# Proofs of Storage

SENY KAMARA

MICROSOFT RESEARCH

# Computing as a Service

- Computing is a vital resource
  - Enterprises, governments, scientists, consumers, …
- Computing is manageable at small scales…
  - e.g., PCs, laptops, smart phones
- …but becomes hard to manage at large scales
  - build and manage infrastructure, schedule backups, hardware maintenance, software maintenance, security, trained workforce, …
- Why not outsource it?

# Cloud Services

- Software as a service
  - Gmail, Hotmail, Flickr, Facebook, Office365, Google Docs, …
  - Service: customer makes use of provider applications
  - Customer: consumers & enterprise
- Platform as a service
  - MS SQL Azure, Amazon SimpleDB, Google AppEngine
  - Service: customer makes use of provider's software stack
  - Customer: developers
- Infrastructure as a service
  - Amazon EC2, Microsoft Azure, Google Compute Engine
  - Service: customer makes use of provider's (virtualized) infrastructure
  - Customer: enterprise, developers

# Cloud Advantages

- Providers
  - Monetize spare capacity
- Consumers
  - Convenience: backups, synchronizations, sharing
- Companies
  - Elasticity
  - Can focus on core business
  - Cheaper services

# Cloud Risks

- Risks
  - 100% reliability is impossible
  - Downtime can be costly (startups can go out of business)
- AWS outages
  - December 12th, 2010: EC2 down for 30 mins (Europe)
  - April 21, 2011: storage down for 10-12 hours (N. Virginia)
    - Foursquare, Reddit, Quora, BigDoor and Hootsuite affected
  - August 6th, 2011: storage down for 24 hours (Ireland)
  - August 8th, 2011: network connectivity down for 25 mins (N. Virginia)
    - Reddit, Quora, Netflix and FourSquare affected
  - July 7th, 2012: storage down for few hours (Virginia)
    - Instagram, Netflix, Pinterest affected

Q: is my data still there?

# Outline

- Motivation
- Naïve Solutions
- Overview of Proofs of Storage
- Defining Proofs of Storage
- Designing Proofs of Storage
- Applying Proofs of Storage

Q: is my data still there?

# Digital Signatures/MACs

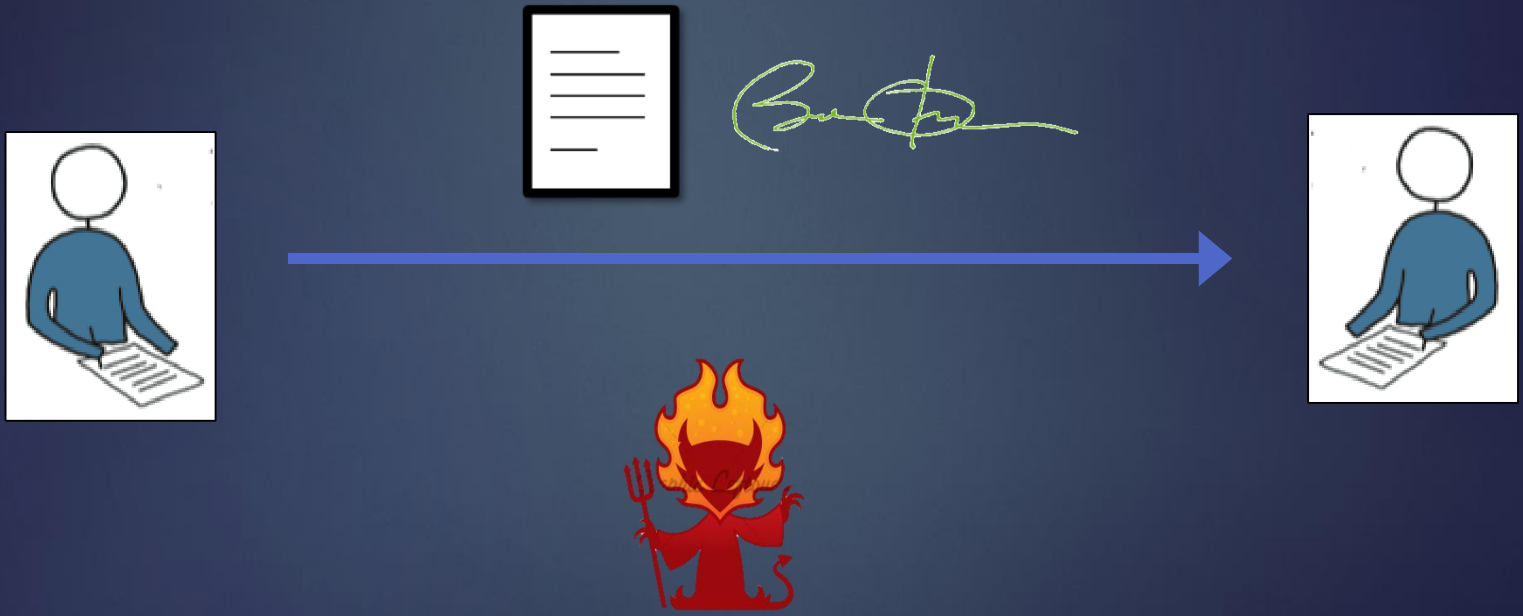
- Signatures
  - $\mathsf{Gen}(1^k) \Longmapsto (sk, vk)$
  - $\mathsf{Sign}(sk, m) \Longmapsto \sigma$
  - $\mathsf{Vrfy}(vk, m, \sigma) \Longmapsto b$

- Message Authentication Codes
  - $\mathsf{Gen}(1^k) \Longmapsto sk$
  - $\mathsf{Tag}(sk, m) \Longmapsto \sigma$
  - $\mathsf{Vrfy}(sk, m, \sigma) \Longmapsto b$

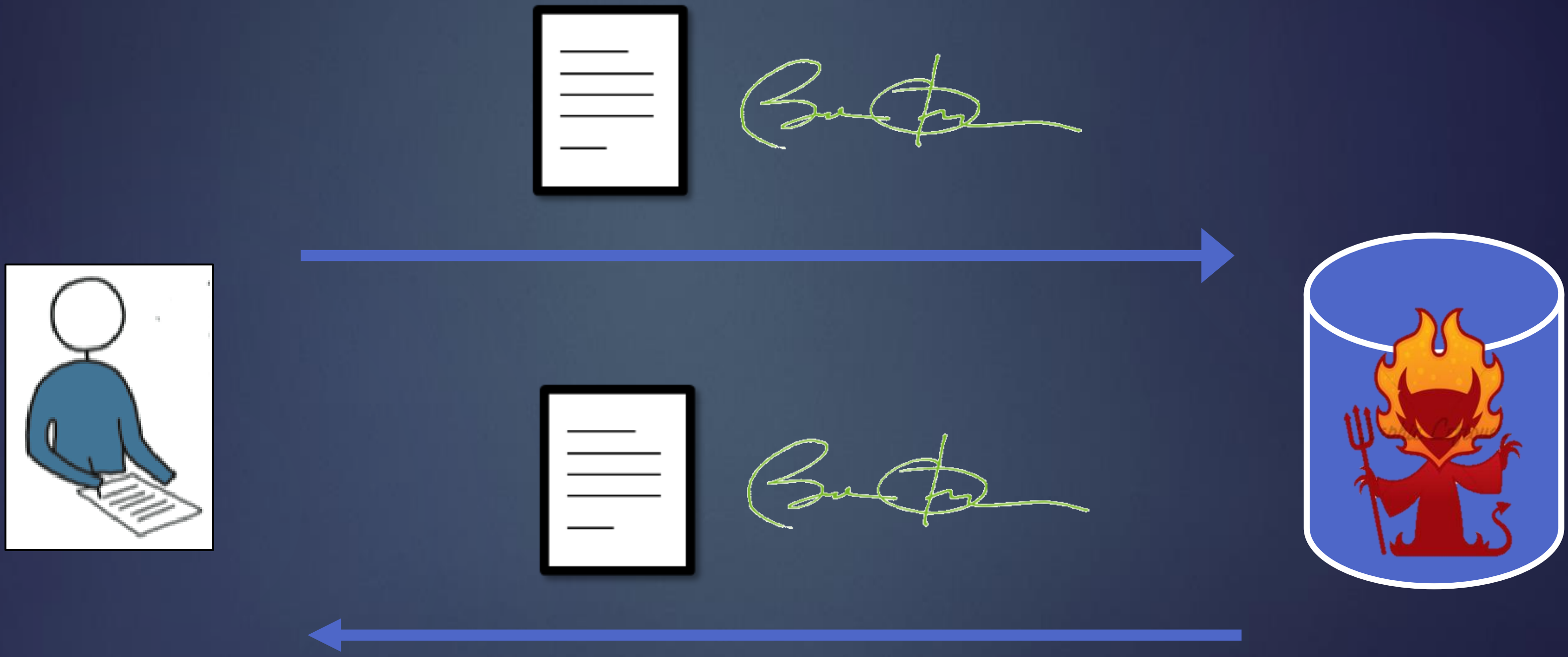
- Security

**UNF**: "given m and σ, no $\mathcal{A}$ can output a valid σ′ for an element m' ≠ m "

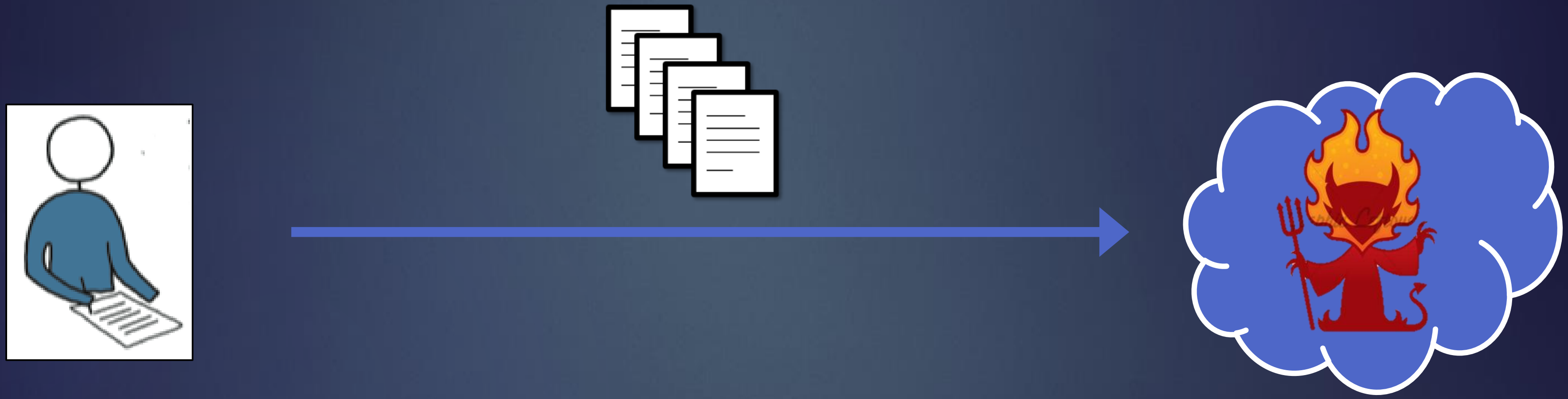
# Communication Channels



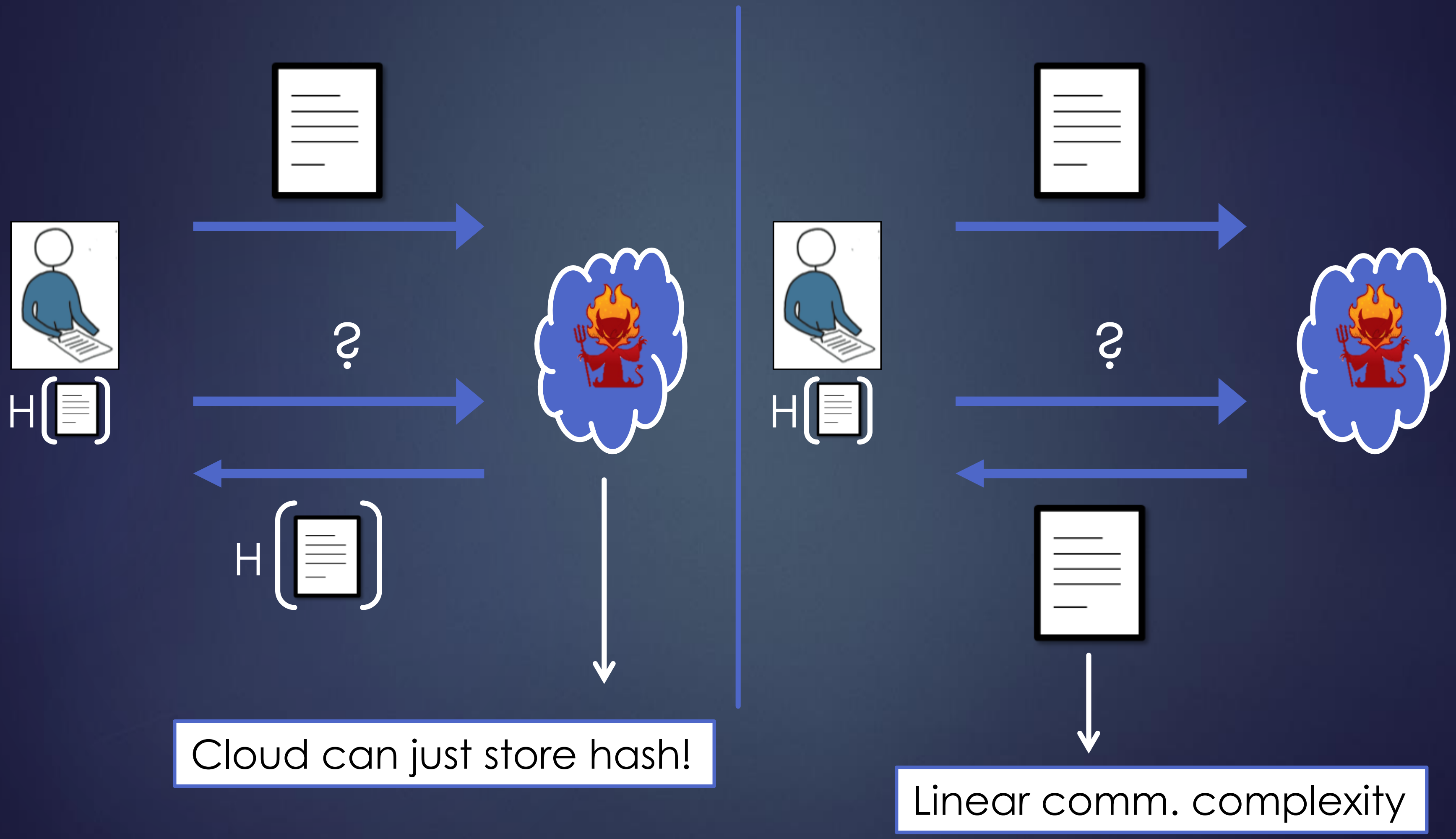
# Local Storage



# Cloud Storage



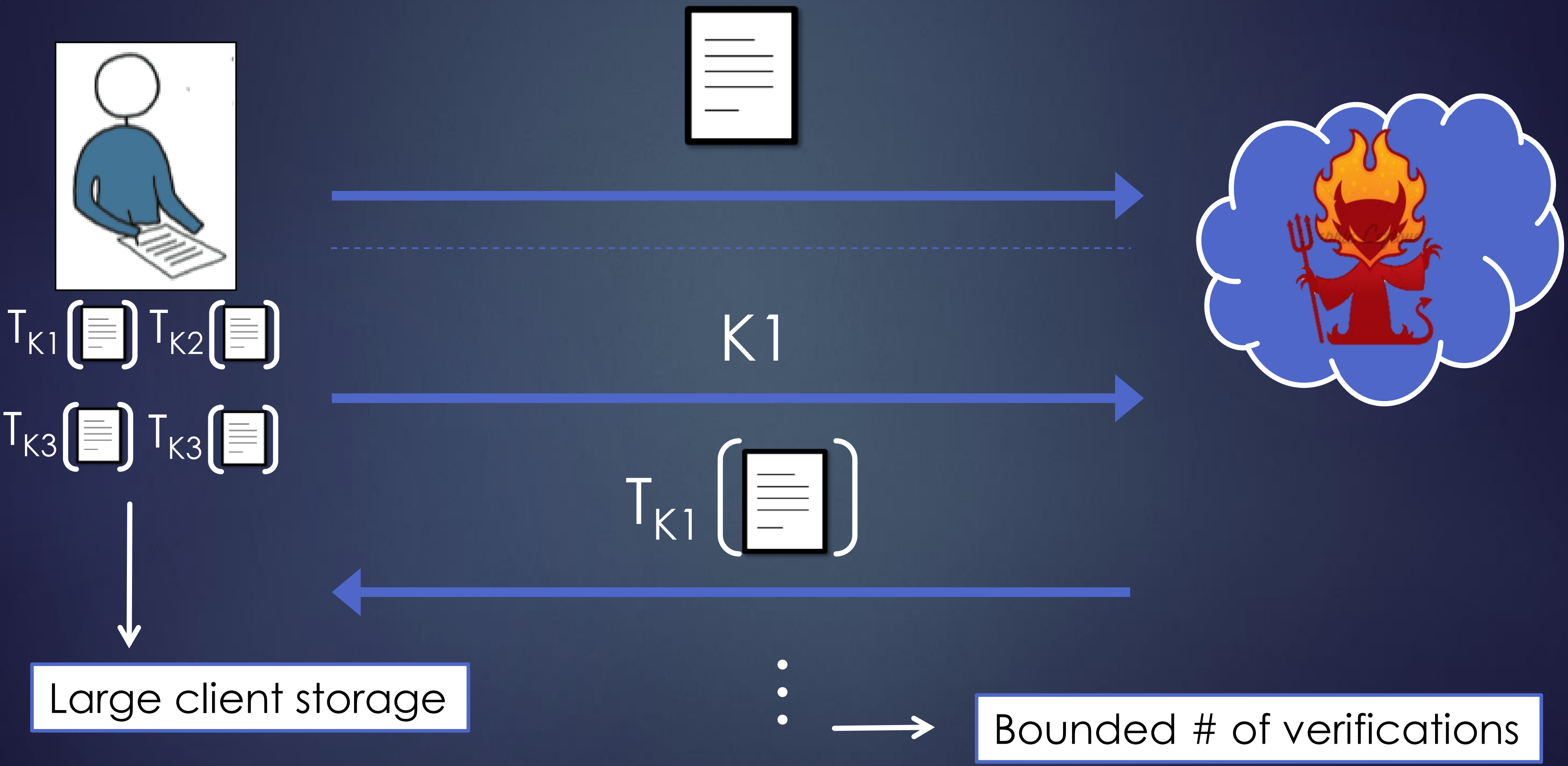
# Simple Solutions



Cloud can just store hash!

Linear comm. complexity

# Simple Solutions



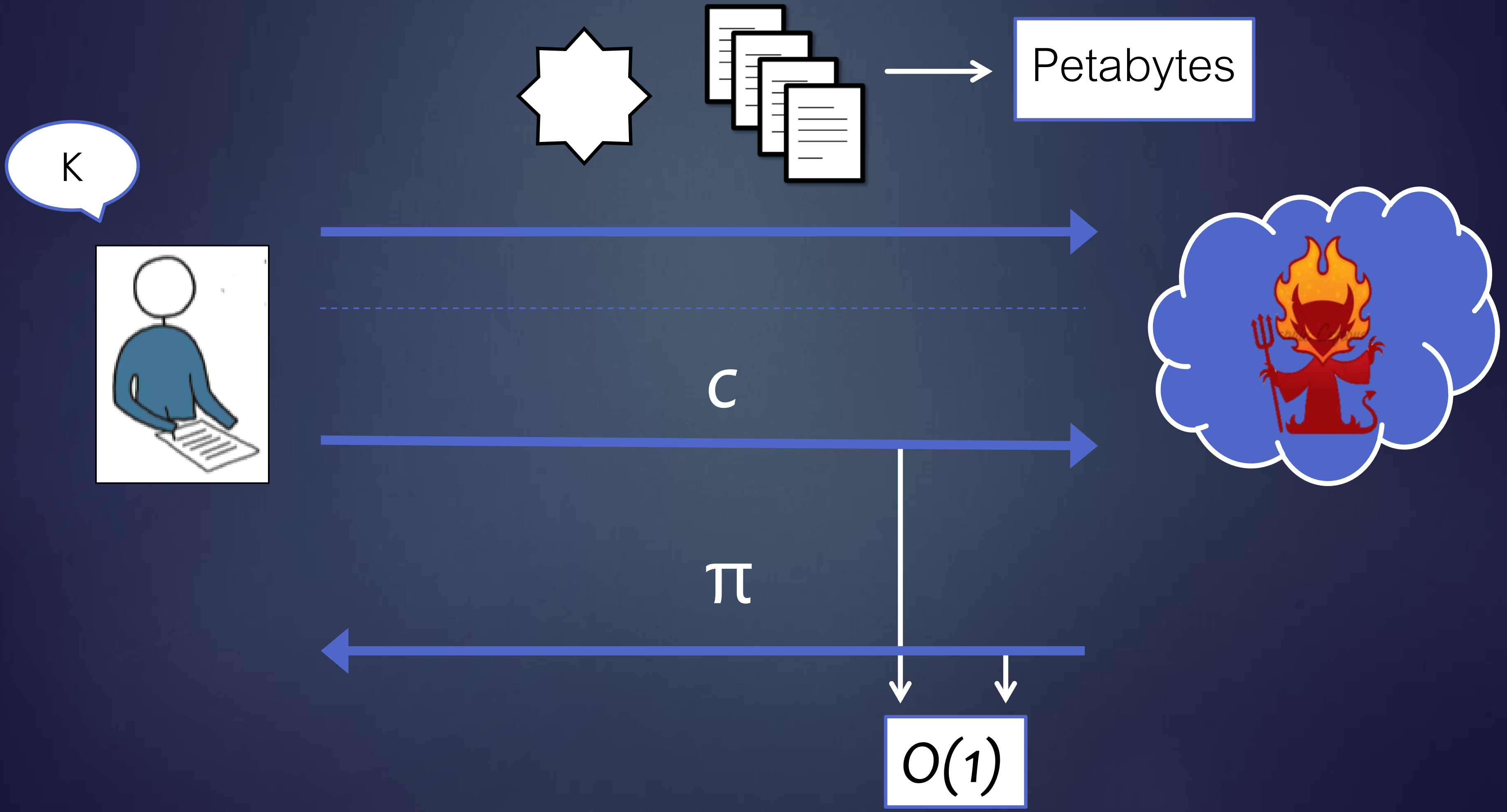
Large client storage

Bounded # of verifications

# Proofs of Storage

# Proof of Storage

[Ateniese+07,Juels-Kaliski07]



# PoS = PoR or PDP

- Proof of retrievability [Juels-Kaliski07]
  - High tampering: detection
  - Low tampering: retrievability
- Proof of data possession [Ateniese+07]
  - Detection

# PoS Security

- Completeness

**COMP**: “if Server possesses file, then Client accepts proof”

- Soundness

**SOUND**: “if Client accepts proof, then Server *possesses* file”

# Formalizing Possession

- Knowledge extractor
  - [Feige-Fiat-Shamir88, Feige-Shamir90, Bellare-Goldreich92]
  - Algorithm that extracts information from other algorithms
  - Typically done by *rewinding*
- Adapted to PoS soundness

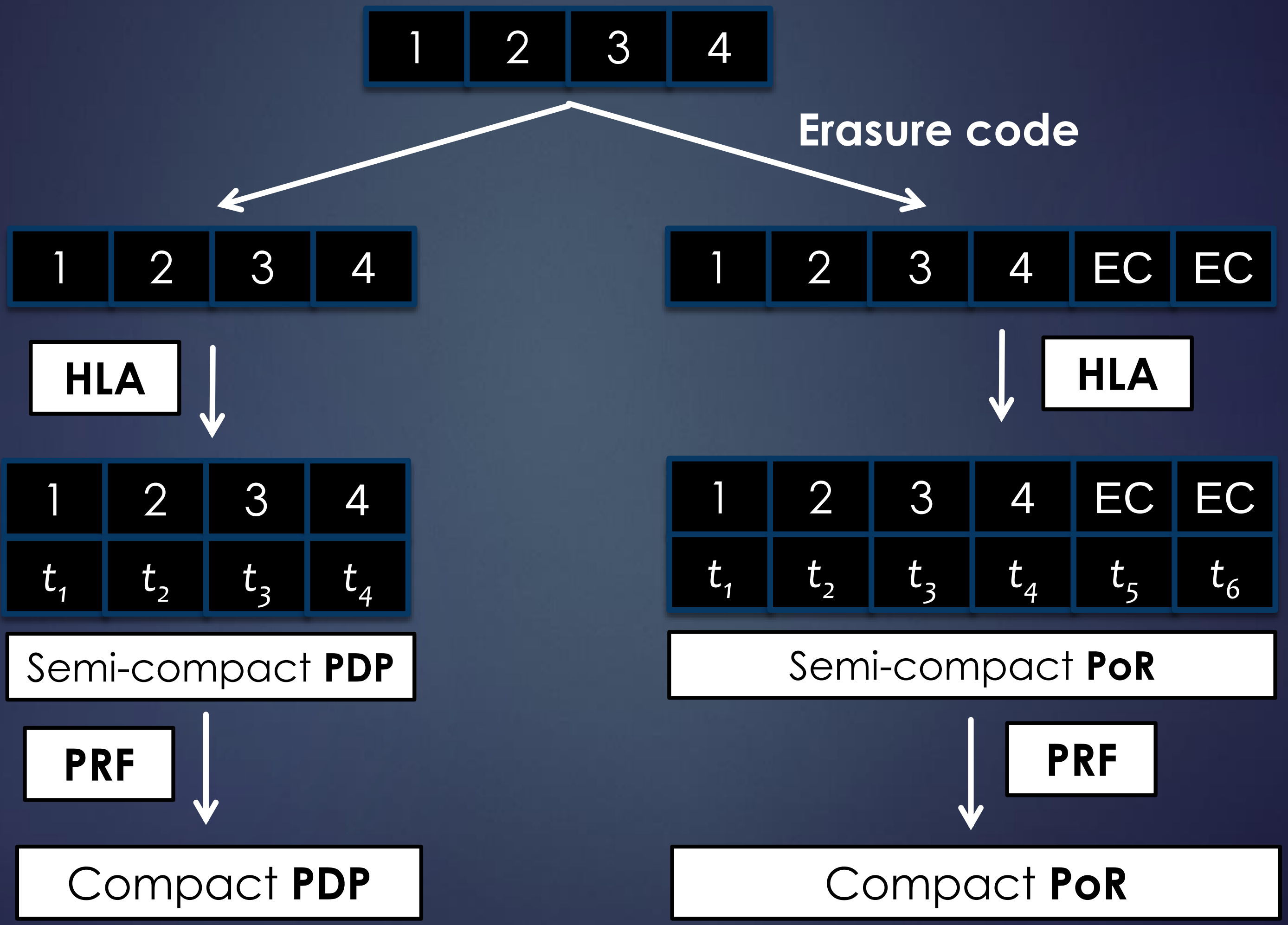
**SOUND:** "there exists an expected poly-time extractor $\mathcal{K}$ that extracts the file from any poly-time $\mathcal{A}$ that outputs valid proofs"

# Designing PoS

# Designing PoS

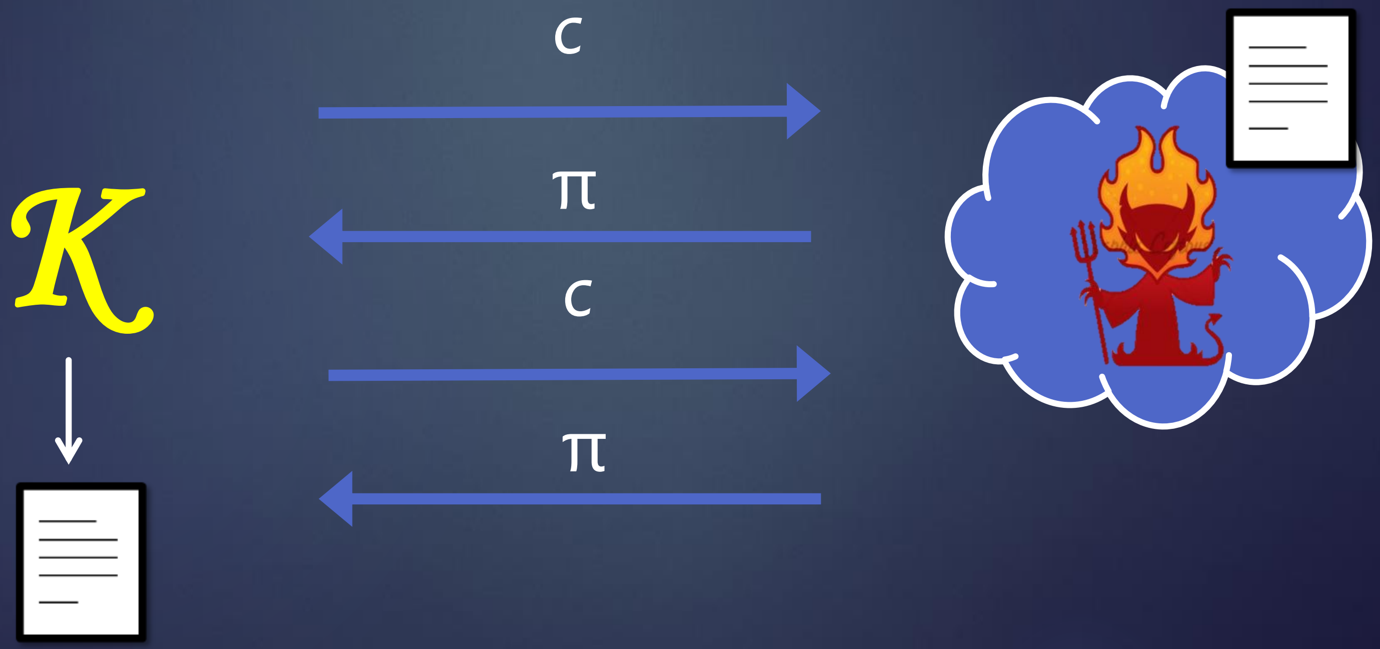
- Based on sentinels
  - [Juels-Kaliski07]
  - Embed secret blocks in data and verify their integrity
  - ☺ Very efficient encoding
  - ☹ Only works with private data
- Based on homomorphic linear authenticators (HLA)
  - [Ateniese+07]
  - Authenticates data with tags that can be aggregated
  - ☺ works with public data

# HLA-based PoS



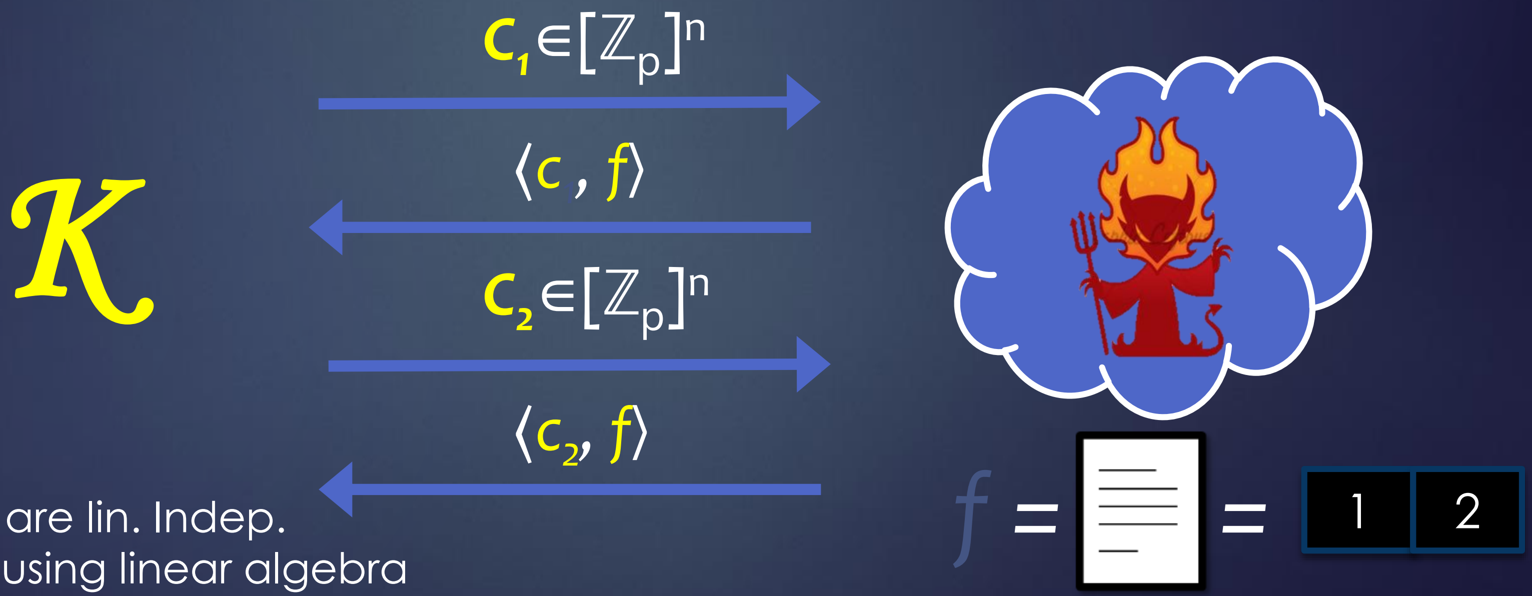
# Extracting via Linear Algebra

**SOUND**: "there exists an expected poly-time extractor $\mathcal{K}$ that extracts the file from any poly-time $\mathcal{A}$ that outputs valid proofs"



# Extracting via Linear Algebra

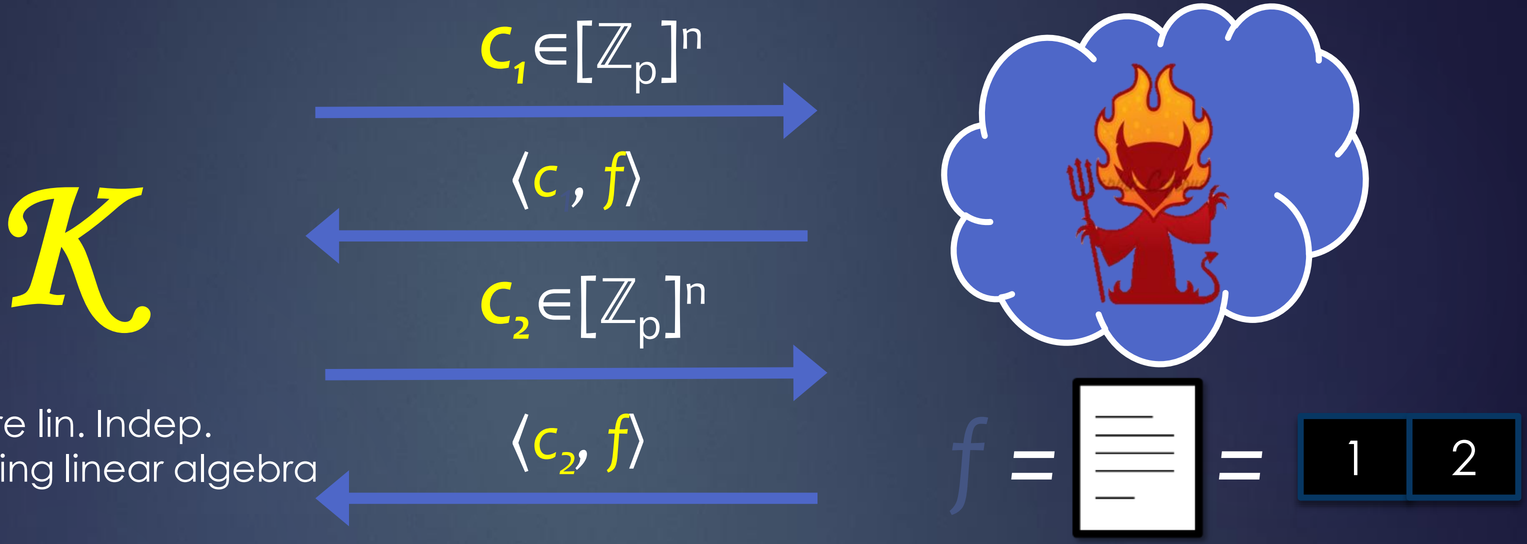
**SOUND:** "there exists an expected poly-time extractor $\mathcal{K}$ that extracts the file from any poly-time $\mathcal{A}$ that outputs valid proofs"



**Extract $f$**

1. If $c_1$ and $c_2$ are lin. Indep.
2. solve for $f$ using linear algebra

# Extracting via Linear Algebra



$\mathcal{K}$

$c_1 \in [\mathbb{Z}_p]^n$

$\langle c , f \rangle$

$c_2 \in [\mathbb{Z}_p]^n$

$\langle c_2, f \rangle$

$f =$ $=$ 1 | 2

**Extract** $f$

1. If $c_1$ and $c_2$ are lin. Indep.
2. solve for $f$ using linear algebra

- What if $c_1$ and $c_2$ are not linearly independent?
  - Just pick them at random
- What if $\mathcal{A}$ doesn't compute inner product?
  - Use HLAs!

# HLA

- Syntax
  - $\mathrm{Gen}(1^k) \Longrightarrow K$
  - $\mathrm{Tag}(K, \mathbf{f}) \Longrightarrow (\mathbf{t}, st)$
  - $\mathrm{Chall}(1^k) \Longrightarrow \mathbf{c}$
  - $\mathrm{Auth}(K, \mathbf{f}, \mathbf{t}, \mathbf{c}) \Longrightarrow \alpha$
  - $\mathrm{Vrfy}(K, \mu, \mathbf{c}, st) \Longrightarrow b$
- Security

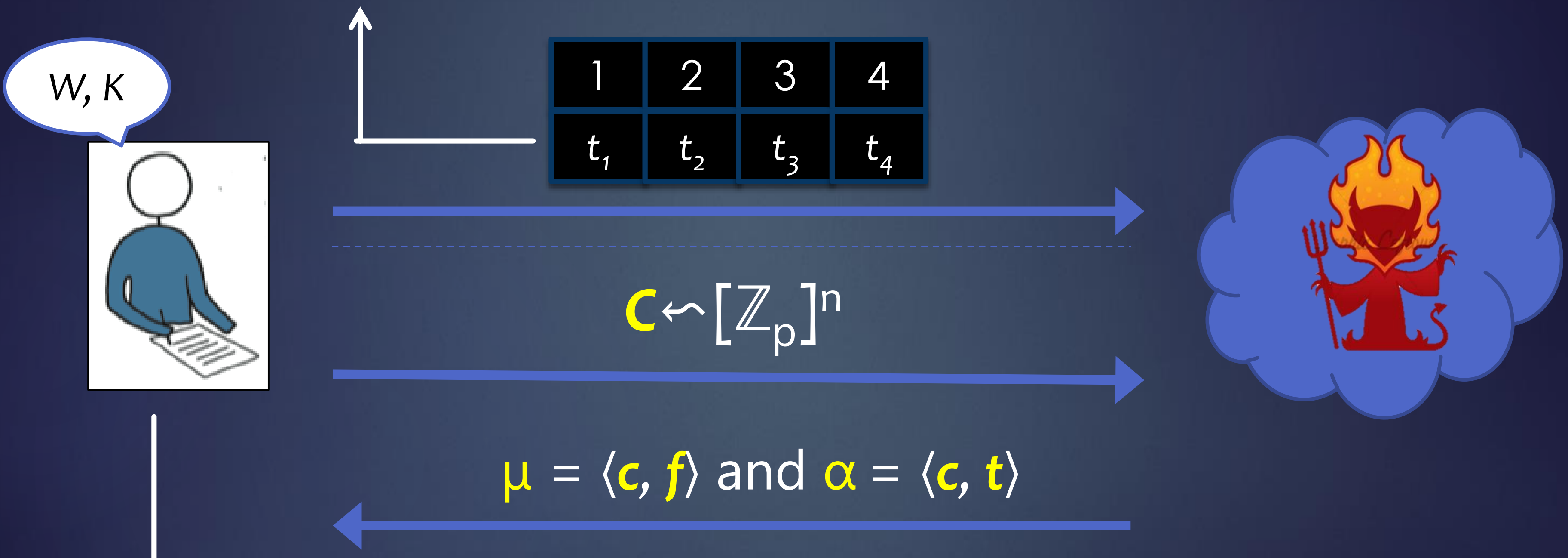
**UNF:** “given **f** and **c**, no $\mathcal{A}$ can output a valid $\alpha$ for an element $\mu \neq \langle \mathbf{c}, \mathbf{f} \rangle$”

# Constructing HLAs [AKK09]

- HLAs from *homomorphic* identification protocols
  - Multiple execs. can be verified at once (i.e., batched)
- Identification schemes
  - roughly zero-knowledge proofs of knowledge
  - Ex: Schnorr, Guillou-Quisquater, Shoup,…
- Previous HLAs are instances of AKK transform
- New HLA based on Shoup’s ID scheme

# Simple HLA [Shacham-Waters08]

$t_i = H_K(i) + f_i \cdot w$



| 1 | 2 | 3 | 4 |
|---|---|---|---|
| $t_1$ | $t_2$ | $t_3$ | $t_4$ |

$\mathbf{c} \leftarrow\!\!\sim [\mathbb{Z}_p]^n$

$\mu = \langle \mathbf{c}, \boldsymbol{f} \rangle$ and $\alpha = \langle \mathbf{c}, \mathbf{t} \rangle$

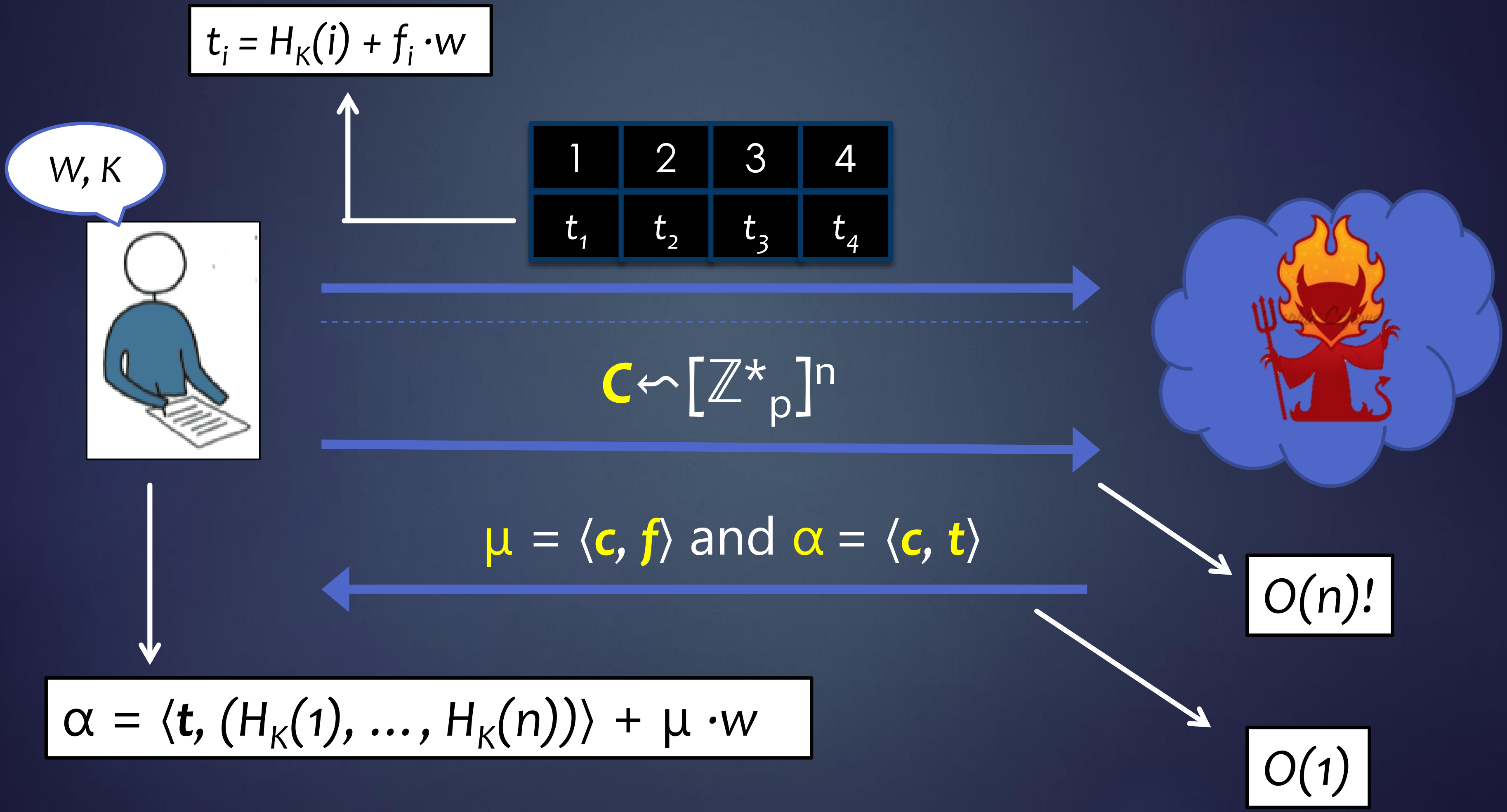
$\alpha = \langle \mathbf{c}, (H_K(1), \ldots, H_K(n)) \rangle + \mu \cdot w$

# Simple HLA

**UNF**: "given **f** and **c**, no $\mathcal{A}$ can output a valid α for an element μ ≠ ⟨**c**, **f**⟩"

- UNF: α proves that μ is the inner product of **f** and **c**
- Why is Simple HLA unforgeable?
  - For intuition see [Ateniese-K.-Katz10]
  - Connection to 3-move identification protocols

# Simple HLA = Semi-Compact PoS



$t_i = H_K(i) + f_i \cdot w$

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| $t_1$ | $t_2$ | $t_3$ | $t_4$ |

$c \leftsquigarrow [\mathbb{Z}^*_p]^n$

$\mu = \langle \mathbf{c}, \boldsymbol{f} \rangle$ and $\alpha = \langle \mathbf{c}, \mathbf{t} \rangle$

$O(n)!$

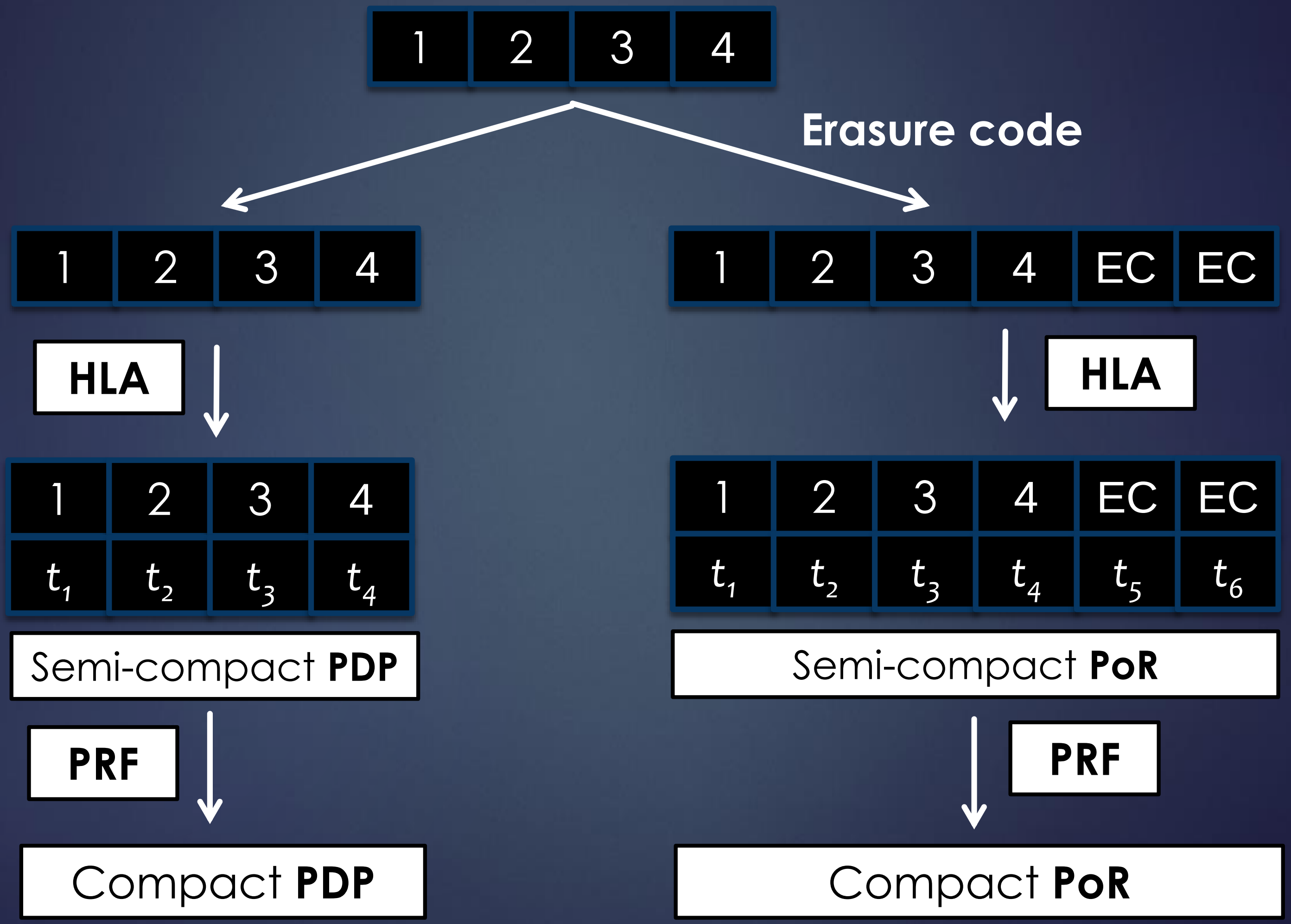
$O(1)$

$\alpha = \langle \mathbf{t}, (H_K(1), \ldots, H_K(n)) \rangle + \mu \cdot w$

# Compressing Challenges

- Idea #1
  - [Ateniese+07]
  - Send key to a PRF and have server generate challenge vector
  - Problem: how do we reduce to PRF security if $\mathcal{A}$ knows the PRF key?
- Idea #2
  - [Shacham-Waters08] Use a random oracle
- Idea #3
  - [Dodis-Vadhan-Wichs10] Use an expander-based derandomized sampler
- [Ateniese-K.-Katz10]
  - Idea#1 is secure
  - Security of PRF implies that PRF-generated vectors are linearly independent with high probability

# HLA-based PoS



1 2 3 4

Erasure code

1 2 3 4

1 2 3 4 EC EC

HLA

1 2 3 4

$t_1$ $t_2$ $t_3$ $t_4$

Semi-compact **PDP**

PRF

Compact **PDP**

HLA

1 2 3 4 EC EC

$t_1$ $t_2$ $t_3$ $t_4$ $t_5$ $t_6$

Semi-compact **PoR**

PRF

Compact **PoR**

# Constructions

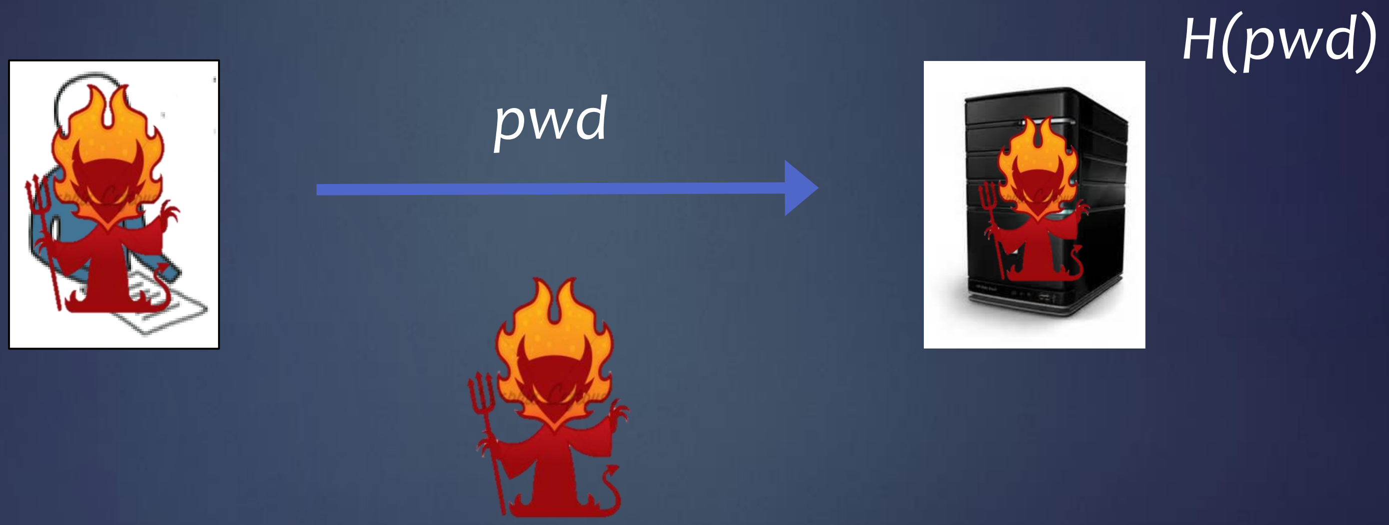
| | Assmpt. | Verif. | ROM | Dyn. | Unbounded |
|---|---|---|---|---|---|
| [ABC07+] | RSA+KEA | public | Yes | No | Yes |
| [JK07] | OWF | private | No | Yes | No |
| [SW08] | BDH | public | Yes | No | Yes |
| [SW08] | OWF | private | No | No | Yes |
| [APMT09] | OWF | private | Yes | Yes | No |
| [EKPT09] | Fact | public | Yes | Yes | Yes |
| [DVW09] | OWF | private | No | No | No |
| [AKK09] | Fact | Public | Yes* | No | Yes |

# Applying PoS

# PoS Applications

- Verifying integrity [Juels-Kaliski07, ABC+07,…]
- Providing availability
  - HAIL [Bowers-Juels-Oprea09]
  - Iris [Stefanov-vDijk-Juels-Oprea12]
- Verifying fault tolerance [Bowers-vDijk-Juels-Oprea11]
- Verifying geo-location
  - [Benson-Dowsley-Shacham11, Watson-SafaviNaini-Alimomeni-Locasto-Naranayan12, