



# Rensselaer

why not change the world?®

## The Feasibility of a Smart Contract “Kill Switch”

**Oshani Seneviratne** (*senevo@rpi.edu*)

*Assistant Professor in Computer Science*

**Rensselaer Polytechnic Institute**

# Is the EU Trying to Kill Smart Contracts?

By Leo Ji

## The EU's "Kill Switch": What Does It Mean For The Future Of Decentralization?

The  
that  
NFT:



The EU's "Kill Switch"  
What Does It Mean  
For The Future Of  
Decentralization?

Blockchain Council

September 17,

Legal and IP · News · Technology Media Telecoms

### EU Data Act requires smart contracts to have kill switch, not be permissionless

March 14, 2023 · by Larissa Inciarte

## Blockchain devs expect complications from EU smart contract kill switch

The EU's Data Act could introduce "kill switch" requirements for smart contracts, raising questions about how decentralized projects would handle such a scenario.

3506 Total views

13 Total shares

Listen to article



7:41



# The need for “kill switches”

How many of you trust your phone never to crash or that your favorite app always works perfectly?

Now, imagine trusting a million-dollar financial contract built the same way.

This **trust gap** is exactly what regulators are trying to address with the idea of a ‘kill switch.’



# Introduction

Smart Contracts are autonomous and self-executing code, enabling trustless transactions.

## Key Challenges (for Regulators):

- Addressing consumer protection, privacy, and financial stability concerns.
  - For example, **23327 vulnerable contracts** on Ethereum were reported (Perez et al. , 2021).
- **Consumer protection:** How can users be protected from errors or malicious contracts?
  - To freeze transactions in light of suspicious activities or security breaches.
- **Privacy:** Ensuring compliance with global data privacy laws while maintaining transparency.
  - To protect privacy by terminating contracts in case of data breaches in compliance with regulations like HIPAA.
- **Financial stability:** Avoiding large-scale vulnerabilities that could destabilize financial ecosystems.

# Regulation

**EU Data Act (Article 30):** Outlines **four key requirements** for smart contracts to ensure they remain reliable and compliant to be implemented by platform providers:

1. **Robustness:** Contracts must be resilient against errors and malicious attacks.
2. **Safe Termination & Interruption:** Mechanisms must allow contracts to be halted securely when necessary.
3. **Data Archiving & Continuity:** Data, logic, and code should be archived for **auditability** and continuity.
4. **Access Control:** Strong mechanisms must prevent unauthorized access and ensure data integrity.

## Tension with Blockchain Principles:

- Conflict with **decentralization** and **immutability**.

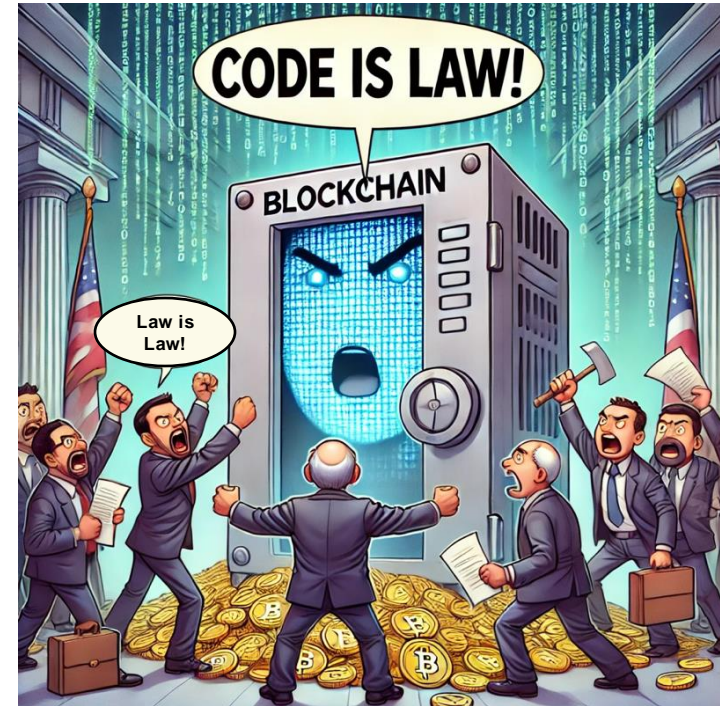


Image credit: OpenAI

# Related Work (1)

---

The role of smart contracts under the EU Data Act focuses on challenges like interoperability, robustness, and termination.

- **Casolari et al; 2023**
  - Examine the role of smart contracts in the architecture of the EU's Data Act, identifying key challenges and proposing recommendations to address those issues.
  - Gaps: Specific mechanisms for smart contract termination in various blockchains are lacking.
- **Olivieri and Pasetto; 2023**
  - Analysis of EU Data Act requirements for smart contracts, focusing on interoperability, robustness, and safe termination.
  - Gaps: Specific mechanisms for smart contract termination in various blockchains are lacking.

## Related Work (2)

---

### Methods for proving smart contract termination:

- **Le et al; 2018.**
  - Method for proving conditional termination of smart contracts utilizing the F\* programming language.
  - Gaps: Limitations in automatically inferring termination proofs for complex programs, necessitating manual intervention for complex cases.
- **Genet et al; 2020.**
  - Formal and mechanized proof of termination based on measures of EVM call stacks for intrinsic system-wide safeguards (gas and call stack limits).
  - Gaps: Comparative analysis of termination mechanisms across different blockchains is lacking.

## Related Work (3)

---

### Solutions for handling unexpected situations in smart contracts:

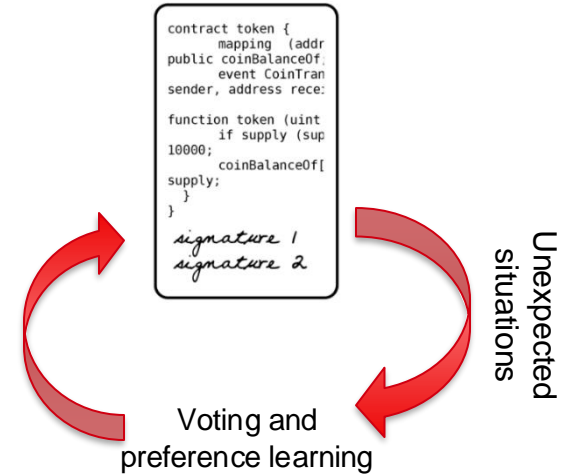
- **Liu et al., 2019.**
  - Handling unexpected situations in smart contracts after a voting round has proven to be resilient in the face of adversarial collusion.
  - Gaps: The generalizability of the proposed method to other smart contract termination scenarios is questionable, and the sandbox environment for voting may not be practical
- **Mohsin et al., 2019.**
  - Utilizing community-accepted off-chain ontologies as a guiding framework for action in case of anomalies or errors in deployed contracts.
  - Gaps: Ontology as a decision-support mechanism requires strong governance and trust guarantees.



# Handling Unexpected Situations in Smart Contracts with Computational Social Choice Mechanisms

**Strengthening Smart Contracts to Handle Unexpected Situations;** *Shuze Liu, Farhad Mohsin, Lirong Xia, Oshani Seneviratne;* International Conference on Decentralized Applications and Infrastructures 2019

- **Pre-processor** determines an action list
- **Smart Contract Execution Engine** executes the action that the voters selected
- Learning Voter Preferences



*Smarter* Contracts!  
Learning to adapt

# Example Hyperledger Fabric Implementation

```
Model
asset A{
  a1
  a2
}
participant
B{
  b1
  b2
}

Script{
  transaction
t1{
  //...
}
```

Analyze  
Smart  
Contract



Preprocessor

```
Model
asset A{
  a1
  a2
}
participant B{
  b1
  b2
}

learning_data
}

Script{
  transaction
t1{
  //...
}
//new
transactions
start_vote{}
submit_vote{}
end_vote{}
//action list
change_a1{}
change_a2{}
change_b1{}
change_b2{}
}
```



Voting



```
Model
asset A{
  a1
  a2
}
participant B{
  b1
  b2
  learning_data
}

Script{
  transaction t1{
  //...
}
//new transactions
start_vote{}
submit_vote{}
end_vote{}
//action list
change_a1{}
change_a2{}
change_b1{}
change_b2{}
}
```

Smart  
Contract  
Execution



Smart  
Contract  
Execution  
Engine

execute  
change\_a1()



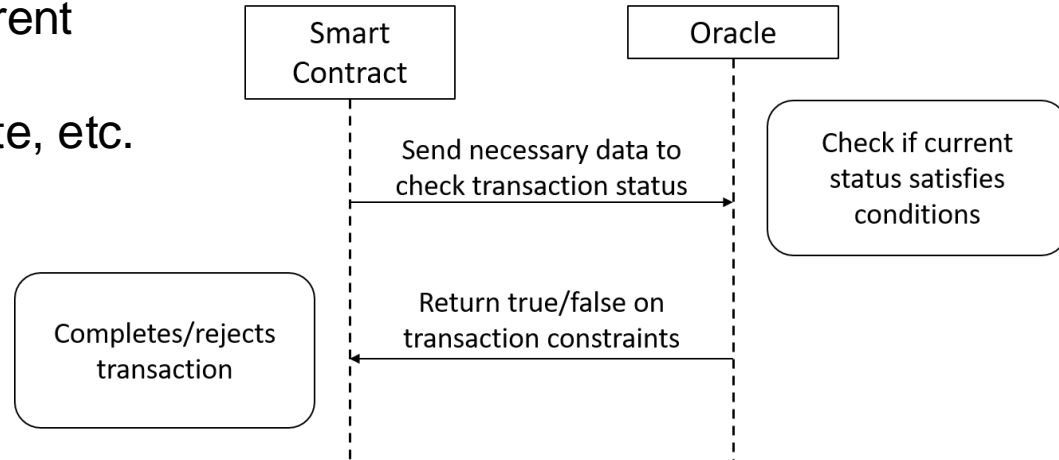
# Executing Transactions Aided by Off-Chain Oracles

## 1. An ontology for characterizing unexpected situations

- To share a common understanding of the structure of an unexpected situation.
- To enable reuse across different dApps.
- To determine who gets to vote, etc.

## 2. External rules to augment the smart contract logic

**Ontology Aided Smart Contract Execution for Unexpected Situations;** Farhad Mohsin, Xingjian Zhao, Zhuo Hong, Lirong Xia, Oshani Seneviratne; International Semantic Web Conference, 2019



# Our Contribution in Work

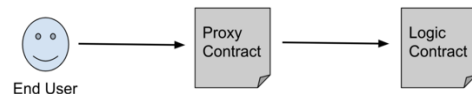
Investigated approaches for smart contract termination already available in several prominent blockchains and how they could support the EU Data Act mandate (or other such regulations) for smart contract “Kill Switches.”

Evaluated these solutions along the following dimensions:

- **Strategy** (built-in functions, design patterns, or other relevant features)
- **Strengths** (inherent strengths of the platform for smart contract termination)
- **Weaknesses** (inherent weaknesses of the platform for smart contract termination)
- **Governance** (protocols within the blockchain allow network participants to intervene or make decisions regarding the termination or pausing of smart contracts)
- **Regulation Support** (potential or existing support for compliance with regulatory frameworks)



- **Built-in function** in the Solidity language
  - Contracts can be deleted from the blockchain by calling `selfdestruct` - removes contract state and sends all remaining Ether stored in the contract to a designated address.
- **Emergency Stop Pattern**
  - **State Variable:** A Boolean (`isStopped`) tracks the contract's active or paused state.
  - **Modifiers:** `stoppedInEmergency`: Prevents function execution during an emergency; `onlyWhenStopped`: Allows functions specifically designed for emergencies; `onlyAuthorized`: Ensures only authorized entities (e.g., contract owner) can trigger a stop or resume.
  - **Pausable contract** from the OpenZeppelin library - allows children to implement an emergency stop mechanism that can be triggered by an authorized account
- Upgradeable contracts via the **Proxy Pattern**
  - Proxy contract uses storage variables to keep track of the addresses of other smart contracts that make up the dApp.
  - Using `delegatecall` via the `fallback` function of the Proxy contract, code in the logic contract has the power to change the state of the dApp.



# Strategies in Other Blockchains

- Emergency Stop / Pause Patterns
- Upgradable Contracts



CARDANO



ethereum



BINANCE  
SMART CHAIN



SOLANA

- Strong type system
- Robust programming languages

APTES



- Code lifecycle management
- Administrative control



HYPERLEDGER  
FABRIC

Explicit Termination  
Conditions

c.rda

- Ability to respond to external inputs or triggers that could include termination signals







# Summary of Solutions

PDF Preview

Blockchain	Strategies	Strengths	Weaknesses	Gov.	Regulation Support
Ethereum [21] & BNB Smart Chain [22]	Self-destruct function included in the Solidity language; Pause and emergency stop design patterns; Upgradeable contracts.	Provides built-in functions for contract termination; Compatible with the widespread tools and infrastructure.	No external mechanism; Potential security risks; Possible removal of the self-destruct function raises concerns about long-term viability.	No	Yes, through custom implementations using the Solidity features.
Cardano [23]	Design-specific conditions within smart contracts built into Plutus; Stateful smart contracts; Seamless interaction with off-chain code.	Uses a robust functional programming language (Haskell) for Plutus; Strong on-chain governance mechanisms.	No external mechanism; Complex implementation and limited adoption compared to Ethereum.	Yes	Yes, through design-specific conditions.
Solana [24]	Upgradable programs; State management.	High throughput and low latency with upgradable programs.	No external mechanism; Immaturity of the ecosystem and less community support for governance models.	No	Yes, through upgradable programs.
Hyperledger Fabric [25]	Chaincode lifecycle management; Endorsement policies; Private data collection; Administrative control.	Permissioned blockchain with strong lifecycle management and administrative controls.	Centralized nature might not align with decentralization principles.	Yes	Yes, through administrative control and governance mechanisms.
Corba [26]	Built-in contract upgrade; Explicit termination conditions; Administrative control.	Focus on privacy and business transactions with upgradable contracts.	Limited use cases outside of enterprise applications.	Yes	Yes, through explicit contract conditions.
IOTA [27]	State management built into the ISCP; Ability to respond to external inputs or triggers that could include termination signals.	Scalable with no transaction fees suitable for IoT.	Still evolving with ongoing updates to smart contract capabilities.	Yes	Yes, through decentralized control mechanisms.
Aptos [28] & Sui [29]	Move language flexibility for contract updates; Expressive smart contract implementations tracking and managing assets.	Strong type system for formal verification and security; Supports more complex governance and transaction models.	Newer ecosystems with less mature tooling and support.	Yes	Yes, through explicit contract conditions.



# Pros and Cons of Smart Contract Kill Switches

Aspect	Pros	Cons
Security 	Enhances protection against vulnerabilities and bugs.	Target for malicious actors if the keys are not securely managed, and thus potential loss of assets.
Compliance 	Facilitates compliance with regulations like the EU Data Act.	Conflicts with the principle of immutability and decentralization in blockchains.
Governance 	Can be designed to involve community consensus.	Might introduce elements of central control.
User Trust 	Increases confidence in safety mechanisms.	Users may fear misuse or overreach, i.e., concerns regarding the true decentralization of blockchain.



# Discussion and Q&A

- **Kill switches are like emergency brakes:** great when things go wrong, but they come at a cost (immutability and decentralization challenges).
  - Their implementation needs to be carefully balanced to avoid undermining blockchain's core principles.
- **There's already diverse support across blockchains** when terminating or pausing smart contracts is thought out in advance.
- Just like we debate kill switches for AI (to prevent the rise of robot overlords), smart contract kill switches **demand similar scrutiny to ensure they are safe without stifling innovation.**

**Contact:** Oshani Seneviratne ([senevo@rpi.edu](mailto:senevo@rpi.edu))

<https://faculty.rpi.edu/oshani-seneviratne>



# Rensselaer

**why not change the world?®**

# Ethereum and BNB Smart Chain

- **Strategies**
  - Self-destruct function in the Solidity language
  - Pause and emergency stop design patterns
  - Upgradeable contracts
- **Strengths**
  - Provides built-in functions for contract termination
  - Compatible with the widespread tools and infrastructure.
- **Weaknesses**
  - No external mechanism
  - Potential security risks
- **Governance**
  - No
- **Regulation Support**
  - Yes, through custom implementations using the Solidity features



ethereum



# Cardano

- **Strategies**
  - Design-specific conditions within smart contracts built into Plutus
  - Stateful smart contracts
- **Strengths**
  - Uses a robust functional programming language (Haskell) for Plutus
  - Strong on-chain governance mechanisms.
- **Weaknesses**
  - No external mechanism
  - Complex implementation and limited adoption compared to Ethereum.
- **Governance**
  - Yes
- **Regulation Support**
  - Yes, through design-specific conditions.





- **Strategies**
  - Upgradable programs
  - State management.
- **Strengths**
  - High throughput and low latency with upgradable programs.
- **Weaknesses**
  - No external mechanism
  - Immaturity of the ecosystem and less community support for governance models
- **Governance**
  - No
- **Regulation Support**
  - Yes, through upgradable programs

# Hyperledger Fabric



- **Strategies**
  - Chaincode lifecycle management
  - Endorsement policies
  - Private data collection
  - Administrative control
- **Strengths**
  - Permissioned blockchain with strong lifecycle management and administrative controls
- **Weaknesses**
  - Centralized nature might not align with decentralization principles
- **Governance**
  - Yes
- **Regulation Support**
  - Yes, through administrative control and governance mechanisms



- **Strategies**
  - Built-in contract upgrade
  - Explicit termination conditions
  - Administrative control
- **Strengths**
  - Focus on privacy and business transactions with upgradable contracts
- **Weaknesses**
  - Limited use cases outside of enterprise applications
- **Governance**
  - Yes
- **Regulation Support**
  - Yes, through explicit contract conditions

- **Strategies**
  - State management built into the protocol
  - Ability to respond to external inputs or triggers that could include termination signals
- **Strengths**
  - Scalable with no transaction fees suitable for IoT
- **Weaknesses**
  - Still evolving with ongoing updates to smart contract capabilities
- **Governance**
  - Yes
- **Regulation Support**
  - Yes, through decentralized control mechanisms





# Aptos & Sui

- **Strategies**
  - Move language flexibility for contract updates
  - Expressive smart contract implementations tracking and managing assets
- **Strengths**
  - Strong type system for formal verification and security
  - Supports more complex governance and transaction models
- **Weaknesses**
  - Newer ecosystems with less mature tooling and support
- **Governance**
  - Yes
- **Regulation Support**
  - Yes, through explicit contract conditions

