WASHINGTON STATE UNIVERSITY

How Can Blockchain Enhance Trust in Digital Forensics and Evidence?

Asma Jodeiri Akbarfam

WSU Tri-Cities School of Engineering and Applied Sciences

About me!

- Recently got my PhD in computer and cyber sciences
- **Research Focus**

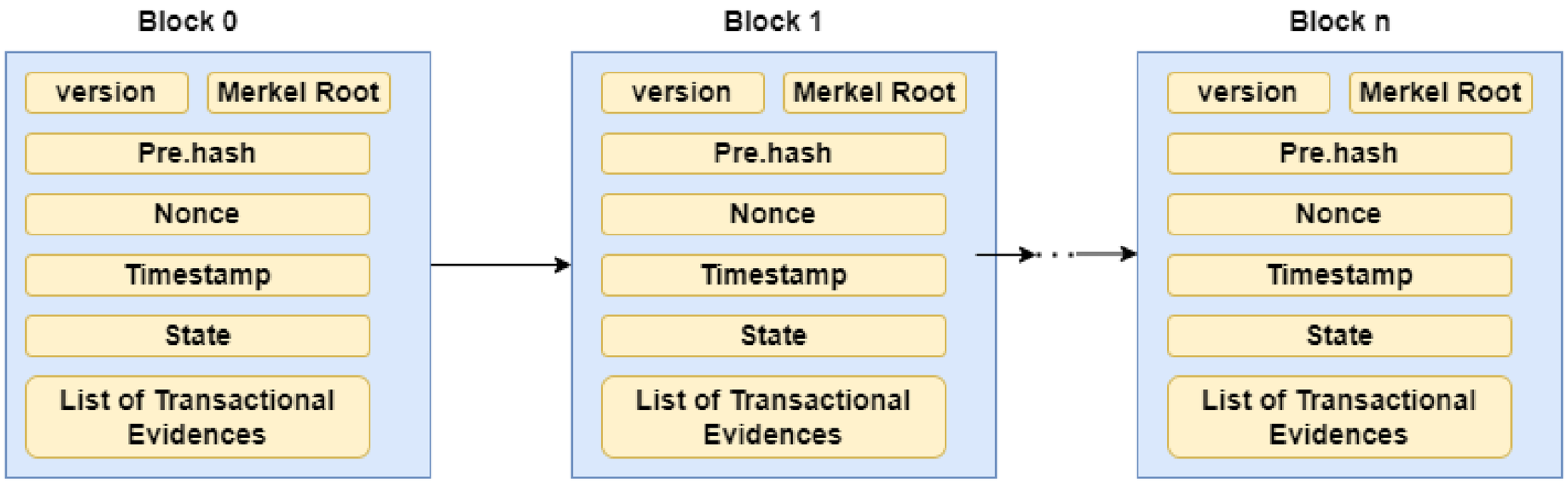
  Secure & scalable frameworks for blockchain,

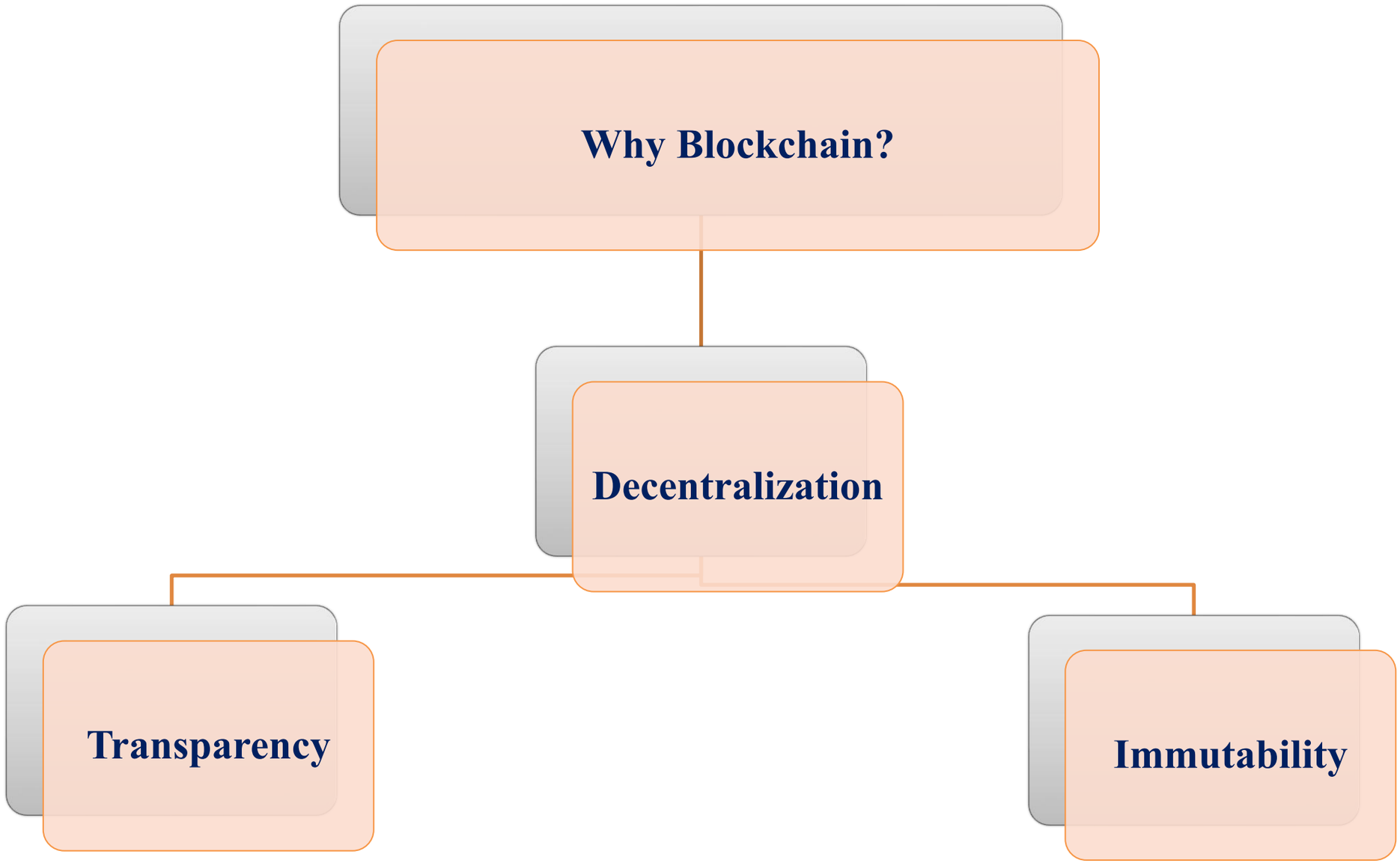
  AI-driven intrusion detection, and network security

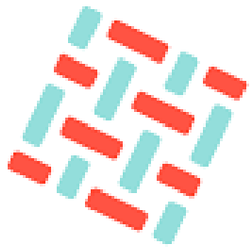
- **Faculty Advisor, Women in Cybersecurity (WiCyS), WSU Tri-Cities**

Walmart

Blockchain

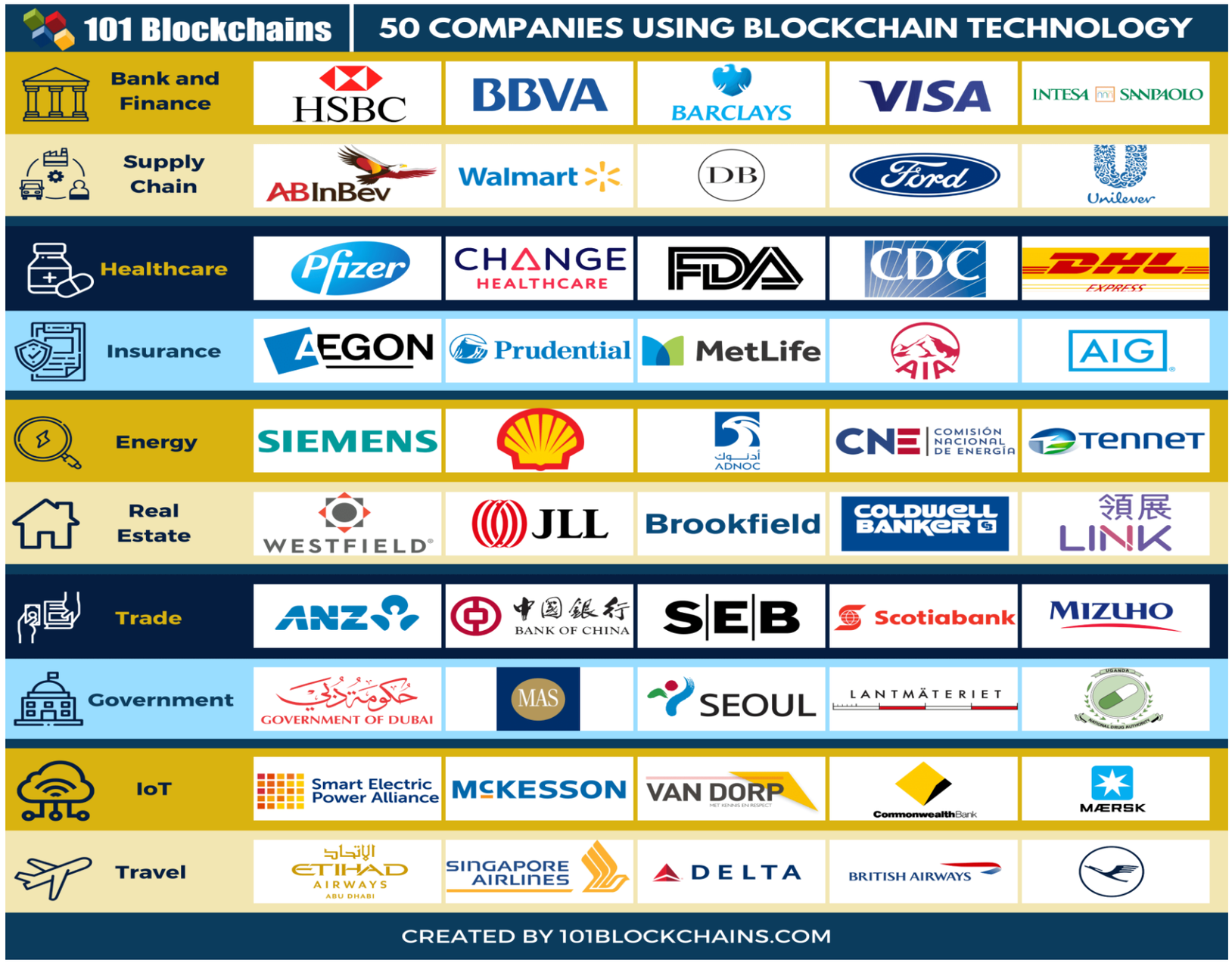

Block 0
version
Merkel Root
Pre.hash
Nonce
Timestamp
State
List of Transactional Evidences
Block 1
version
Merkel Root
Pre.hash
Nonce
Timestamp
State
List of Transactional Evidences
Block n
version
Merkel Root
Pre.hash
Nonce
Timestamp
State
List of Transactional Evidences


Why Blockchain?
Decentralization
Transparency
Immutability



Types Of Blockchains

Blockchain Applications



What are the Frameworks?

Data Collections	Neisse et al. (2017)
IoT	MLAKumar, Narendra, et al (2025). Hu et al. (2020), Caro et al. (2018), Pahl et al. (2018), Javaid et al. (2018), Ali et al. (2018)
Supply Chain Management	Gurupatham, Twinkle et al (2025),Cui et al. (2019)
Machine Learning	Kulothungan, Vikram(2025),Luthi et al. (2020)
Cloud Computing	Zhang et al. (2017)
Scientific Workflows	Haq, Rashid Ul, et al(2025),Coelho et al. (2019), Hoopes et al. (2022), Ramachandran et al. (2018), Nizamuddin et al. (2019)
Digital Forensics	Mbimbi, Butrus(2025) ,Siddiqui et al. (2023), Troyer et al. (2021)Li et al. (2019), Borse et al. (2021), Ahmed et al. (2023), Agbedanu et al. (2023), Siddiqui et al. (2023), Hoopes et al. (2022), Ruan et al. (2021), Tosh et al. (2019), Demichev et al. (2018), Sifah et al. (2018), Duong et al. (2021), Lone et al. (2018), Tsai et al. (2021)

Provenance in Different Domains

Domains

Meta Data

Product Supply Chain	Digital Forensics	Scientific Collaboration
Unique Product ID	Case Number	Task ID
Batch or Lot Number	Investigation Stage	Workflow ID
Manufacturing and Expiration Date	Case Start Date	Execution Time
Travel Trace	Case Closure Date	User ID
Product Type or Category	File Types	Input Data
Manufacturer ID	Access Patterns	Output Data
Quick Access URL or QR Code	Files Dependency	Invalidated Results

What are the Important Requirements of such Frameworks?

