}, $Contract_{info}$
(9) M_8 = resp: SW_{info}, $Contract_{info}$
(10) M_9 = resp: SW_{info}, $Contract_{info}$
(10) M_9 = resp: SW_{info}, $Contract_{info}$
(11) Compare(SW_{info}, Payment)
(12) Parse($Contract_{info}$, $Proceeds_{info}$)
(13) Calculate($Proceeds_{info}$, Payment, Proceeds)
(14) Send(Proceeds, $Contract_{info}$)
(15) Calculate($Contract_{info}$, Proceeds, $Proceeds_{PG}$)
(16) Send($Proceeds_{PG}$)
(17) Send($Proceeds_{PG}$)
(18) Send(SW)
(19) M_{10} = req: $SBOM_{Extended}$
(20) M_{11} = req: $SBOM_{Extended}$
(21) M_{12} = resp: $SBOM_{Extended}$
(22) M_{13} = resp: $SBOM_{Extended}$
(23) Parse($SBOM_{Extended}$, $Dev_{Contribution}$)
(24) Calculate(Profit, fee, $Profit_D$)
(25) Calculate($Dev_{Contribution}$, $Profit_D$, $Profit_{Dev}$)
(26) Send(fee, $Profit_{Dev}$)
(27) Send($Profit_{Dev}$)
(28) Send($Profit_{Dev}$)
(29) Monitor(Env)
(30) M_{14} = req: Enable KS
(31) Consensus(Opinion$_{Dev}$)
(31) Consensus(Opinion$_{Consumer}$)
(31) Enable(KS)

Phase 1: SW Registration

- The developer registers the software they wish to distribute, along with the Software Bill of Materials (SBOM), in the software transaction system

Phase 2: Settlement/Distribution

- This phase involves the settlement and distribution of sales revenue for the software developed by the developer, as well as the settlement and distribution of any additional revenue generated from the sold software

1. Developer's Contribution to SW

$$\lambda_i = 1 + \left(\sum_{j=1}^{n} t_j\right)$$

※ t_j: Work performed by the developer per task

$$t_j = \frac{pd}{\sum_{i=1}^{n} ad_j}$$

(ad: Deadline for the task, pd≒ Actual Completion of the task)

2. Adjustment of Evaluation Scores based on Significance Levels

$$\tau_j = \frac{a_j}{\sum_{j=1}^{n} a_j}$$

※ a_j: Significance weight for evaluator j

$$a_j = \Pr(P^2(n-1) \geq S_j^2$$

S_j^2: Score assigned to developer by evaluator;

$$S_j^2 = \sum_{i=1}^{n} \frac{\left(P_{(i,j)} - \bar{P}_{(i,j)}\right)^2}{\bar{P}_{(i,j)}}$$

3. Fairness-Validated Peer Review Score

$$CP_i = \sum_{i=1}^{n} \tau_j P_{(i,j)}$$

4. Developer Contribution Score Calculation

$$C_i = CP_i \lambda_i$$

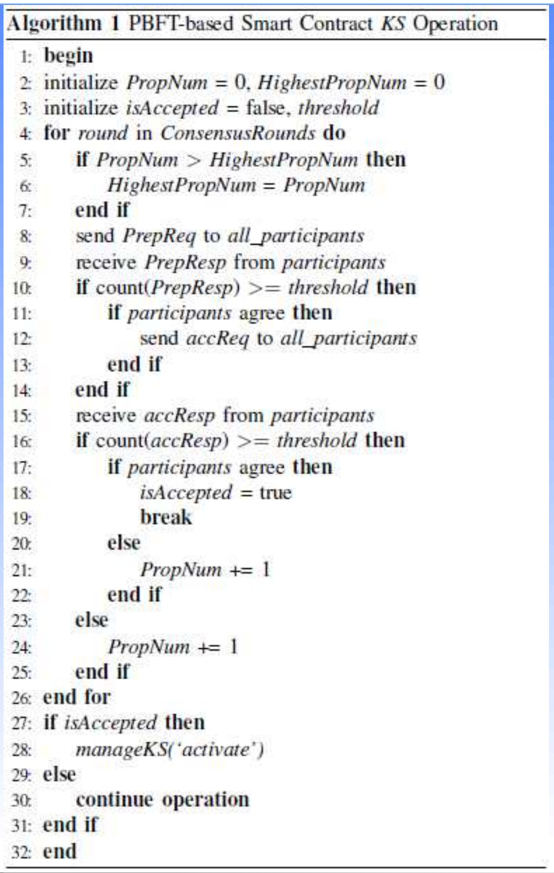
Phase 3: Incident Response Phase

- A stage where the system responds to abnormal behavior detected during the monitoring of the software transaction system

Proposed Framework (5/8)

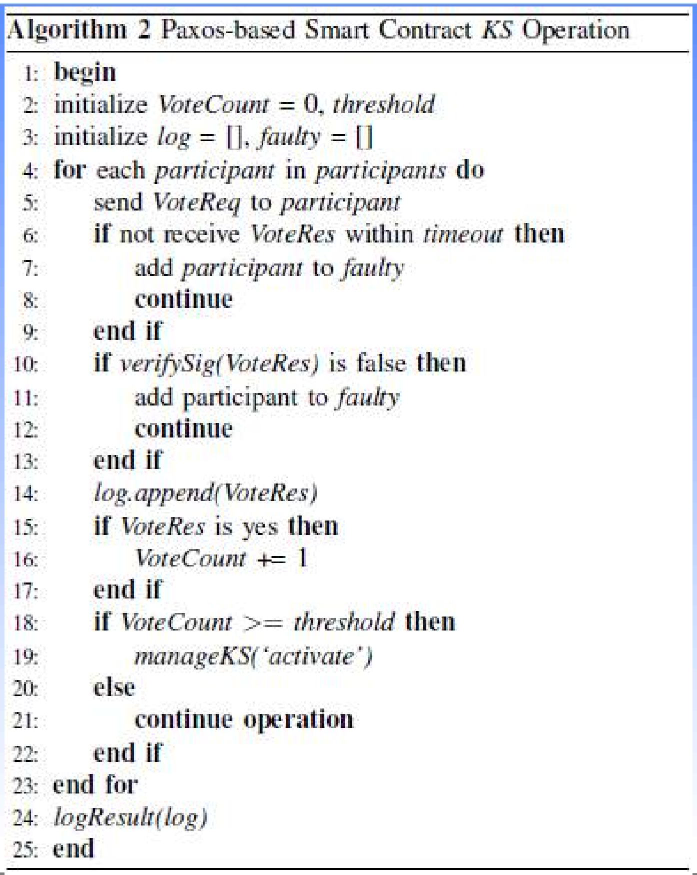
Smart Contract Kill Switch Activation Decision Algorithm

- If the activation of a smart contract kill switch is carried out by a third party, it could compromise the decentralization of the blockchain
- A solution for activating the smart contract kill switch based on a consensus algorithm is proposed



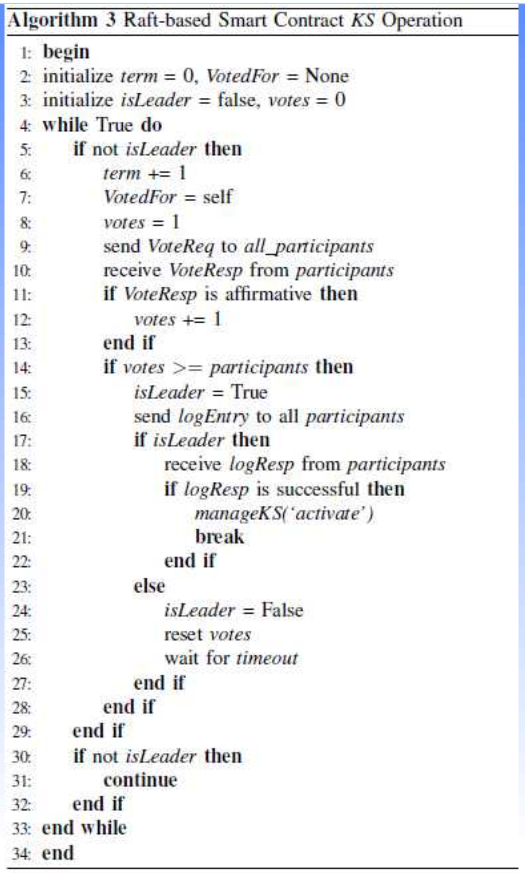
Algorithm 1 PBFT-based Smart Contract *KS* Operation

```
begin
initialize PropNum = 0, HighestPropNum = 0
initialize isAccepted = false, threshold
for round in ConsensusRounds do
    if PropNum > HighestPropNum then
        HighestPropNum = PropNum
    end if
    send PrepReq to all_participants
    receive PrepResp from participants
    if count(PrepResp) >= threshold then
        if participants agree then
            send accReq to all_participants
        end if
    end if
    receive accResp from participants
    if count(accResp) >= threshold then
        if participants agree then
            isAccepted = true
            break
        else
            PropNum += 1
        end if
    else
        PropNum += 1
    end if
end for
if isAccepted then
    manageKS('activate')
else
    continue operation
end if
end
```



Algorithm 2 Paxos-based Smart Contract *KS* Operation

```
begin
initialize VoteCount = 0, threshold
initialize log = [], faulty = []
for each participant in participants do
    send VoteReq to participant
    if not receive VoteRes within timeout then
        add participant to faulty
        continue
    end if
    if verifySig(VoteRes) is false then
        add participant to faulty
        continue
    end if
    log.append(VoteRes)
    if VoteRes is yes then
        VoteCount += 1
    end if
    if VoteCount >= threshold then
        manageKS('activate')
    else
        continue operation
    end if
end for
logResult(log)
end
```



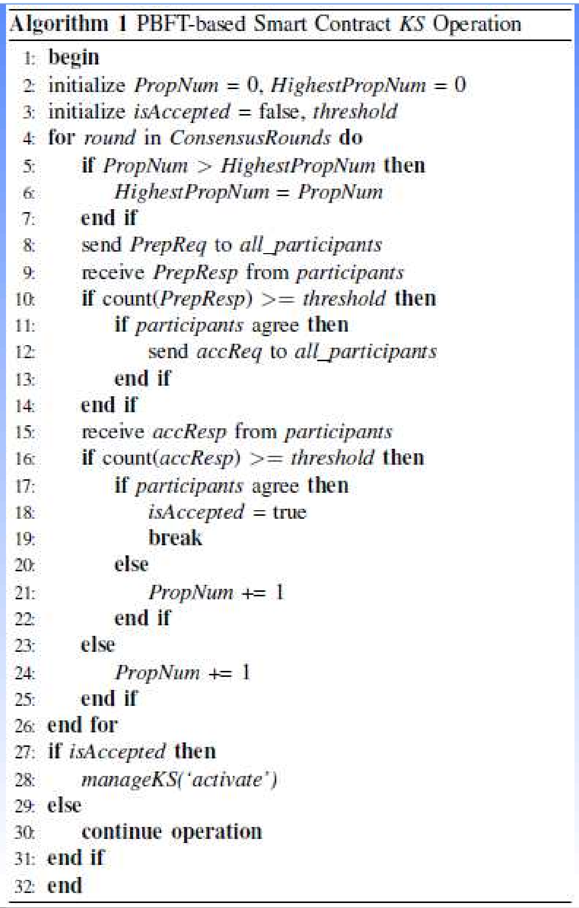
Algorithm 3 Raft-based Smart Contract *KS* Operation

```
begin
initialize term = 0, VotedFor = None
initialize isLeader = false, votes = 0
while True do
    if not isLeader then
        term += 1
        VotedFor = self
        votes = 1
        send VoteReq to all_participants
        receive VoteResp from participants
        if VoteResp is affirmative then
            votes += 1
        end if
        if votes >= participants then
            isLeader = True
            send logEntry to all participants
            if isLeader then
                receive logResp from participants
                if logResp is successful then
                    manageKS('activate')
                    break
                end if
            else
                isLeader = False
                reset votes
                wait for timeout
            end if
        end if
    end if
    if not isLeader then
        continue
    end if
end while
end
```

Proposed Framework (6/8)

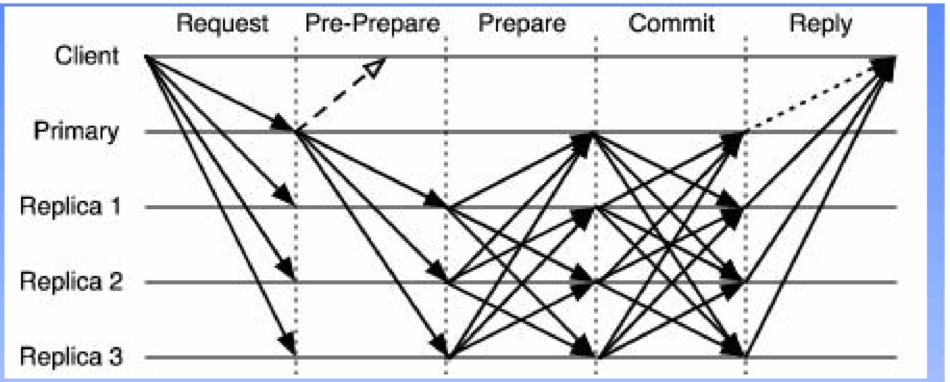
PBFT-based Smart Contract Kill Switch Activation Decision Algorithm

- A Byzantine Fault Tolerant (BFT) consensus algorithm designed to address the Byzantine problem, where 51% of the nodes are malicious
  - With a total of N participants, the system can function without issues in an environment where F participants are faulty, as long as N=3F+1



Algorithm 1 PBFT-based Smart Contract *KS* Operation

```
begin
initialize PropNum = 0, HighestPropNum = 0
initialize isAccepted = false, threshold
for round in ConsensusRounds do
    if PropNum > HighestPropNum then
        HighestPropNum = PropNum
    end if
    send PrepReq to all_participants
    receive PrepResp from participants
    if count(PrepResp) >= threshold then
        if participants agree then
            send accReq to all_participants
        end if
    end if
    receive accResp from participants
    if count(accResp) >= threshold then
        if participants agree then
            isAccepted = true
            break
        else
            PropNum += 1
        end if
    else
        PropNum += 1
    end if
end for
if isAccepted then
    manageKS('activate')
else
    continue operation
end if
end
```



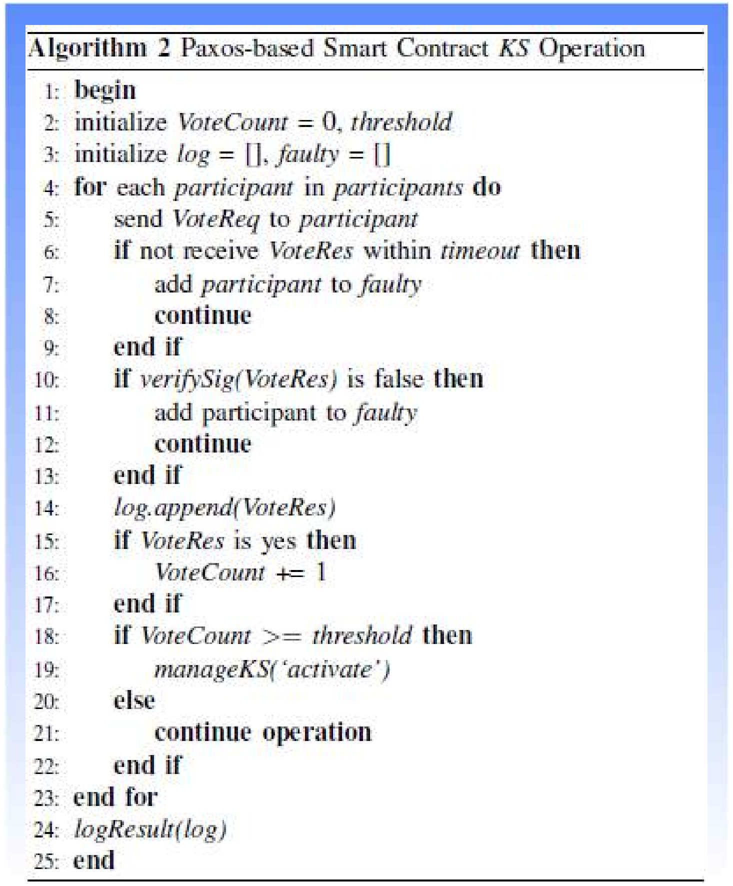
1. Initialize voteCount to calculate the number of agreement votes, consensusThreshold as the minimum number of agreement votes required for consensus, a messageLog array to store vote responses, and a faultyParticipants array to record faulty participants
2. Send signed vote requests to all participants and add verified responses to the messageLog
3. If the voteCount for responses agreeing to activate the kill switch meets or exceeds the consensusThreshold, activate the KS. If not, maintain the existing system operation

※ Note: If responses agreeing to activate the KS are not received within the timeout period, the corresponding participant is added to faultyParticipants. Similarly, participants providing invalidly signed vote responses are also added to faultyParticipants

Proposed Framework (7/8)

Paxos-based Smart Contract Kill Switch Activation Decision Algorithm

- A protocol for reaching consensus on a single value among multiple processes in a distributed system
  - Multiple values may be proposed simultaneously, but only one value is chosen



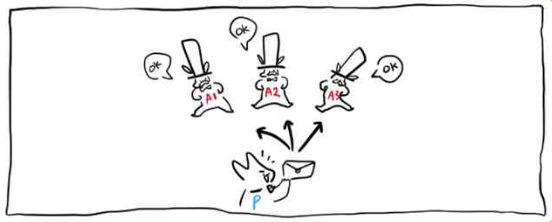
Algorithm 2 Paxos-based Smart Contract *KS* Operation

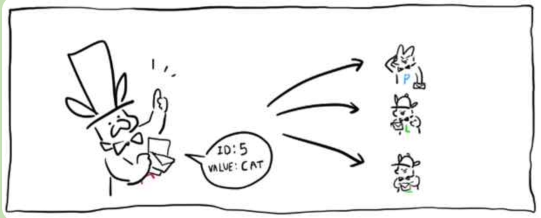
```
begin
initialize VoteCount = 0, threshold
initialize log = [], faulty = []
for each participant in participants do
    send VoteReq to participant
    if not receive VoteRes within timeout then
        add participant to faulty
        continue
    end if
    if verifySig(VoteRes) is false then
        add participant to faulty
        continue
    end if
    log.append(VoteRes)
    if VoteRes is yes then
        VoteCount += 1
    end if
    if VoteCount >= threshold then
        manageKS('activate')
    else
        continue operation
    end if
end for
logResult(log)
end
```

5
P
ID:5
OK
AAA

- **(PREPARE)** The Proposer sends a proposal number (ID) to the Accepters to propose agreement on whether to activate the kill switch
- **(PROMISE)** The Accepter promises not to accept any values lower than the proposed number

- **(ACCEPT)** If a majority of Accepters send a PROMISE message with the same ID to the Proposer, the Proposer sends the VALUE associated with that ID to the Accepters.



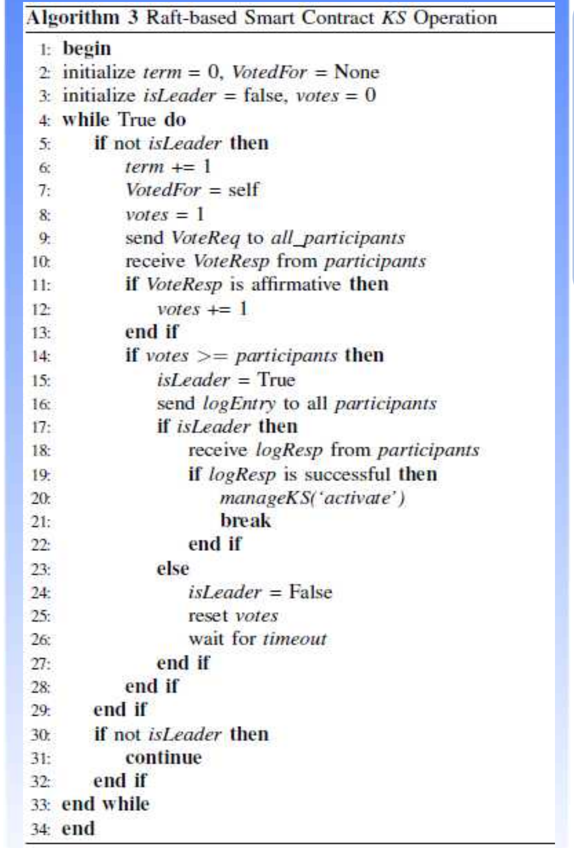


- **(ACCEPTED)** The Accepter accepts the VALUE only if the ID matches the last promised value, and then propagates the VALUE to both the Proposer and the Learner, completing the consensus

Proposed Framework (8/8)

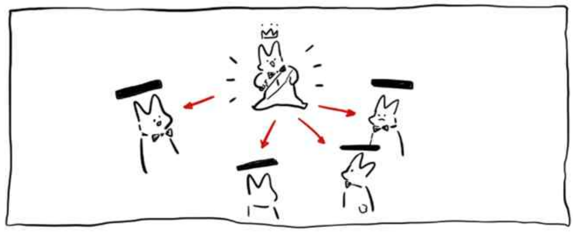
Raft-based Smart Contract Kill Switch Activation Decision Algorithm

- An algorithm designed to ensure that all nodes in a distributed system maintain the same state and that the entire system continues to operate seamlessly even if some nodes experience failures



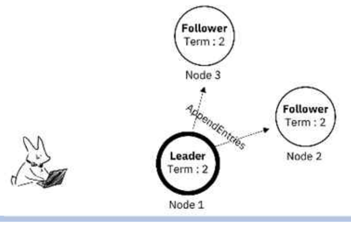
Algorithm 3 Raft-based Smart Contract *KS* Operation

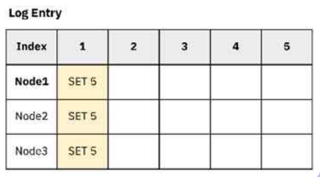
```
begin
initialize $term$ = 0, $VotedFor$ = None
initialize $isLeader$ = false, $votes$ = 0
while True do
    if not $isLeader$ then
        $term$ += 1
        $VotedFor$ = self
        $votes$ = 1
        send $VoteReq$ to $all\_participants$
        receive $VoteResp$ from $participants$
        if $VoteResp$ is affirmative then
            $votes$ += 1
        end if
        if $votes$ >= $participants$ then
            $isLeader$ = True
            send $logEntry$ to all $participants$
            if $isLeader$ then
                receive $logResp$ from $participants$
                if $logResp$ is successful then
                    $manageKS('activate')$
                    break
                end if
            else
                $isLeader$ = False
                reset $votes$
                wait for $timeout$
            end if
        end if
    end if
    if not $isLeader$ then
        continue
    end if
end while
end
```



- The Leader periodically sends empty AppendEntries RPC messages to Followers to signal that it is alive
- If a Follower does not receive a signal from the Leader within a specified timeout period (150ms–300ms), it nominates itself to become the Leader
- The Follower transitions to a Candidate, votes for itself, and sends vote requests to other Followers

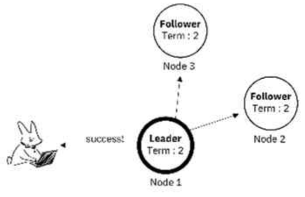
1. The Client sends changes to the Leader
2. The changes are stored in the Leader's log entries
3. The Leader calls the AppendEntries RPC to replicate the log to the Followers
4. The Followers save the newly received log entries and send a success response

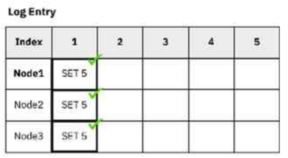




Log Entry

Index	1	2	3	4	5
Node1	SET 5				
Node2	SET 5				
Node3	SET 5				





Log Entry

Index	1	2	3	4	5
Node1	SET 5 ✓				
Node2	SET 5 ✓				
Node3	SET 5 ✓				

- When the Leader receives responses from a majority of Followers, it commits its log entry and sends a response to the Client
- The Leader also notifies the Followers that the changes have been committed

Analysis (1/2)

Evaluation of the Suitability of the Smart Contract Kill Switch Activation Decision Algorithm

- Determining the suitability of each consensus algorithm based on the number of nodes performing the leader role for each algorithm and the stability indicators of the consensus algorithms

Table 2. Performance per consensus algorithm

Item	Consensus Algorithm		
	PBFT	Paxos	Raft
Number of Leader Nodes	$\frac{2N+1}{3}$	$\frac{N+1}{2}$	$\frac{N+1}{2}$
Stability	$\beta * N\ time + TPS$	$\beta * \left(\frac{N}{2}\right) + 1\ time + TPS$	$\beta * \left(\frac{N}{2}\right) + 1\ time + TPS$
Activation Time	0.405	0.623	0.619
Recovery Time	0.064	0.090	0.120
Operation Accuracy	100%	100%	100%

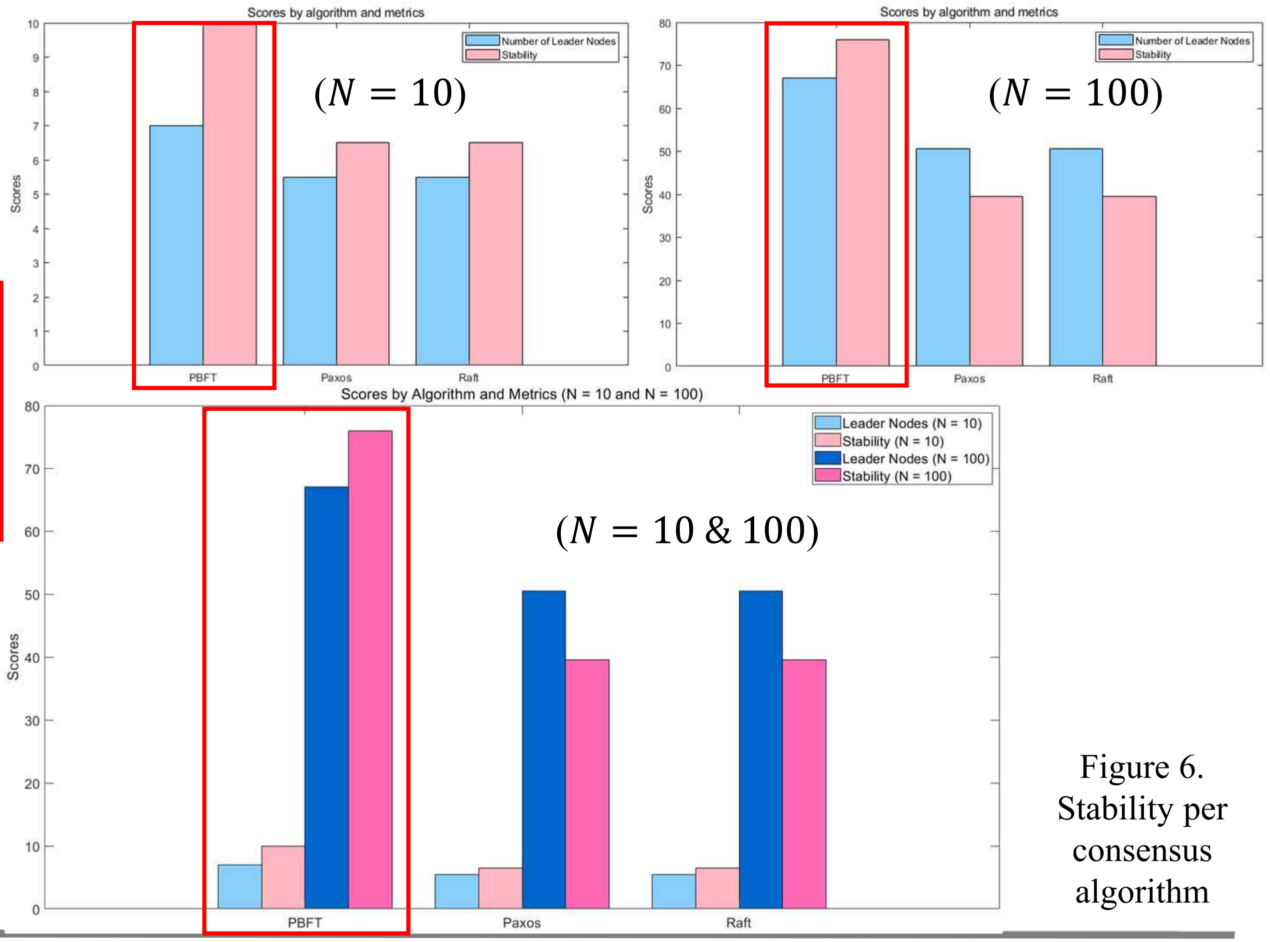


Figure 6. Stability per consensus algorithm

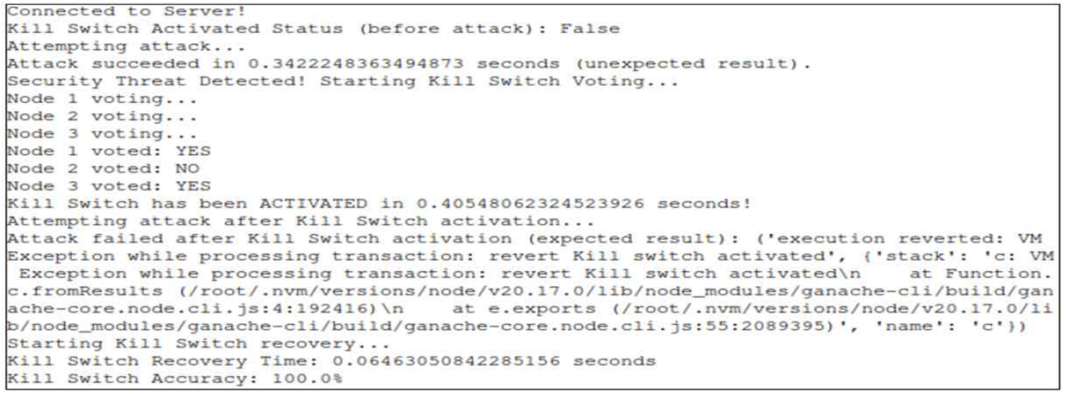
Analysis (2/2)

Evaluation of the Suitability of the Smart Contract Kill Switch Activation Decision Algorithm

- To evaluate the suitability of the smart contract kill switch activation decision algorithm, the activation time, recovery time, and operational accuracy of the smart contract kill switch are assessed

Table 2. Performance per consensus algorithm

Item	Consensus Algorithm		
	PBFT	Paxos	Raft
Number of Leader Nodes	$\frac{2N+1}{3}$	$\frac{N+1}{2}$	$\frac{N+1}{2}$
Stability	$\beta * N\ time + TPS$	$\beta * \left(\frac{N}{2}\right) + 1\ time + TPS$	$\beta * \left(\frac{N}{2}\right) + 1\ time + TPS$
Activation Time	0.405	0.623	0.619
Recovery Time	0.064	0.090	0.120
Operation Accuracy	100%	100%	100%



```
Connected to Server!
Kill Switch Activated Status (before attack): False
Attempting attack...
Attack succeeded in 0.3422248363494873 seconds (unexpected result).
Security Threat Detected! Starting Kill Switch Voting...
Node 1 voting...
Node 2 voting...
Node 3 voting...
Node 1 voted: YES
Node 2 voted: NO
Node 3 voted: YES
Kill Switch has been ACTIVATED in 0.40548062324523926 seconds!
Attempting attack after Kill Switch activation...
Attack failed after Kill Switch activation (expected result): ('execution reverted: VM
Exception while processing transaction: revert Kill switch activated', {'stack': 'c: VM
 Exception while processing transaction: revert Kill switch activated\n    at Function.
c.fromResults (/root/.nvm/versions/node/v20.17.0/lib/node_modules/ganache-cli/build/gan
ache-core.node.cli.js:4:192416)\n    at e.exports (/root/.nvm/versions/node/v20.17.0/li
b/node_modules/ganache-cli/build/ganache-core.node.cli.js:55:2089395)', 'name': 'c'})
Starting Kill Switch recovery...
Kill Switch Recovery Time: 0.06463050842285156 seconds
Kill Switch Accuracy: 100.0%
```

Figure 7. PBFT-based Smart Contract Kill Switch Operation

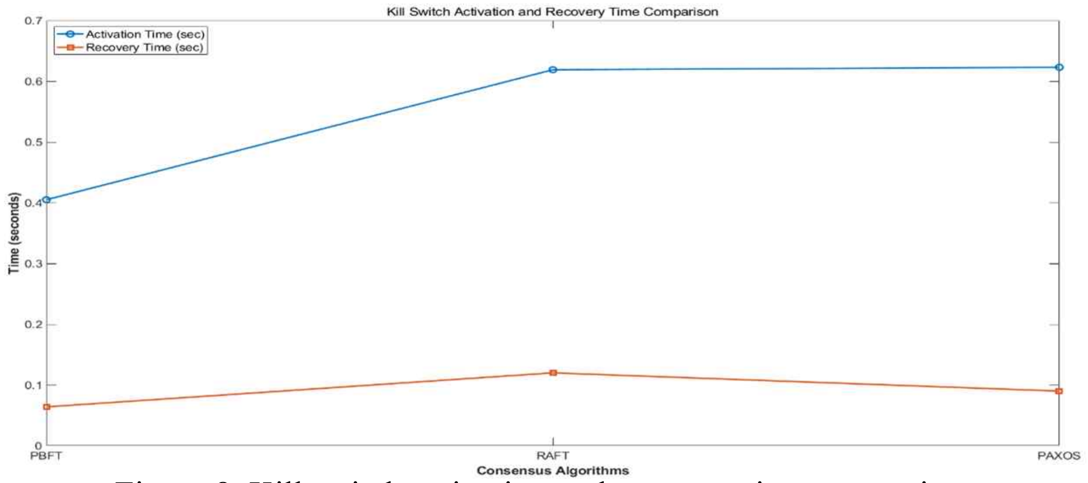


Figure 8. Kill switch activation and recovery time comparison

Conclusion

- Key Contributions
  - Trust Assurance: Ensures buyers receive software free from licensing issues
  - Transparent Settlement: Provides fair payment and additional revenue distribution to developers
  - Risk Minimization: Prevents financial loss through the smart contract kill switch mechanism
  - EU Data Act Compliance: Addresses regulatory requirements and ensures cross-national scalability
- Future Research
  - Multi-Chain Scalability: Analyze the suitability of the smart contract kill switch in multi-blockchain environments

Thanks!

Soonhong Kwon (soonhong@pel.sejong.ac.kr)