- Access control
- Provenance
  - Capturing
  - Querying
  - Verifying
- Domain Specific Needs
- Security
- Expansion
- Evaluate the framework

Digital Forensics

Identification	Preservation	Analysis	Documentation	Presentation
				
Locating digital evidence (e.g.. emaiis, logs, deleted files)	Making exact copies (images) of data without altering it	Examining the data using forensic tools to uncover patterns or reconstruct events	Keeping detailed logs of how evidence was handled	Explaining findings clearly in court or a formal report

ForensiBlock [29]

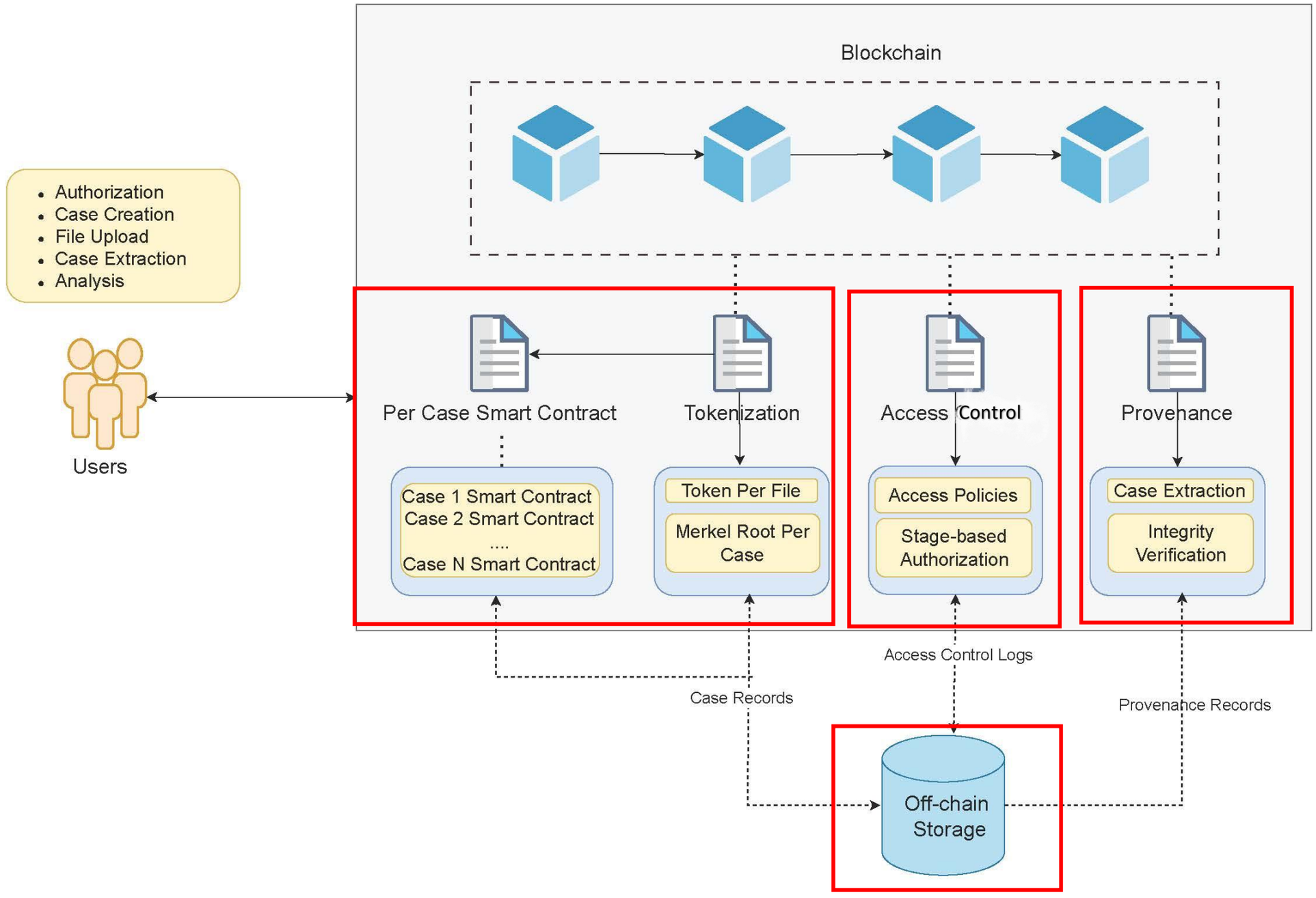
Objectives

- Demonstrate and manage distinct investigation stages
- Track data versioning across the forensic process
- Control access and manage dynamic policy updates
- Ensure a verifiable chain of custody throughout the lifecycle of digital evidence

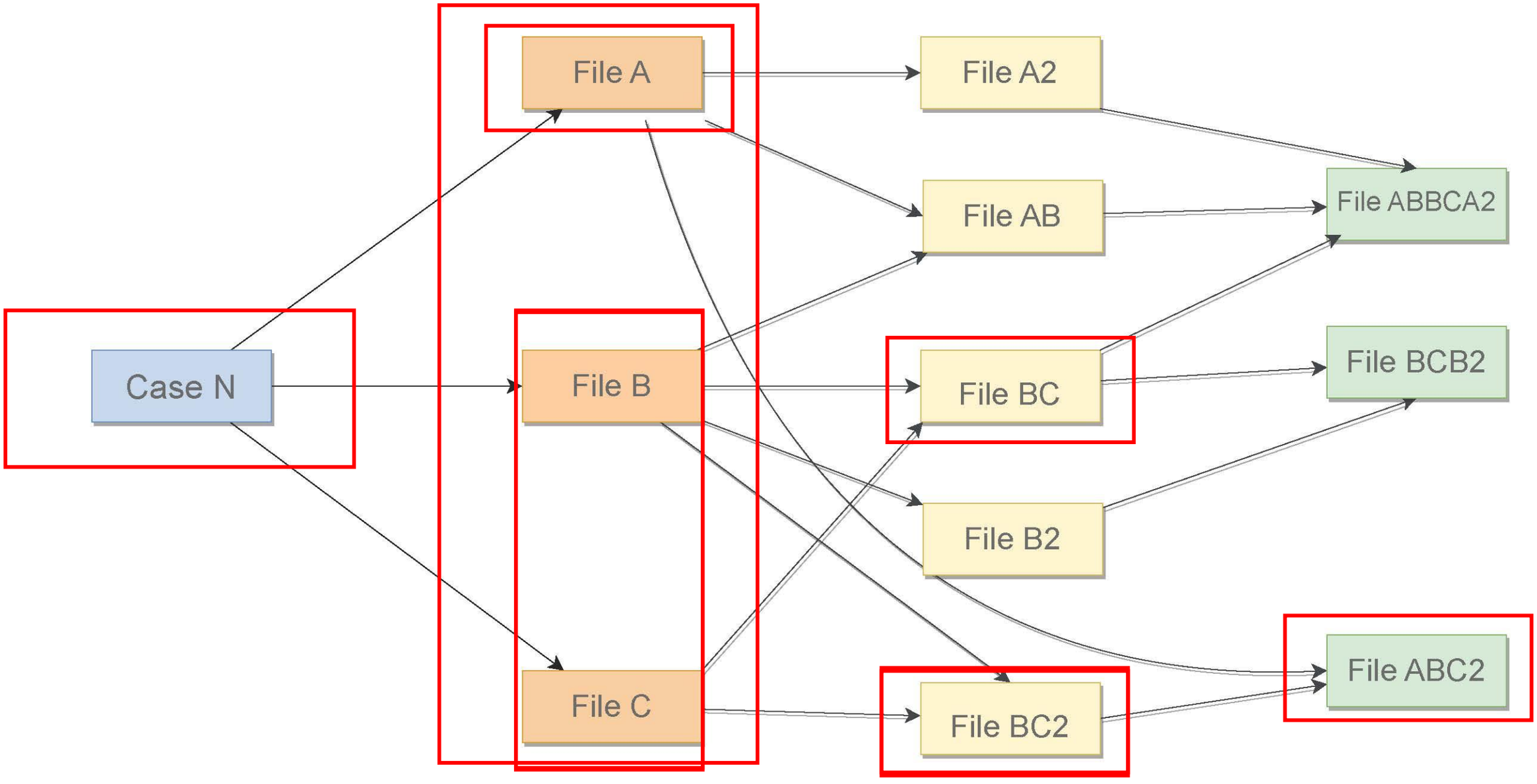
Research Questions

- How can blockchain be adaptively integrated with digital forensics workflows?
- How can we verify and manage multiple versions of digital evidence?
- How can we log forensic provenance?
- How can provenance be extracted efficiently and reliably?
- What access control method best fits the needs of digital forensics?

ForensiBlock Architecture



Data Dependency



Role-Based Access Control with Staged Authorization (RBAC-SA)

Protocol 1 RBAC-SA Protocol

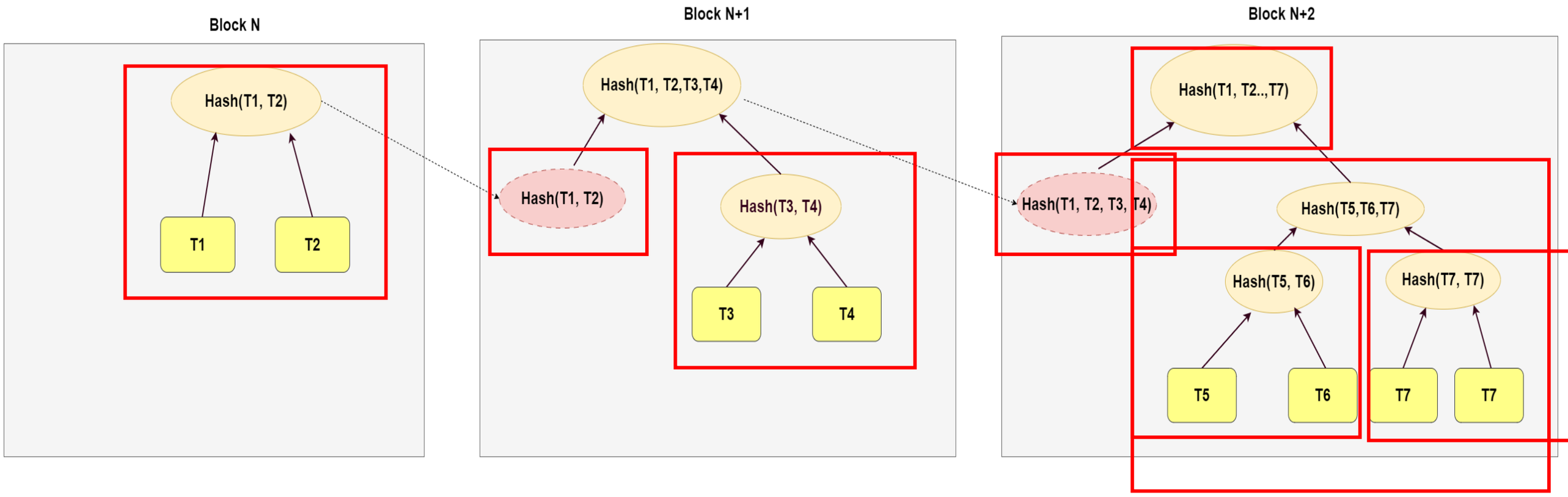
```
1:  procedure RETRIEVEACCESSINFO(transaction)
2:      Read role and stage information from files
3:      Stored_stage = transaction['current_stage']
4:      Sender_publik_key = transaction['sender_public_key']
5:      if current_stage is invalid or ≠ Stored_stage then
6:          return 'Invalid stage'
7:      Retrieve public_key_role based on Sender_public_key
8:      if public_key_role is None then
9:          return 'Access Denied'
10:     Retrieve public_key_rights based on current_stage and
11:     public_key_role
12:     if public_key_rights is None then
13:         return 'No access rights'
14:     return public_key_rights
```

What about Provenance records?

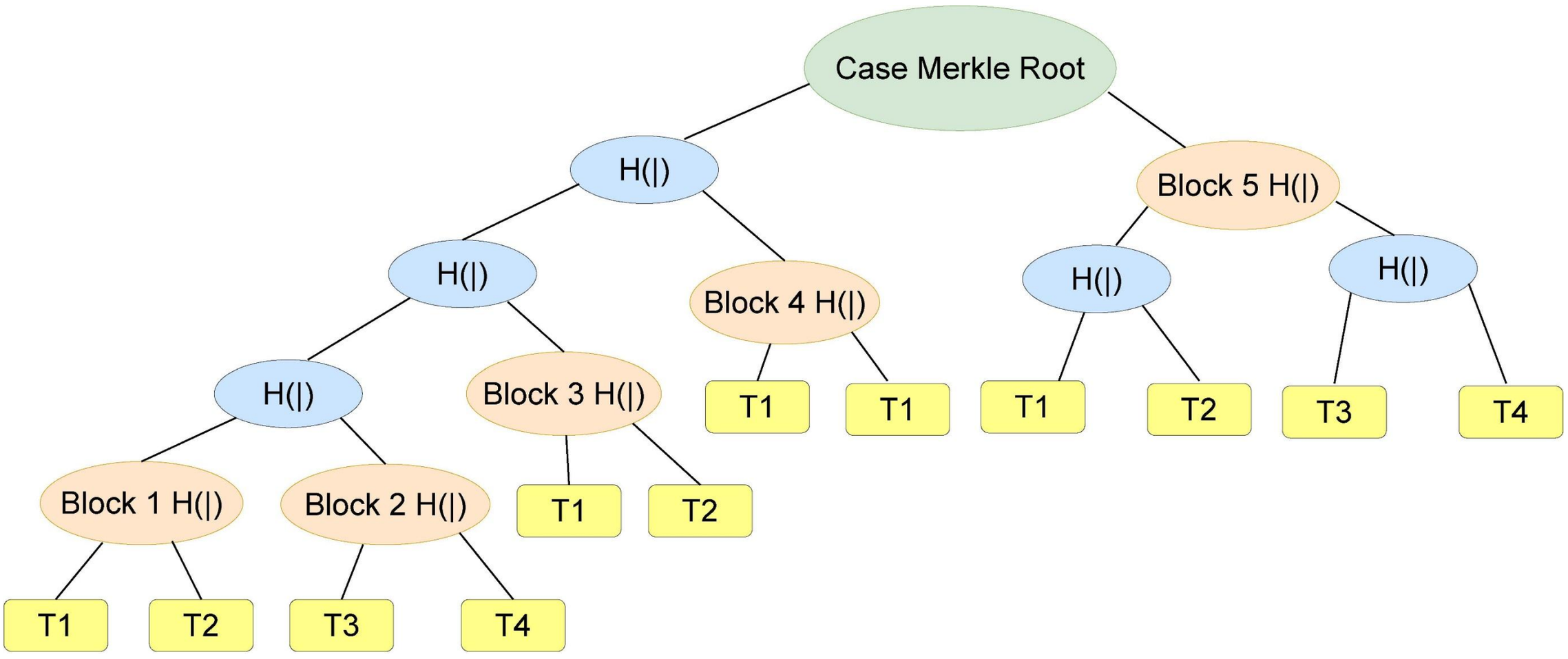
Record	**Description**
Case Number	Unique identifier for a case
Timestamp	Time of the action/event
Initial Block Number	Block where the initial case transaction was recorded
Current Stage	Current stage of the case
Token List	List of tokens associated with the case
Access Request	User request to access specific case data
Client Info	Information about the user/client
Token dependency	analyzed Tokens drived from a files
Validity of Access	Matching system permissions for the access request
Stage Change	Change in the tokenized smart contract's stage
Type of Data Upload	Indicates raw data or analyzed data

How to Accelerate the Extraction of Provenance Records?

Merkle Root Construction for Distributed Case Tracking



Merkle Tree per each case for validation with storage



Verification of Merkle Root

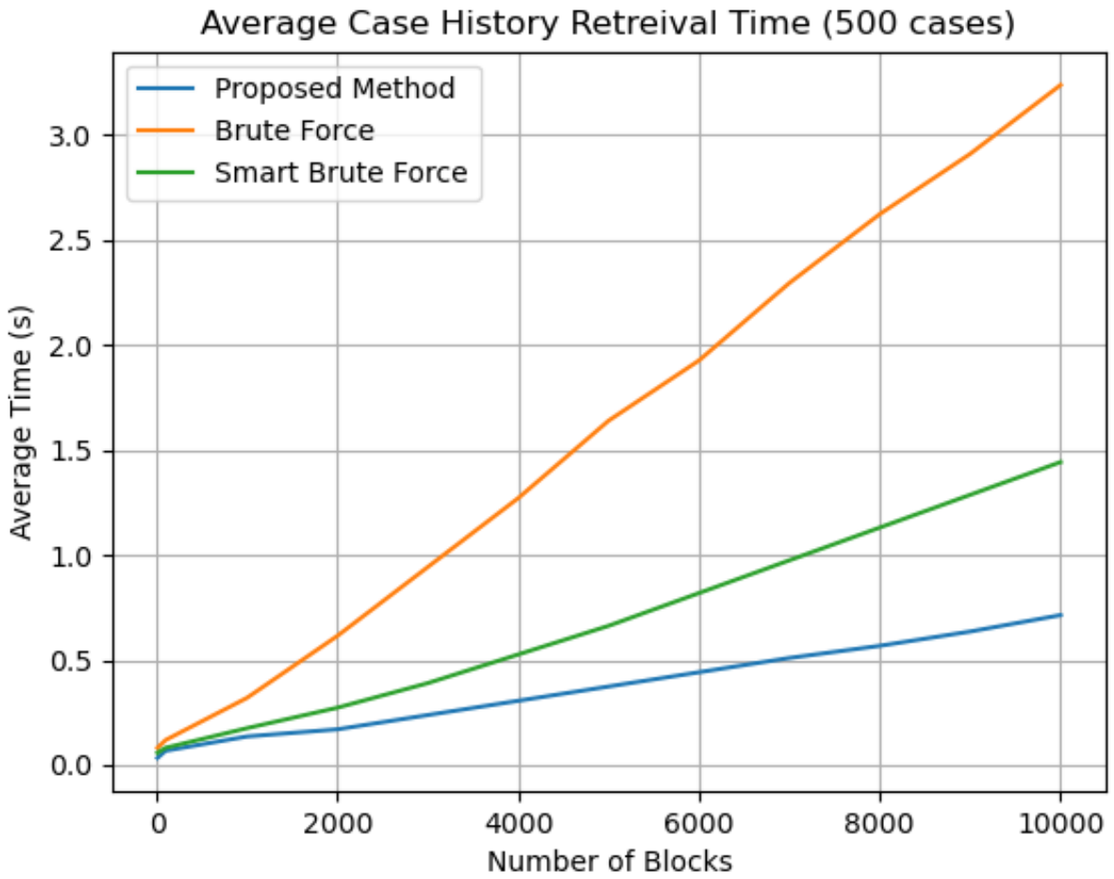
Algorithm 1 Enhanced Storage Verification

procedure STORAGEVERIFICATION(*case_number*, *sorted_case_list*)
    $M_{\text{case}} \leftarrow$ null
    for *block* **in** *sorted_case_record* **do**
        $M_{\text{block}} \leftarrow$ null
        $M_{\text{block}} \leftarrow$ BlockMerkleRoot(*block_case_record*)
        $M_{\text{case}} \leftarrow M_{\text{case}} \parallel M_{\text{block}}$
    stored_merkle_root ← Retrieve stored Merkle root for the investigation
    if M_{case} matches stored_merkle_root **then**
        Data integrity verified
    else
        Data integrity compromised

Implementations

- Evaluate ForensiBlock under varying conditions with up to **10,000 blocks** and **1,000 synthetic cases**
- **Case Initialization:**
  - Cases begin with InitialUpload to trigger first stage (e.g., *Investigation*, *Court Presentation*).
  - New cases introduced probabilistically
- **Randomized sampling**
  - Case Progression
  - Synthetic Transaction Types

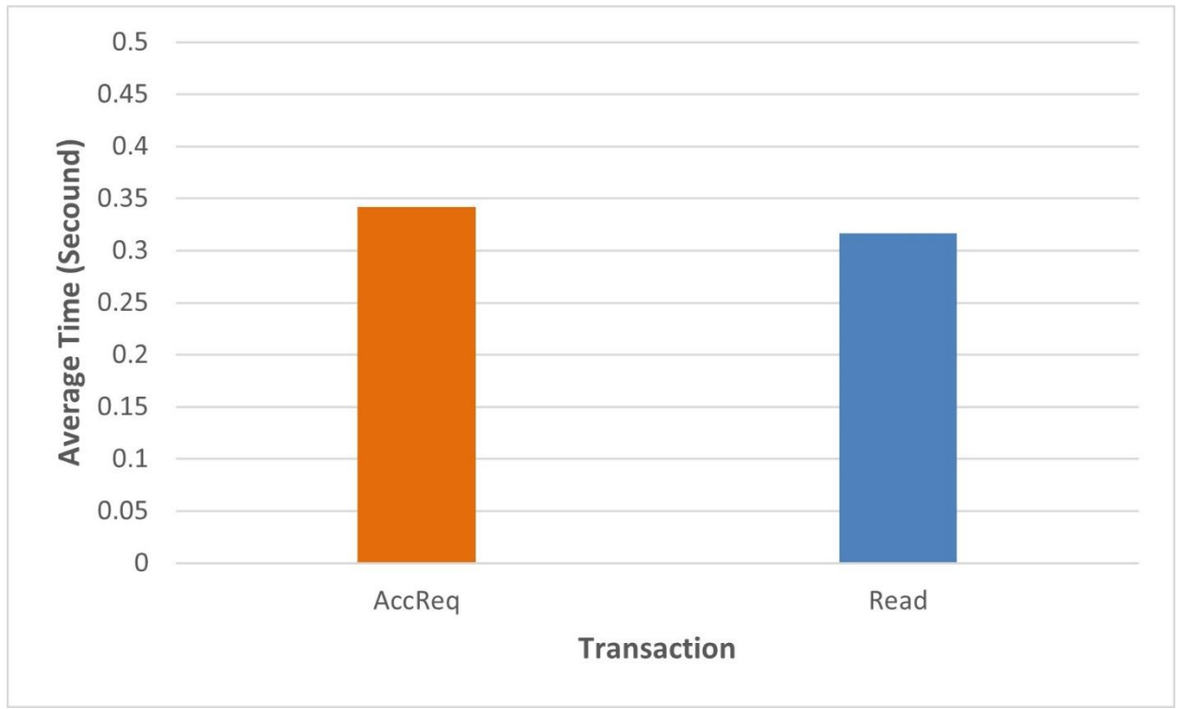
Retrieval Time for Different Quantity of Cases



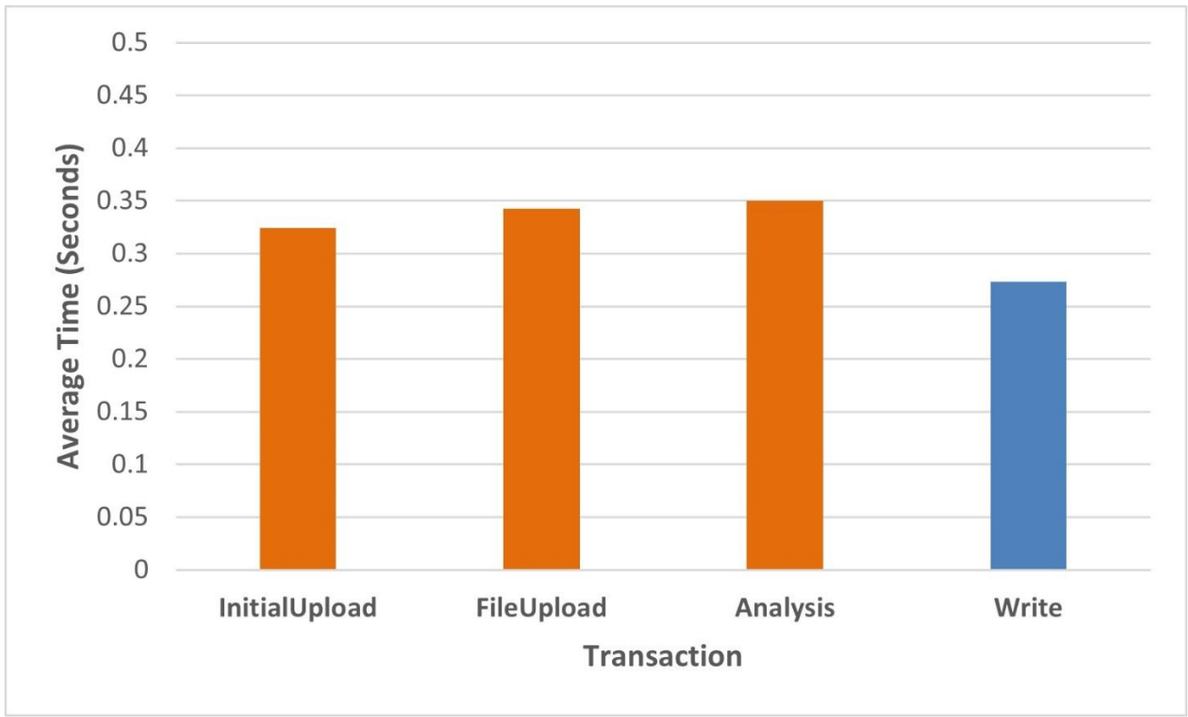
- Proposed Method: Retrieval from storage + algorithm
- Brute Force: Searching inside the blockchain
- Smart Brute Force: Searching in the blockchain starting from the initial block number

Results With Transaction Processing

Read Oriented transactions



Write Oriented transactions



- Simple read-only transactions; by contrast, the read-type transactions in our workload are significantly richer and therefore harder to process
- Write transactions; existing results use straightforward write operations, whereas our write-type cases involve considerably greater complexity

ForensiBlock a Secure Framework

Security Guarantee	Implementation Element	Purpose
Integrity	Hashes, signature checks	Ensure data wasn’t altered
Accountability	Mem logs, user IDs, audit trail	Know who did what
Non-Repudiation	Signature validation	Prevent denial of actions

❑ Research questions:

- ❑ How can we prove that ForensiBlock is secure under adversarial conditions?
- ❑ Can ForensiBlock preserve its security when composed with other protocols or frameworks?

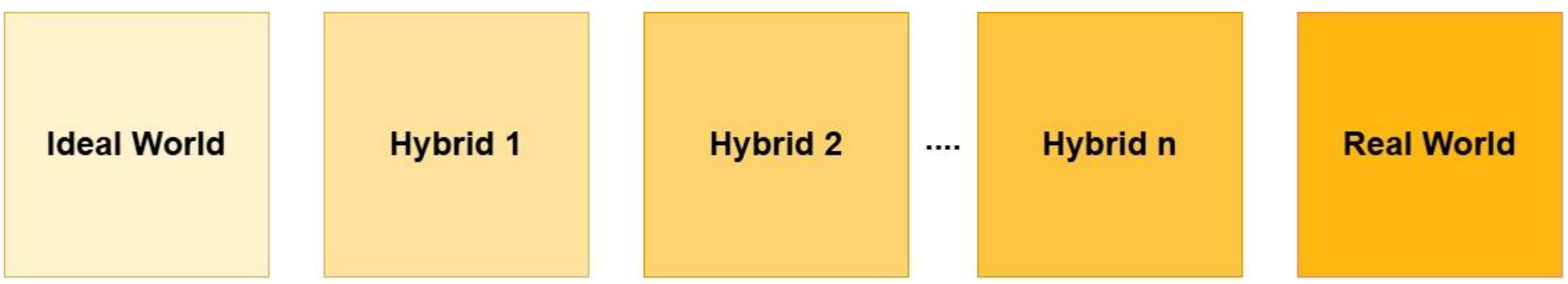
Secure when reused or combined

- ❑ Support Modular Design
- ❑ Other Security-Proof Paradigms
  - **Trusted-party paradigm** Goldreich *et al[1]*
  - **Synchronous communication** Goldwasser *et al[2]*
  - X Multiple protocol executions running simultaneously
  - **Modular composition** Goldwasser *et al[3]*
  - **Reactive simulatability** Pfitzmann *et al* [4],
    - X Limited in scope, either focusing on single-session security or static environments
- ❑ Universal Composability [5]
  - ✓ Security preserved under arbitrary concurrency
  - ✓ Handles adaptive corruptions
  - ✓ Prove each module once ⇒ plug-and-play reuse

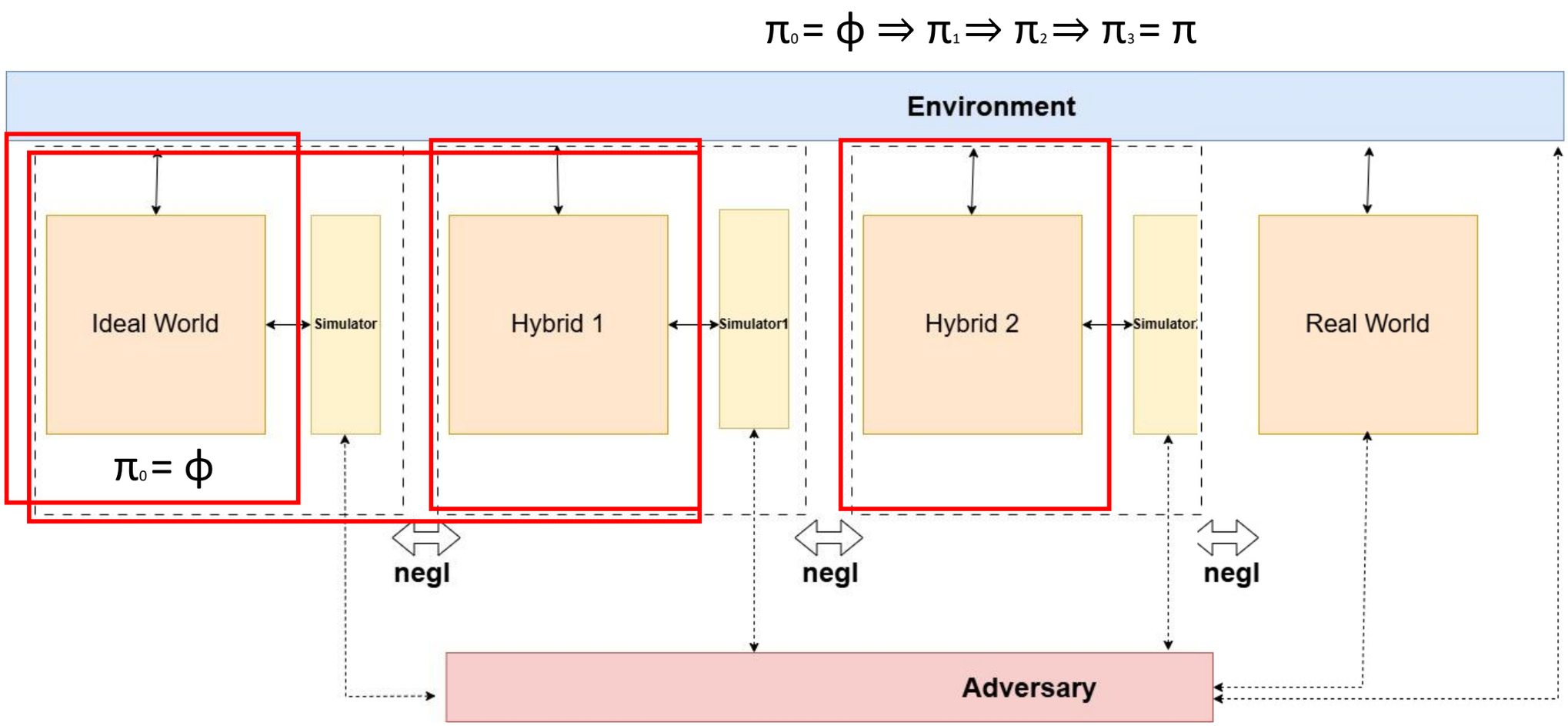
[1] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game,or a completeness theorem for protocols with honest majority,” in ProvidingSound Foundations for Cryptography: On the Work of Shafi Goldwasser andSilvio Micali, 2019, pp. 307–328.
[2] S. Goldwasser and L. Levin, “Fair computation of general functions inpresence of immoral majority,” in Conference on the Theory and Application ofCryptography. Springer, 1990, pp. 77–93.
[3] R. Canetti, “Security and composition of multiparty cryptographic protocols,”Journal of CRYPTOLOGY, vol. 13, pp. 143–202, 2000.[20] B. Pfitzmann, M. Schunter, and M. Waidner, “Cryptographic security ofreactive systems,” Electronic Notes in Theoretical Computer Science, vol. 32,pp. 59–77, 2000.
[4] B. Pfitzmann, M. Schunter, and M. Waidner, “Cryptographic security ofreactive systems,” Electronic Notes in Theoretical Computer Science, vol. 32,pp. 59–77, 2000
[5] Canetti, Ran. "Universally composable security." *Journal of the ACM (JACM)* 67.5 (2020): 1-94.

Universal Composable Security (UC)

- ❑ Remains secure even in real-world environments with adversaries
- ❑ Security proven through ideal vs. real world indistinguishability
- ❑ Enables stepwise security proofs via intermediate hybrid models

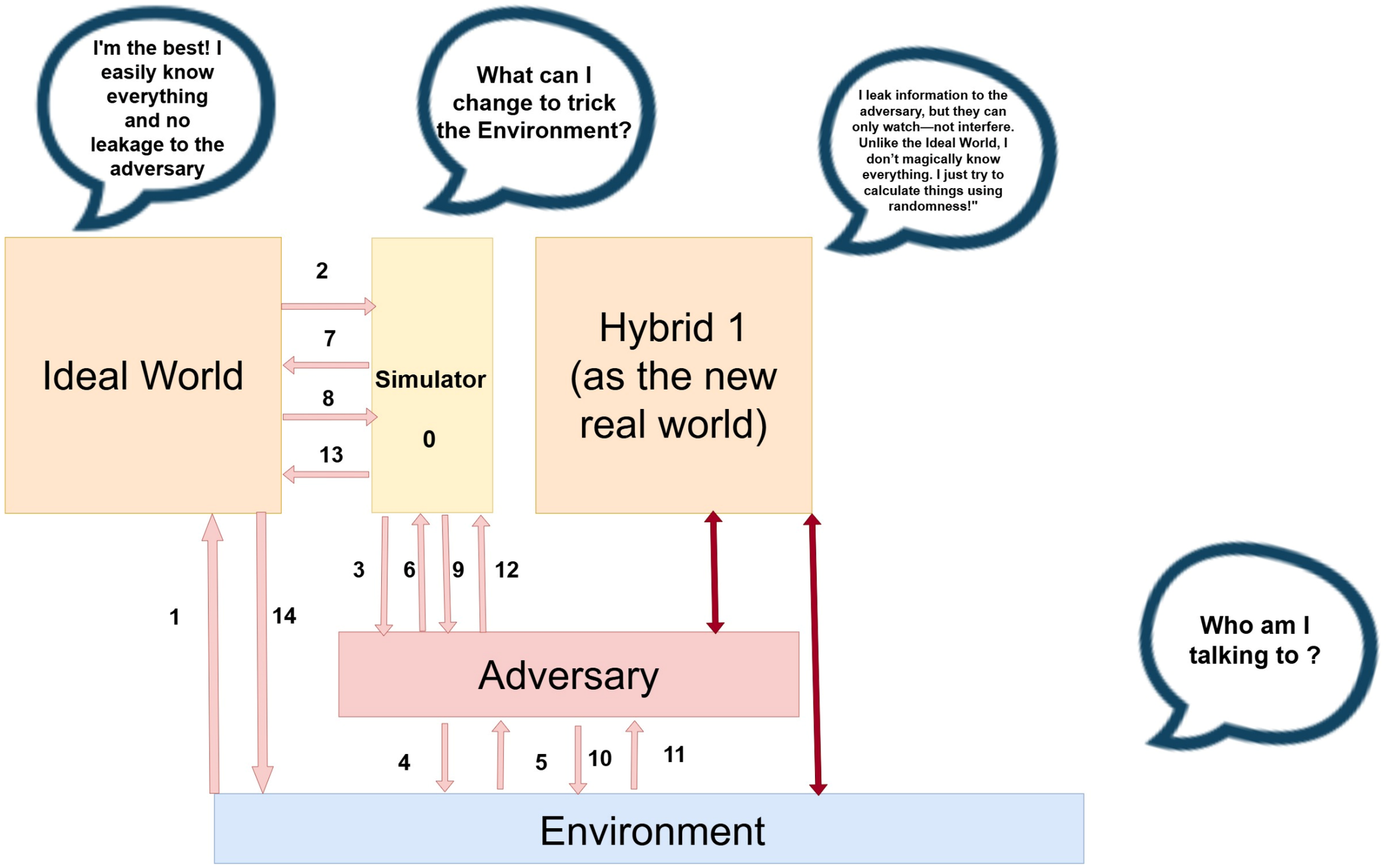


UC Security Analysis of ForensiBlock

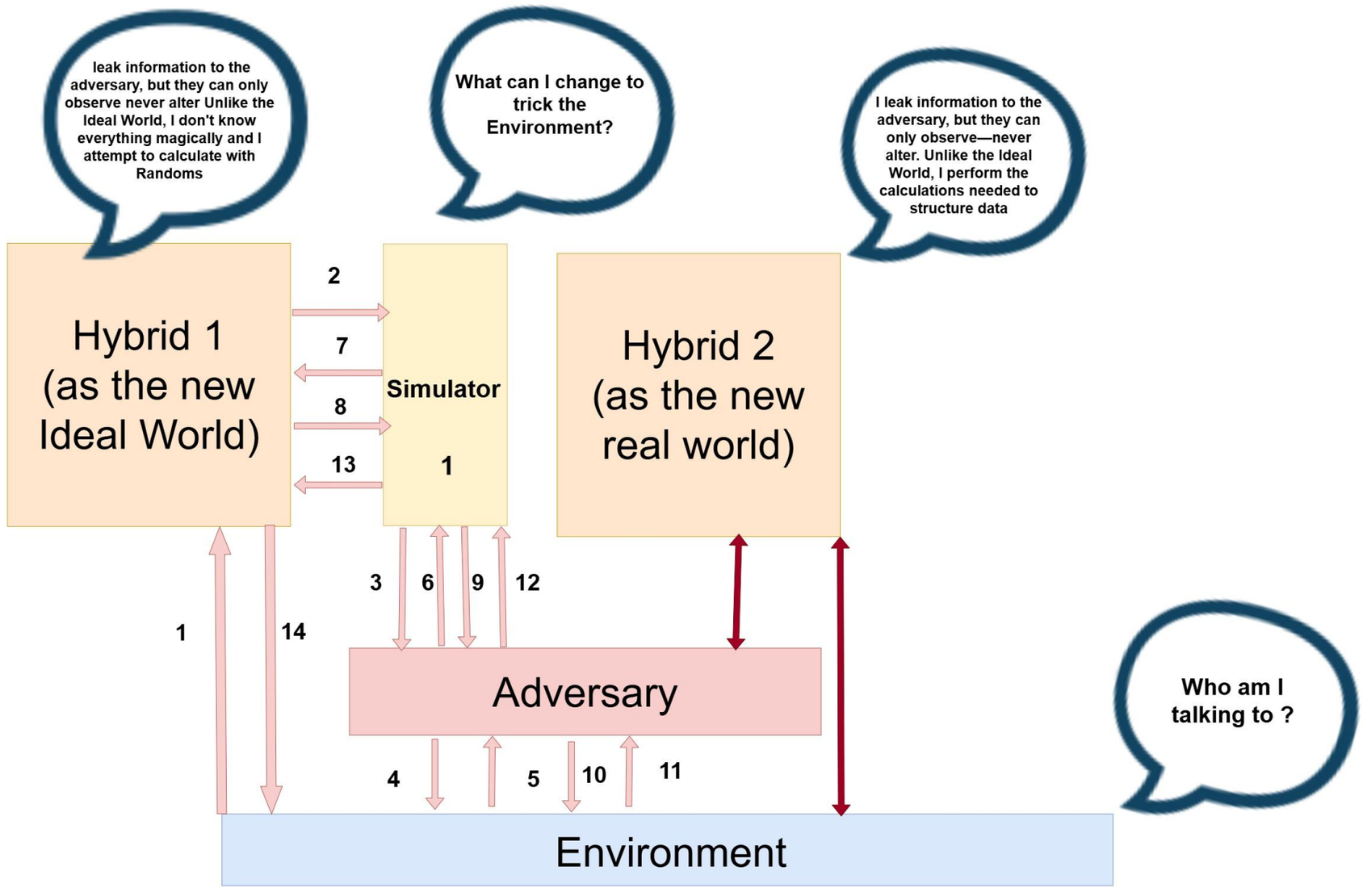


Hybrid 1: Adversary can observe
Hybrid 2: Adding Contract Functionality
Hybrid 3 (real world) : Adding Signature Scheme, Adversary can change

First Step (π1, A) & (π0, S)



Second Step (π2, A) & (π1, A)



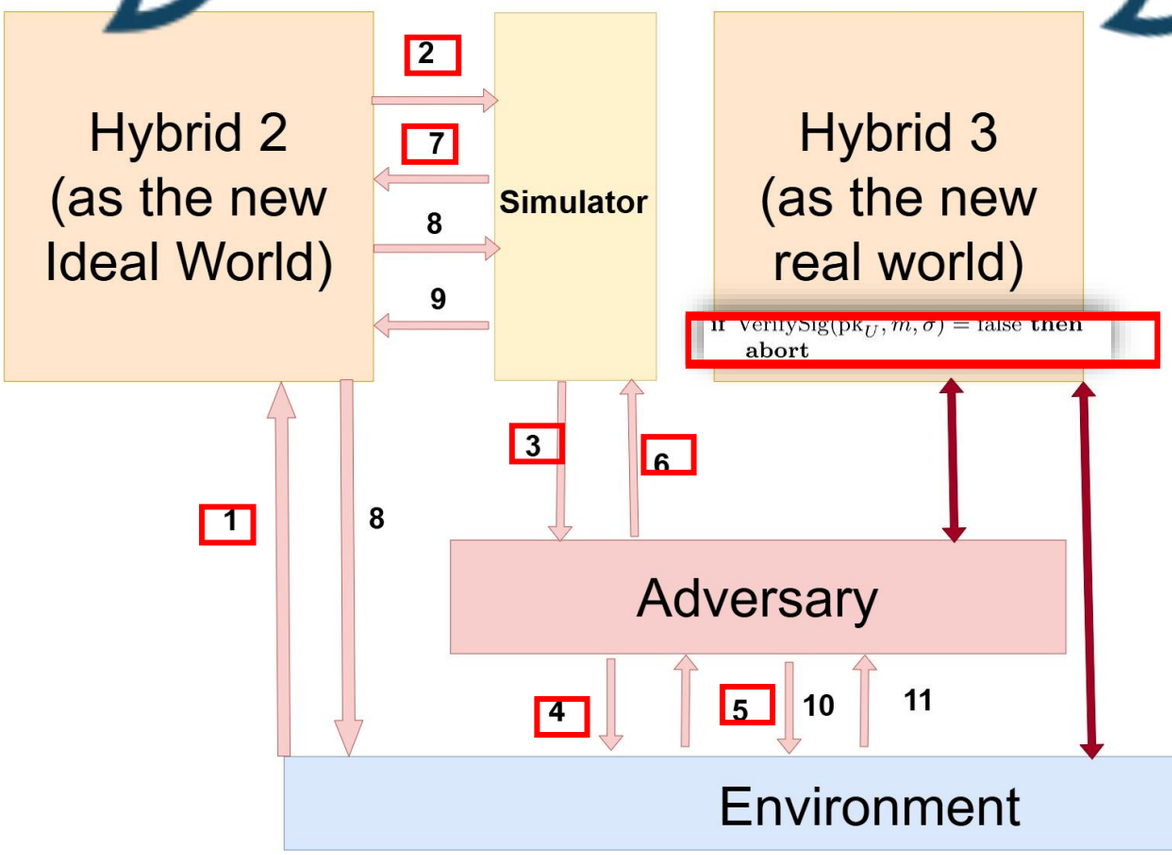
Third Step (π2, A) & (π3, A)

function HYBRID2CASECREATION(CreateCase, CID, Roles, U, σ)
Upon receiving (**CreateCase**, **CID**, **Roles**, U, σ) **from** E
Assert the current time T.
Record (CreateCase, CID, Roles, U, Firststep, T) in Mem.
Send (CreateCase, CID, Roles, U, σ, Firststep) to S

I leak information to the adversary, but they can only observe—never alter. I perform the calculations needed to structure data

What can I change to trick the Environment?

The adversary can edit and I can detect it!



function SIMCASECREATIONHYBRID2(CreateCase, CID, Roles, U)
Upon receiving (**CreateCase**, **CID**, **Roles**, U, σ, **Firststep**) **from** (Π_2)
Record (CreateCase, CID, Roles, U, Firststep, σ) in Mem
(PK, SK) ← KEYGEN(U)
Record (PK, SK) in Mem where CID = CID″
σ″ := Sign$_{SK}$(CreateCase, CID, Roles, U, σ)
PendingSessions ← [(CID)]
Send (CreateCase, CID, Roles, U, σ) to $\mathcal{A}$
Upon receiving (**CreateCase′**, **CID′**, **Roles′**, U', σ''') **from** $\mathcal{A}$
Extract CID″ from PendingSessions
Extract (CreateCase, CID, Roles, U, SK, PK, step, σ″) from Mem where CID″ = CID
$\sigma' :=$ Sign$_{SK}$(CreateCase′, CID′, Roles′, U', σ''')
if $\sigma' \neq \sigma$″ **then**
ForgeryDetected ← true
else
ForgeryDetected ← false
if ForgeryDetected = true **then**
abort
if step = Firststep **then**
PendingSessions.remove(entry)
Send (CreateCase, CID, Roles, U, ForgeryDetected) to (Π_2)
else
abort

Who am I talking to ?

Algorithm 3.13 Hybrid 2 ForensiBlock (Π_2)

function HYBRID2CASECREATION(CreateCase, CID, Roles, U, σ)
  Upon receiving (**CreateCase**, **CID**, **Roles**, U, σ) **from** E
  Assert the current time T.
  Record (CreateCase, CID, Roles, U, Firststep, T) in Mem.
  Send (CreateCase, CID, Roles, U, σ, Firststep) to S
  Upon receiving (**CreateCase**, **CID′**, **Roles**, U) **from** S
  if record with CID′ exists in Mem and step is Firststep **then**
    Extract (CreateCase, CID, Roles, U, Firststep) from Mem where CID′ = CID.
    Extract the set `CorruptedUsers` from `Mem`
    if $U \in$ `CorruptedUsers` **then**
      Abort
    Extract the set `CorruptedUsers` from `Mem`
    if $U \in$ `CorruptedUsers` **then**
      Abort
    Extract ($\text{Roles}_{\text{stored}}$, StoredStage) from Mem where CID′ = CID
    Set CurrentStage ← stage0
    if $U' \notin \text{Roles}_{\text{stored}}$ **then**
      abort
    else if CurrentStage ≠ StoredStage **then**
      Result ← "fail" ▷ Invalid stage
    else
      Role ← `lookupRole`(U', $\text{Roles}_{\text{stored}}$)
      Rights ← `lookupRights`(CurrentStage, Role)
      if "CreateCase" ∉ Rights **then**
        Result ← "fail" ▷ Role not permitted to create case
      else
        Result ← "Successful"
    Extract (M_{prev}, block_number, $H(T_{\text{block}})$) from Mem where CID′ = CID
    $M_{\text{case}} \leftarrow \text{Hash}(\text{CreateCase}\|\text{CID}\|\text{Roles}\|U\|\text{"Result"}\|H(T_{\text{block}}))$ ▷ Generate Merkle root
    Update Mem with (Result, CID, $M_{\text{case,Roles},U}$, T_number, U, T, Finalstep)
    Record (Result, CreateCase, CID, M_{case}, block_number, T_{block}, U) in *SimProvenance*
    Send (Result, CreateCase, CID, Roles, U, σ, block_number, $H(T_{\text{block}})$, Finalstep) to S
  else
    abort
  Upon receiving (**Result**, **CreateCase**, **CID**, **Roles**, U) **from** S
  if record with CID′ exists in Mem **then**
    Extract (Result, CreateCase, CID, Roles, U, σ, T, Step) from Mem where CID′ = CID.
    if Step = Finalstep **then**
      Send (Result, CreateCase, CID, Roles, U, σ, T) to E.
      else
      abort.
    else
    abort.

Hybrid 2 ForensiBlock (Π_2) Part 2

function HYBRID2EVIDENCESUBMISSION(SubmitEvidence, CID, Hash(Data), U, σ)
  Upon receiving (**SubmitEvidence**, **CID**, **Hash(Data)**, U, σ) **from** E
  Assert the current time T.
  Record (SubmitEvidence, CID, Hash(Data), U, Firststep, T) in Mem.
  Send (SubmitEvidence, CID, Hash(Data), U, σ, Firststep) to S.
  Upon receiving (**SubmitEvidence**, **CID′**, **Hash(Data)**, U) **from** S
  if record with CID′ exists in Mem and step is Firststep **then**
    Extract (SubmitEvidence, CID, Hash(Data), U, Firststep) from Mem where CID′ = CID.
    Extract (M_{prev}, block_number, $H(T_{\text{block}})$) from Mem where CID′ = CID
    $M_{\text{case}} \leftarrow \text{Hash}(\text{SubmitEvidence}\|\text{CID}\|\text{Hash(Data}\|U\|\text{"Result"}\|H(T_{\text{block}}))$ ▷ Generate Merkle root
    Extract the set `CorruptedUsers` from `Mem`
    if $U \in$ `CorruptedUsers` **then**
      Abort
    Extract ($\text{Roles}_{\text{stored}}$, StoredStage) from Mem where CID′ = CID
    if $U' \notin \text{Roles}_{\text{stored}}$ **then**
      abort
    else if CurrentStage ≠ StoredStage **then**
      Result ← "fail" ▷ Invalid stage
    else
      Role ← `lookupRole`(U', $\text{Roles}_{\text{stored}}$)
      Rights ← `lookupRights`(CurrentStage, Role)
      if "CreateCase" ∉ Rights **then**
        Result ← "fail" ▷ Role not permitted to create case
      else
        Result ← "Successful"
    if Result = "Successful" **then**
      if Hash(Data) is a list of parent tokens **then**
        Concatenate all K_i in Hash(Data) to get T_{concat}
        Hash(Data) ← Hash($T_{\text{concat}}\|T$)
    Update Mem with (Result, CID, Hash(Data), U, M_{case}, T_{block}, T, Finalstep)
    Record (Result, SubmitEvidenc, CID, U, M_{case}, block_number, T_{block}) in *SimProvenance*
    Send (Result, SubmitEvidence, CID, Hash(Data), U, σ, block_number, $H(T_{\text{block}})$, Finalstep) to S
  else
    abort
  Upon receiving (**Result**, **SubmitEvidence**, **CID**, **Hash(Data)**, U) **from** S
  if record with CID′ exists in Mem **then**
    Extract (Result, SubmitEvidence, CID, Hash(Data), U, σ, T, Step) from Mem where CID′ = CID
    if Step = Finalstep **then**
      Send (Result, SubmitEvidence, CID, Hash(Data), U, σ, T) to E
    else
      abort
  else
    abort

Hybrid 2 ForensiBlock (Π_2) Part 3

function HYBRID2STAGEUPDATE(UpdateStage, CID, NewStage, U, σ)
  Upon receiving (**UpdateStage**, **CID**, **NewStage**, U, σ) **from** E
  Assert current time T
  Record (UpdateStage, CID, NewStage, U, Firststep, T) in Mem
  Send (UpdateStage, CID, NewStage, U, σ, Firststep) to S
  Upon receiving (**UpdateStage**, **CID′**, **NewStage**, U) **from** S
  if record with CID′ exists in Mem and step is Firststep **then**
    Extract (UpdateStage, CID, NewStage, U, Firststep) from Mem where CID′ = CID
    Extract the set `CorruptedUsers` from `Mem`
    if $U \in$ `CorruptedUsers` **then**
      Abort
    Extract ($\text{Roles}_{\text{stored}}$, StoredStage) from Mem where CID′ = CID
    Set CurrentStage ← StoredStage
    if $U \notin \text{Roles}_{\text{stored}}$ **then**
      Result ← "fail"
    else
      Role ← `lookupRole`(U, $\text{Roles}_{\text{stored}}$)
      Rights ← `lookupRights`(CurrentStage, Role)
      if "UpdateStage" ∉ Rights **then**
        Result ← "fail"
      else
        Result ← "Successful"
        Update Mem with stage where CID to NewStage
    Extract (M_{prev}, block_number, $H(T_{\text{block}})$) from Mem where CID′ = CID
    $M_{\text{stage}} \leftarrow \text{Hash}(\text{UpdateStage}\|\text{CID}\|\text{NewStage}\|U\|\text{"Result"}\|H(T_{\text{block}}))$
    Update Mem with (Result, CID, M_{stage}, NewStage, U, T, Finalstep)
    Record (Result, UpdateStage, CID, NewStage, M_{stage}, block_number, T_{block}, U) in *SimProvenance*
    Send (Result, UpdateStage, CID, NewStage, U, σ, Finalstep) to S
  else
    abort
  Upon receiving (**Result**, **UpdateStage**, **CID**, **NewStage**, U) **from** S
  if record with CID′ exists in Mem **then**
    Extract (Result, UpdateStage, CID, NewStage, U, σ, T, Step) from Mem where CID′ = CID
    if Step = Finalstep **then**
      Send (Result, UpdateStage, CID, NewStage, U, σ, T) to E
    else
      abort
  else
    abort

Hybrid 2 ForensiBlock (Π_2) Part 4

function HYBRID2PROVENANCEVERIFICATION(VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$)
  Upon receiving (**VerifyProvenance**, **CID**, $M_{\text{computed case}}, U, \sigma$) **from** E
  Assert current time T
  Record (VerifyProvenance, CID, $M_{\text{computed case}}, U$, Firststep, T) in Mem
  Send (VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$, Firststep) to S
  Upon receiving (**VerifyProvenance**, **CID**, $M_{\text{computed case}}, U$) **from** S
  if record with CID′ exists in Mem and Step is Firststep **then**
    Extract the set `CorruptedUsers` from `Mem`
    if $U \in$ `CorruptedUsers` **then**
      Abort
    Extract ($\text{Roles}_{\text{stored}}$, StoredStage) from Mem where CID′ = CID
    if $U \notin \text{Roles}_{\text{stored}}$ **then**
      abort
    else if CurrentStage ≠ StoredStage **then**
      Result ← "fail" ▷ Invalid stage
    else
      Role ← `lookupRole`(U, $\text{Roles}_{\text{stored}}$)
      Rights ← `lookupRights`(CurrentStage, Role)
      if "SubmitEvidence" ∉ Rights **then**
        Result ← "fail"
      else
        Result ← "Successful"
    if Result = "Successful" **then**
      Extract M_{case} from Mem where CID = CID′
      if $M_{\text{case}} = M_{\text{computed case}}$ **then**
        Result ← Result‖"-Successful"
      else
        Result ← Result‖"-fail"
    Extract (M_{case}, block_number, $H(T_{\text{block}})$) from Mem where CID′ = CID
    $M_{\text{case}} \leftarrow \text{Hash}(M_{\text{case}}\|\text{"VerifyProvenance"}\|\text{"Result"}\|H(T_{\text{block}}))$
    Update Mem with (Result, CID, M_{case}, U, T, Finalstep)
    Record (Result, VerifyProvenance, CID, M_{Result}, block_number, T_{block}, U) in *SimProvenance*
    Send (Result, VerifyProvenance, CID, U, σ, block_number, T_{block}, Finalstep) to S
  else
    abort
  Upon receiving (**Result**, **VerifyProvenance**, **CID**, $M_{\text{computed case}}, U$) **from** S
  if record with CID′ exists in Mem **then**
    Extract (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma, T$, Step) from Mem where CID′ = CID
    if Step = Finalstep **then**
      Send (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma, T$) to E
    else
      abort
  else
    abort

Hybrid 2 ForensiBlock (Π_2) Part 5

function HYBRID2ACCESS(Access, AccessType, CID, U, σ)
  Upon receiving (**Access**, **AccessType**, **CID**, U, σ) **from** E
  Assert the current time T.
  Record (Access, AccessType, CID, U, Firststep, T) in Mem.
  Send (Access, AccessType, CID, U, σ, Firststep) to S.
  Upon receiving (**Access**, **AccessType**, **CID**, U) **from** S
  Extract the set `CorruptedUsers` from `Mem`
  if $U \in$ `CorruptedUsers` **then**
    Abort
  if record with CID′ exists in Mem and Step is Firststep **then**
    Extract (Access, AccessType, CID, U, Firststep) from Mem where CID′ = CID
    Extract ($\text{Roles}_{\text{stored}}$, StoredStage) from Mem where CID′ = CID
    if $U \notin \text{Roles}_{\text{stored}}$ **then**
      Result ← "Denied"
    else
      Role ← `lookupRole`(U, $\text{Roles}_{\text{stored}}$)
      Rights ← `lookupRights`(StoredStage, Role)
      if AccessType ∉ Rights **then**
        Result ← "Denied"
      else
        Result ← "Accepted"
    Extract (M_{prev}, block_number, $H(T_{\text{block}})$) from Mem where CID′ = CID
    $M_{\text{case}} \leftarrow \text{Hash}(\text{Access}\|\text{AccessType}\|\text{CID}\|U\|\text{Result}\|H(T_{\text{block}}))$
    Update Mem with (Result, Access, CID, U, M_{access}, T, Finalstep)
    Record (Result, Access, AccessType, CID, M_{access}, block_number, T_{block}, U) in *SimProvenance*
    Send (Result, Access, AccessType, CID, U, σ, Finalstep) to S
  else
    abort
  Upon receiving (**Result**, **Access**, **CID**, U) **from** S
  if record with CID′ exists in Mem **then**
    Extract (Result, Access, CID, U, σ, T, Step) from Mem where CID′ = CID.
    if Step = Finalstep **then**
      Send (Result, Access, CID, U, σ, T) to E.
      else
      abort.
    else
    abort.

Algorithm 3.14 Simulator for Hybrid 2 (S_2) Part 1

function SimCaseCreationHybrid2(CreateCase, CID, Roles, U)
  Upon receiving (**CreateCase**, **CID**, **Roles**, U, σ, **Firststep**) **from** (Π_2)
  Record (CreateCase, CID, Roles, U, Firststep, σ) in Mem
  (PK, SK) $\leftarrow$ KeyGen(U)
  Record (PK, SK) in Mem where CID = CID$''$
  σ" := Sign$_{SK}$(CreateCase, CID, Roles, U, σ)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (CreateCase, CID, Roles, U, σ) to $\mathcal{A}$
  Upon receiving (**CreateCase**$'$, **CID**$'$, **Roles**$'$, U', σ''') **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (CreateCase, CID, Roles, U, SK, PK, step, σ") from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(CreateCase$'$, CID$'$, Roles$'$, U', σ''')
  if $\sigma' \neq \sigma$" **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if `ForgeryDetected` = `true` **then**
    abort
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (CreateCase, CID, Roles, U, `ForgeryDetected`) to (Π_2)
  else
    abort
  Upon receiving (**Result**, **CreateCase**, **CID**, **Roles**, $U, \sigma, , T$, **Finalstep**) **from** (Π_2
  Update Mem with (Result, CreateCase, CID, Roles, U, σ, T, Finalstep)
  Extract (SK) from Mem where CID$''$ = CID
  σ" := Sign$_{SK}$(Result, CreateCase, CID, Roles, U, σ, T)

  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Result, CreateCase, CID, Roles, U, σ) to $\mathcal{A}$
  Upon receiving (**Result**$'$, **CreateCase**$'$, **CID**$'$, **Roles**$'$, σ''', U', T') **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (Result, CreateCase, CID, Roles, U, σ, σ") from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Result$'$, CreateCase$'$, CID$'$, Roles$'$, U', σ''', T')
  if $\sigma' \neq \sigma$" **then**
    Detect Forgery
  if step = Finalstep **then**
    `PendingSessions.remove(entry)`
    Send (Result, CreateCase, CID, Roles, U) to (Π_2)
  else
    abort

Simulator for Hybrid 2 (S_2) Part 2

function SimEvidenceSubmissionHybrid3(SubmitEvidence, CID, Hash(Data), U)
  Upon receiving (**SubmitEvidence**, **CID**, **Hash(Data)**, U, σ, **Firststep**) **from** (Π_2)
  Record (SubmitEvidence, CID, Hash(Data), U, Firststep, σ) in Mem
  (PK, SK) $\leftarrow$ KeyGen(U)
  Record (PK, SK) in Mem where CID = CID$''$
  σ" := Sign$_{SK}$(CreateCase, CID, Roles, U, σ)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (SubmitEvidence, CID, Hash(Data), U, σ) to $\mathcal{A}$
  Upon receiving (**SubmitEvidence**$'$, **CID**$'$, **Hash(Data)**$'$, U', σ''') **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (SubmitEvidence, CID, Hash(Data), U, step, SK, PK, σ) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(SubmitEvidence$'$, CID$'$, Hash(Data)$'$, U', σ''')
  if $\sigma' \neq \sigma$" **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (SubmitEvidence, CID, Hash(Data), U) to (Π_2)
  else
    abort
  Upon receiving (**Result**, **SubmitEvidence**, **CID**, **Hash(Data)**, $U, \sigma, , T$, **Finalstep**) **from** (Π_2)
  Update Mem with (Result, SubmitEvidence, CID, Hash(Data), $U, ,$ Finalstep)
  Extract (SK) from Mem where CID$''$ = CID
  σ" := Sign$_{SK}$(Result, SubmitEvidence, CID, Hash(Data), U, σ)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Result, SubmitEvidence, CID, Hash(Data), U, σ, T) to $\mathcal{A}$
  Upon receiving (**Result**$'$, **SubmitEvidence**$'$, **CID**$'$, **Hash(Data)**$'$, U', σ''', T') **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (Result, SubmitEvidence, CID, Hash(Data), U, Finalstep, σ, σ") from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Result$'$, SubmitEvidence$'$, CID$'$, Hash(Data)$'$, U', σ''', T')
  if $\sigma' \neq \sigma$" **then**
    Detect Forgery
  if step = Finalstep **then**
    `PendingSessions.remove(entry)`
    Send (Result, SubmitEvidence, CID, Hash(Data), U) to (Π_2)
  else
    abort

Simulator for Hybrid 2 (S_2) Part 3

function SimStageUpdateHybrid3(UpdateStage, CID, NewStage, U, Firststep)
  Upon receiving (**UpdateStage**, **CID**, **NewStage**, U, σ, **Firststep**) **from** (Π_2)
  Record (UpdateStage, CID, NewStage, U, Firststep, σ) in Mem
  (PK, SK) $\leftarrow$ KeyGen(U)
  Record (PK, SK) in Mem where CID = CID$''$
  σ" := Sign$_{SK}$(CreateCase, CID, Roles, U)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (UpdateStage, CID, NewStage, U, σ) to $\mathcal{A}$
  Upon receiving (**UpdateStage**$'$, **CID**$'$, **NewStage**$'$, U', σ') **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (UpdateStage, CID, NewStage, U, step, SK, PK, σ) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(UpdateStage$'$, CID$'$, NewStage$'$, U', T')
  if $\sigma' \neq \sigma$" **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (UpdateStage, CID, NewStage, U) to (Π_2)
  else
    abort
  Upon receiving (**Result**, **UpdateStage**, **CID**, **NewStage**, $U, \sigma, , T$, **Finalstep**) **from** (Π_2)
  Update Mem with (Result, UpdateStage, CID, NewStage, U, Finalstep, σ)
  Extract (SK) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Result, UpdateStage, CID, NewStage, U, σ, T, Finalstep)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Result, UpdateStage, CID, NewStage, U, σ, T) to $\mathcal{A}$
  Upon receiving (**Result**$'$, **UpdateStage**$'$, **CID**$'$, **NewStage**$'$, U', σ''', T) **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (Result, UpdateStage, CID, NewStage, U, SK, PK, Finalstep, σ, σ") from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Result$'$, UpdateStage$'$, CID$'$, NewStage$'$, U', σ''', T)
  if $\sigma' \neq \sigma$" **then**
    Detect Forgery
  if step = Finalstep **then**
    `PendingSessions.remove(entry)`
    Send (Result, UpdateStage, CID, NewStage, U) to (Π_2)
  else
    abort

Simulator for Hybrid 2 (S_2) Part4

function SimProvenanceVerificationHybrid3(VerifyProvenance, CID, $M_{\text{computed case}}, U$)
  Upon receiving (**VerifyProvenance**, **CID**, $M_{\text{computed case}}, U, \sigma$, **Firststep**) **from** (Π_2)
  Record (VerifyProvenance, CID, $M_{\text{computed case}}, U$, Firststep, σ) in Mem
  (PK, SK) $\leftarrow$ KeyGen(U)
  Record (PK, SK) in Mem where CID = CID$''$
  σ" := Sign$_{SK}$(CreateCase, CID, Roles, U)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$, Firststep) to $\mathcal{A}$
  Upon receiving (**VerifyProvenance**$'$, **CID**$'$, $M'_{\text{computed case}}, U', \sigma'$, **Firststep**) **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (VerifyProvenance, CID, $M_{\text{computed case}}, U$, SK, PK, step, σ) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(VerifyProvenance$'$, CID$'$, $M'_{\text{computed case}}, U', \sigma'''$)
  if $\sigma' \neq \sigma$" **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if `ForgeryDetected` = `true` **then**
    abort
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (VerifyProvenance$'$, CID$'$, $M'_{\text{computed case}}, U'$) to ($\Pi_2$)
  else
    abort
  Upon receiving (**Result**, **VerifyProvenance**, **CID**, $M_{\text{computed case}}, U, \sigma, , T$, **Finalstep**) **from** (Π_2)
  Extract (SK) from Mem where CID$''$ = CID
  Update Mem with (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$, Finalstep)
  $\sigma :=$ Sign$_{SK}$(Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, T$)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$) to $\mathcal{A}$
  Upon receiving (**Result**$'$, **VerifyProvenance**$'$, **CID**$'$, $M'_{\text{computed case}}, U', \sigma'$) **from** $\mathcal{A}$
  Extract CID" from `PendingSessions`
  Extract (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U, \sigma$, SK, PK, Finalstep) from Mem where CID$''$ = CID
  $\sigma :=$ Sign$_{SK}$(Result$'$, VerifyProvenance$'$, CID$'$, $M'_{\text{computed case}}, U', T'$)
  if $\sigma' \neq \sigma''$ **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if `ForgeryDetected` = `true` **then**
    abort
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (Result, VerifyProvenance, CID, $M_{\text{computed case}}, U$) to ($\Pi_2$)
  else
    abort

Simulator for Hybrid 2 (S_2) Part5

function SimAccessHybrid3(Access, AccessType, CID, U)
  Upon receiving (**Access**, **AccessType**, **CID**, U, σ, **Firststep**) **from** Π_2
  Record (Access, AccessType, CID, U, Firststep, σ) in Mem
  (PK, SK) $\leftarrow$ KeyGen(U)
  Record (PK, SK) in Mem where CID = CID$''$
  σ" := Sign$_{SK}$(Access, AccessType, CID, U, σ)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Access, AccessType, CID, U, σ) to $\mathcal{A}$.
  Upon receiving (**Access**$'$, **AccessType**$'$, **CID**$'$, U', σ') **from** $\mathcal{A}$
  Extract CID$''$ $\leftarrow$ CID from `PendingSessions`
  Extract (Access, AccessType, CID, U, step, SK, PK, σ) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Access$'$, AccessType$'$, CID$'$, U', σ)
  if $\sigma' \neq \sigma''$ **then**
    `ForgeryDetected` $\leftarrow$ `true`
  else
    `ForgeryDetected` $\leftarrow$ `false`
  if `ForgeryDetected` = `true` **then**
    abort
  if step = Firststep **then**
    `PendingSessions.remove(entry)`
    Send (Access, AccessType, CID, U) to Π_2
  else
    abort
  Upon receiving (**Result**, **Access**, **CID**, U, σ, T, **Finalstep**) **from** Π_2
  Update Mem with (Result, Access, CID, U, σ, Finalstep) in Mem where CID$'$ = CID
  Extract (SK) from Mem where CID$''$ = CID
  σ" := Sign$_{SK}$(Result, Access, CID, U', σ, T)
  `PendingSessions` $\leftarrow$ [(CID)]
  Send (Result, Access, CID, U, σ) to $\mathcal{A}$
  Upon receiving (**Result**$'$, **Access**$'$, **CID**$'$, U', σ', T') **from** $\mathcal{A}$
  Extract CID$''$ $\leftarrow$ CID from `PendingSessions`
  if record with CID$''$ does not exist in Mem **then**
    abort
  Extract (Result, Access, CID, U, σ, Finalstep) from Mem where CID$''$ = CID
  $\sigma' :=$ Sign$_{SK}$(Result$'$, Access$'$, CID$'$, U', σ', T')
  if $\sigma' \neq \sigma$" **then**
    Detect Forgery
  if step = Finalstep **then**
    `PendingSessions.remove(entry)`
    Send (Result, Access, AccessType, CID, U) to Π_2
  else
    abort

Final take away from Forensiblock's UC

Notation	Meaning
π	Real-world protocol
φ	Ideal functionality
A	Adversary
S	Simulator for the ideal world
E	Environment trying to distinguish
exec·	Execution transcript or environment output

1) $\pi 3 \approx \pi, \pi 0 = \phi$
2) $Pr[E(\pi 1, A) = 1] - Pr[E(\pi 0, S0) = 1] \leq negl1(n)$
3) $|Pr[E(\pi 2, A) = 1] - Pr[E(\pi 1, A) = 1]| \leq negl2(n)$
4) $|Pr[E(\pi 3, A) = 1] - Pr[E(\pi 2, A) = 1]| \leq negl3(n)$
5) **$|Pr[E(\pi, A) = 1] - Pr[E(\phi, S) = 1]| \leq negl1(n) + negl2(n) + negl3(n) = negl(n)$**
6) **$exec_{\pi,A,E} \approx exec_{\phi,S,E}$**

Result
- ❑We formally prove that even a powerful adversary cannot distinguish between the real-world and the ideal-world executions.

Result
- ❑The UC framework guarantees that if ForensiBlock is secure in isolation, it remains secure even when composed with other UC-secure protocols.

What if there are more than one agency?

Crosschain

- Mechanisms used for cross-chain interaction
  - Notary schemes
  - Relay chains
- Frameworks
  - Vassago (Han, Rui, et al)[1] , Synergychain(Chang, Jian, et al)[2] , ARC(Zhang, Shaobo, et al) [3]
- Limitations
  - Not designed for digital forensic purposes
  - Provenance capture or query
  - Access control
  - Security analysis
  - Architectural scalability

[1] Han, Rui, et al. "Vassago: Efficient and authenticated provenance query on multiple blockchains." 2021 40th International Symposium on Reliable Distributed Systems (SRDS). IEEE, 2021.
[2] Chang, Jian, et al. "SynergyChain: A multichain-based data-sharing framework with hierarchical access control." IEEE Internet of Things Journal 9.16 (2021): 14767-14778.
[3] Zhang, Shaobo, et al. "ARC: an asynchronous consensus and relay chain-based cross-chain solution to consortium blockchain." 2022 IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/2022 IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, 2022.

ForensiCross [30]

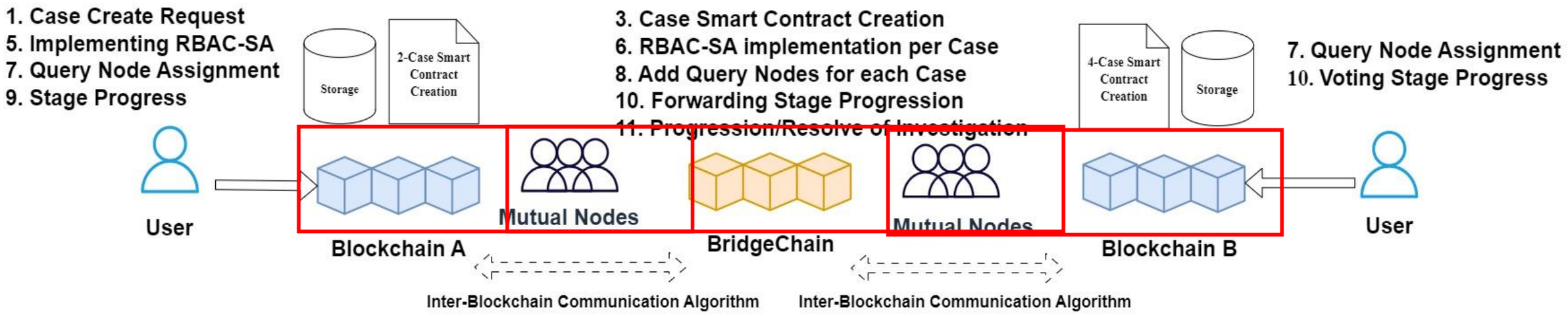
Objectives

- Enable secure collaboration across independent forensic blockchains
- Preserve provenance integrity during cross-chain communication
- Design scalable and interoperable mechanisms for multi-blockchain environments

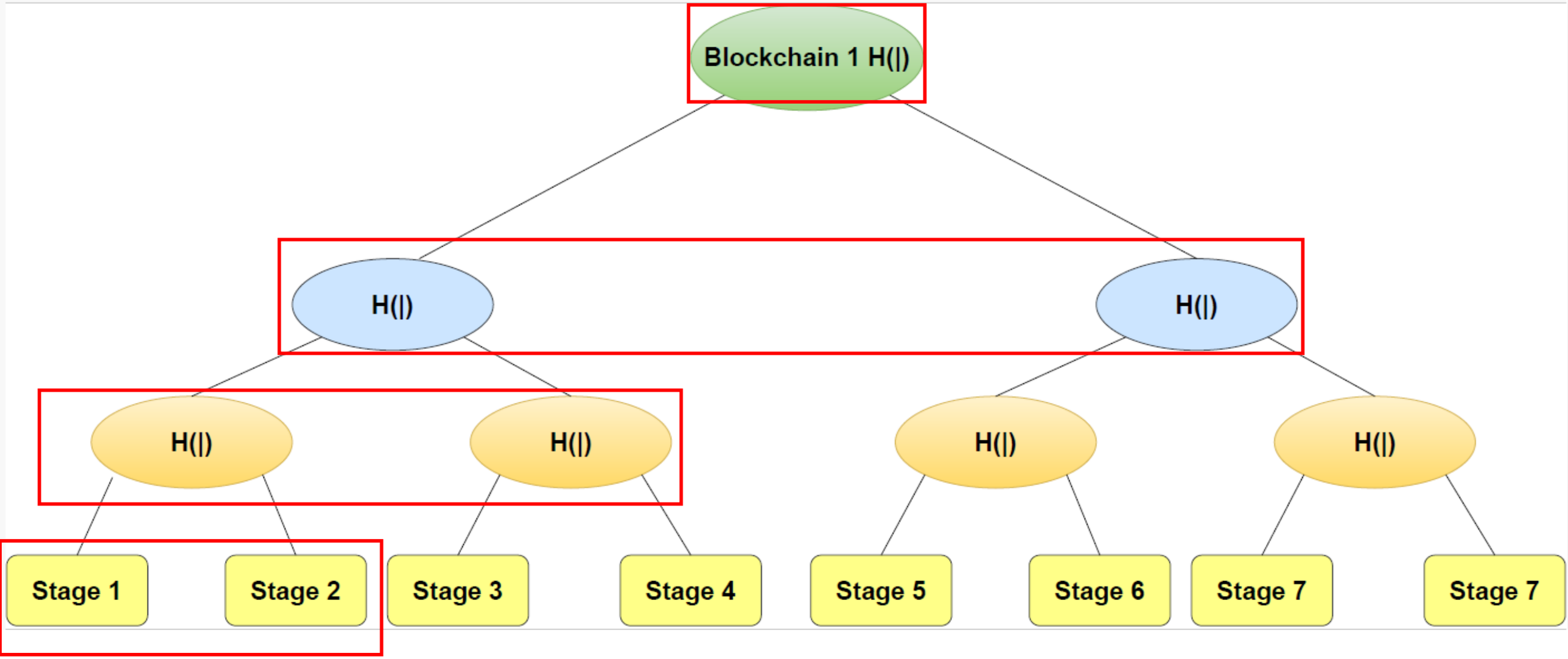
Research Questions

- How can multiple forensic blockchains interoperate securely and efficiently?
- How can provenance be verified as data moves between blockchains?
- What are the key design considerations as the number of participating blockchains increases?

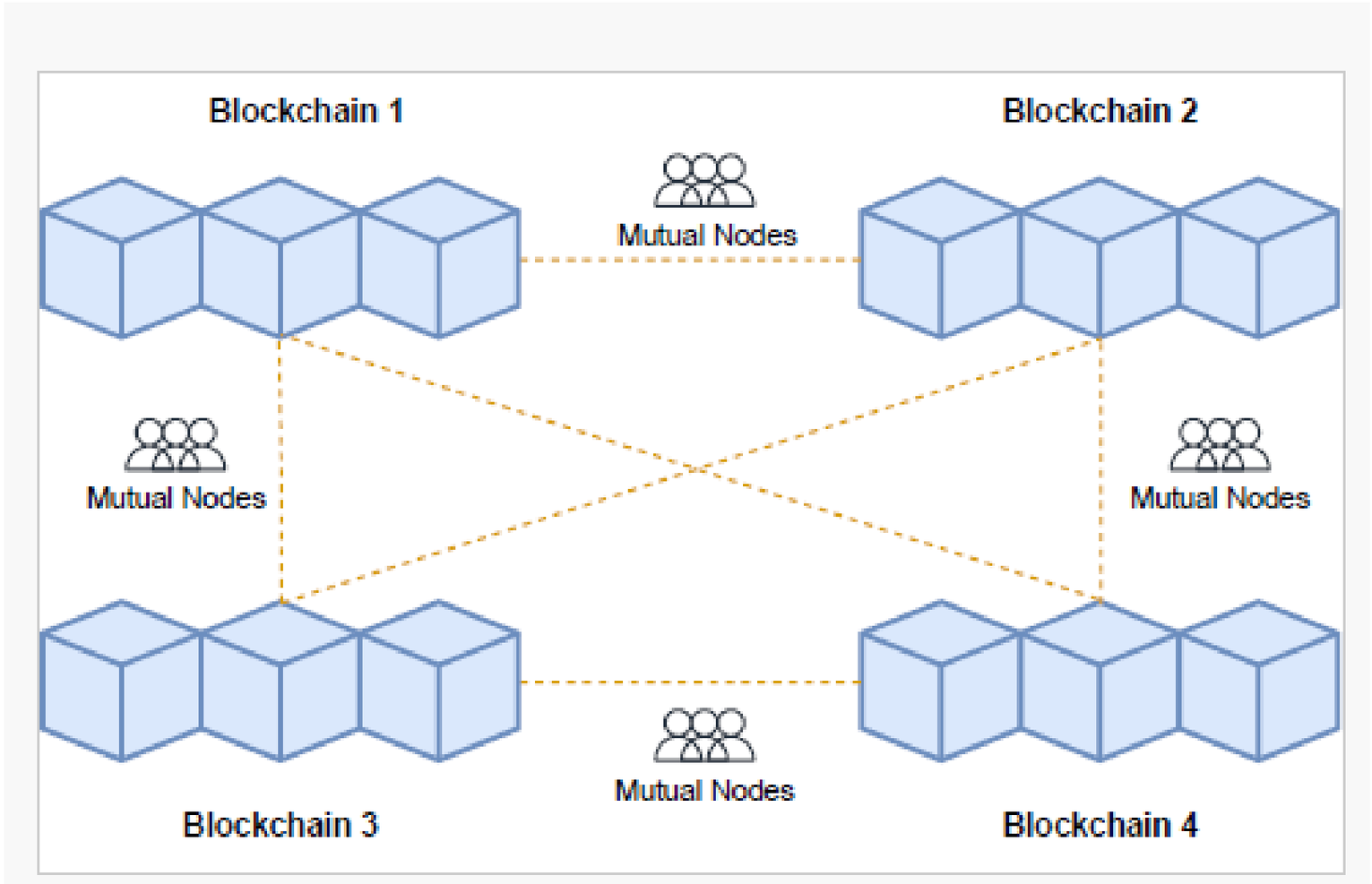
ForensiCross Architecture



Provenance Record Verification

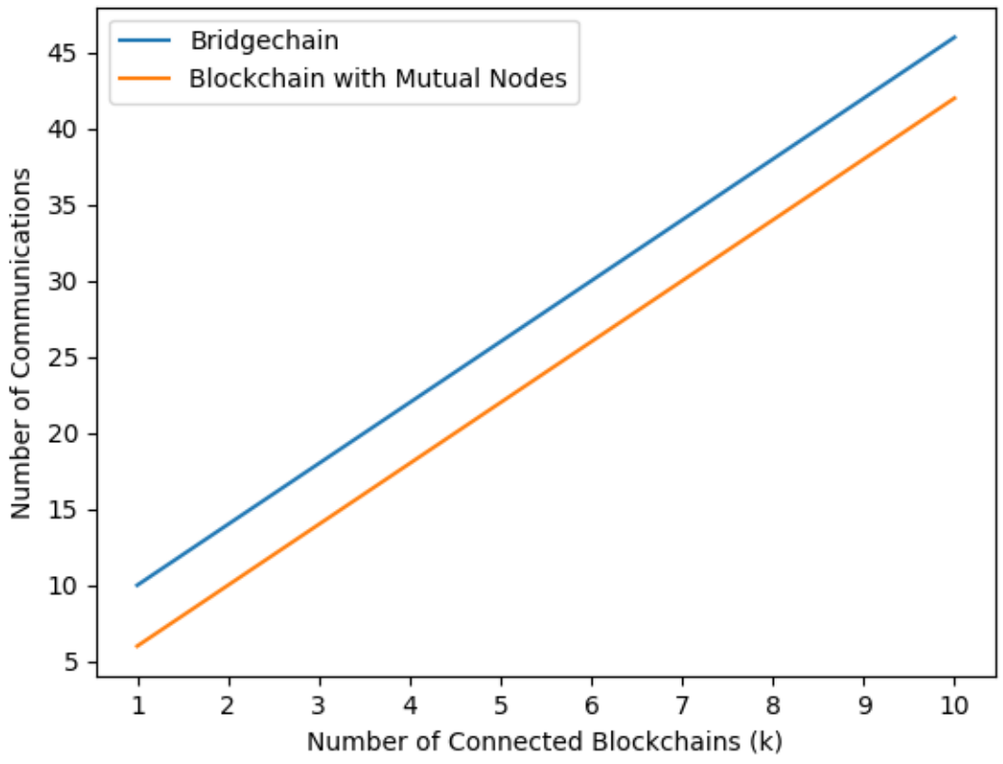


Simple Notary Alternative

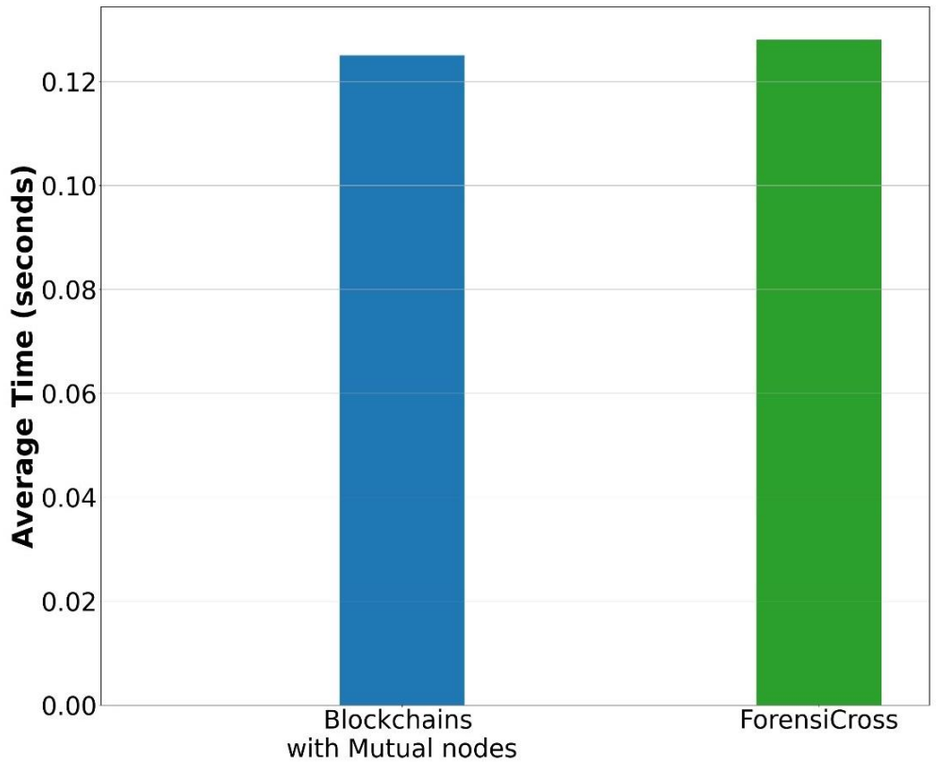


Communication in Brdigdechain VS simple Notary Methods

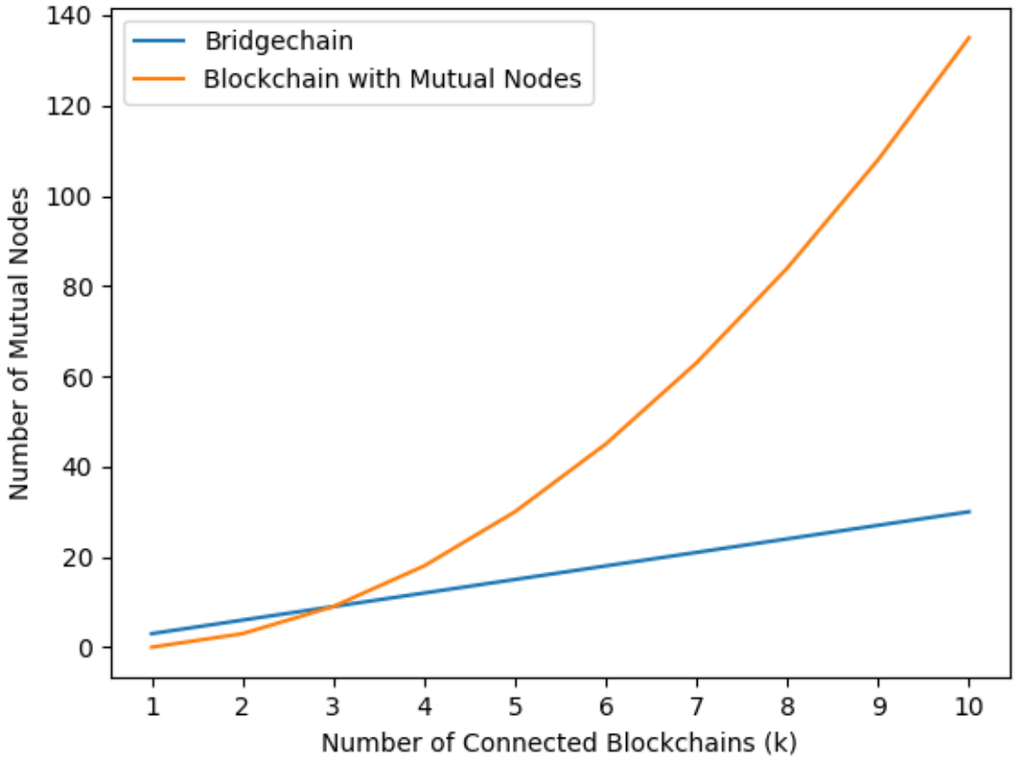
Number of Communication requests with increased number of blockchains



Average time for Communication requests



Brdigdechain vs. Simple Notary Methods



- ❑ Number of mutual nodes increased
- ❑ No provenance tracking
- ❑ No stage synchronization
- ❑ Routing complexity

Analysis of the Bridgechain Design

- Ensures mutual nodes don’t dominate Bridgechain consensus.
- Guarantees odd number of nodes avoids tie votes.

Variable	Definition
k	Number of blockchains
n	Nodes per blockchain
m	Total nodes in Bridgechain
Bi	Mutual nodes between blockchain i and the Bridgechain
x ∈ ℤ	Parameter to generalize mutual nodes as 2x + 1

Constraints for Security & Scalability

- Minimum mutual nodes for each blockchain–Bridgechain link: $|B_i|=2x+1,\ x\geq 1$
- **Total mutual nodes (all blockchains)**
- $b_{total}=k\cdot(2x+1)$
- **Bridgechain node requirement for consensus**
- $m\geq 2(2x+1)k+1$

Open Problems for Future Work

- ❑ Integration with AI/ML tools
- ❑ Alternatives needs for digital forensics relate to Federated Learning
- ❑ Automated Tools for Formal Verification for Blockchain based Provenance systems
- ❑ Intermediary Analysis for Crosschain

Contributions & Conclusion

- ✓ Defining Requirements for Blockchain-Based Data Provenance
- ✓ Designing ForensiBlock: A Blockchain Framework for Digital Forensics
- ✓ Conducting UC Security Analysis
- ✓ Establishing Cross-Chain Connectivity

Thank you

References

[1] MLAKumar, Narendra, et al. "Blockchain Technology in Supply Chain Management: Innovations, Applications, and Challenges." *Telematics and Informatics Reports* (2025): 100204.

[2] Gurupatham, Twinkle Geojini, Yesubai Rubavathi Charles, and Ramnath Muthusamy. "Revolutionizing agricultural supply chain management with blockchain-based IoT: Improving traceability, efficiency and sustainability." *AIP Conference Proceedings*. Vol. 3237. No. 1. AIP Publishing LLC, 2025.

[3] Kulothungan, Vikram. "Using Blockchain Ledgers to Record the AI Decisions in IoT." (2025).

[4] Haq, Rashid Ul, et al. "Transchain: Blockchain-Based Management of Allografts for Enhancing Data Provenance." *IEEE Access* (2025).

[5] Mbimbi, Butrus, David Murray, and Michael Wilson. "Preserving Whistleblower Anonymity Through Zero-Knowledge Proofs and Private Blockchain: A Secure Digital Evidence Management Framework." *Blockchains* 3.2 (2025): 7.

[6] Javaid, M. N. Aman, and B. Sikdar. "BlockPro: Blockchain based data provenance and integrity for secure IoT environments.“ In Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems, 13–18

[7] Aravind Ramachandran and Murat Kantarcioglu. "Smartprovenance: A distributed, blockchain based dataprovenance system.“ In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, 35–42.

[8] Reagan Hoopes, Hamilton Hardy, Min Long, and Gaby G Dagher."SciLedger: A Blockchain-based Scientific Workflow Provenance and Data Sharing Platform.“ In 2022 IEEE 8th International Conference on Collaboration and Internet Computing (CIC). IEEE, 125–134.

[9] Raiane Coelho, Regina Braga, José Maria David, Fernanda Campos, and Victor Ströele. "Blockflow: Trust in scientific provenance data."In Anais do XIII Brazilian eScience Workshop. SBC.

[10] Pinchen Cui, Julie Dixon, Ujjwal Guin, and Daniel Dimase. "A blockchain-based framework for supply chain provenance.“ IEEE Access 7 (2019), 157113–157125.

[11] Deepak Tosh, Sachin Shetty, Xueping Liang, Charles Kamhoua, and Laurent L Njilla. "Data provenance in the cloud: A blockchain-based approach.“ IEEE Consumer Electronics Magazine 8, 4 (2019), 38–44.

[12] Dane Troyer, Justin Henry, Hoda Maleki, Gokila Dorai, Bethany Sumner, Gagan Agrawal, and Jon Ingram. "Privacy-Preserving Framework to Facilitate Shared Data Access for Wearable Devices."

In 2021 IEEE International Conference on Big Data (Big Data). IEEE, 2583–2592.

[13] Shancang Li, Tao Qin, and Geyong Min. "Blockchain-based digital forensics investigation framework in the internet of things and social systems."IEEE Transactions on Computational Social Systems 6, 6 (2019), 1433–1441.

[14]Mamun Ahmed, Akash Roy Pranta, Mst Fahmida Akter Koly, Farjana Taher, and Mohammad Asaduzzaman Khan. "Using IPFS and Hyperledger on Private Blockchain to Secure the Criminal Record System.“ European Journal of Information Technologies and Computer Science 3, 1 (2023), 1–6.

References

[15] Andrey Demichev, Alexander Kryukov, and Nikolai Prikhodko. 2018. The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform. In 2018 Ivannikov Ispras Open Conference (ISPRAS). IEEE, 131–136.

[16] Dang, Tran Khanh, and Thu Anh Duong. "An effective and elastic blockchain-based provenance preserving solution for the open data." *International Journal of Web Information Systems* 17.5 (2021): 480-515.

[17] Li, Shancang, Tao Qin, and Geyong Min. "Blockchain-based digital forensics investigation framework in the internet of things and social systems." *IEEE Transactions on Computational Social Systems* 6.6 (2019): 1433-1441.

[18] Borse, Yogita, et al. "Advantages of blockchain in digital forensic evidence management." *Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST2021)*. 2021.

[19] Ahmed, Mamun, et al. "Using ipfs and hyperledger on private blockchain to secure the criminal record system." *European Journal of Information Technologies and Computer Science* 3.1 (2023): 1-6.

[20] Zhang, Shaobo, et al. "ARC: an asynchronous consensus and relay chain-based cross-chain solution to consortium blockchain." *2022 IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/2022 IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2022.

[21] Han, Rui, et al. "Vassago: Efficient and authenticated provenance query on multiple blockchains." *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2021.

[22] Liang, Xueping, et al. "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability." *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017.

[23] Akbarfam, Asma Jodeiri, and Hoda Maleki. “SOK: Blockchain for Provenance”.VLDB 2024 - Sixth International Workshop on Foundations and Applications of Blockchain(FAB’24)

[24] Imrankhan, M., et al. "Blockchain Based Digital Forensic Data for Unbreakable Trust and Security." *2024 International Conference on System, Computation, Automation and Networking (ICSCAN)*. IEEE, 2024

[25] Chang, Jian, et al. "SynergyChain: A multichain-based data-sharing framework with hierarchical access control." *IEEE Internet of Things Journal* 9.16 (2021): 14767-14778.

[26] Zhang, Shaobo, et al. "ARC: an asynchronous consensus and relay chain-based cross-chain solution to consortium blockchain." *2022 IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/2022 IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2022.

[27] Y. Sun, L. Yi, L. Duan, and W. Wang, “A decentralized cross-chainservice protocol based on notary schemes and hash-locking,” in 2022IEEE International Conference on Services Computing (SCC). IEEE,2022, pp. 152–157

.[28] P. Gaˇzi, A. Kiayias, and D. Zindros, “Proof-of-stake sidechains,” in 2019IEEE Symposium on Security and Privacy (SP), 2019, pp. 139–156.[32] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell,A. Miller, A. Poelstra, J. Tim´on, and P. Wuille, “Enablingblockchain innovations with pegged sidechains,” URL: http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains, vol. 72, pp. 201–224, 2014

[29] Akbarfam, Asma Jodeiri, et al. "Forensiblock: A provenance-driven blockchain framework for data forensics and auditability." *2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2023.

[30] Akbarfam, Asma Jodeiri, Gokila Dorai, and Hoda Maleki. "Secure Cross-Chain Provenance for Digital Forensics Collaboration." *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*. IEEE, 2024.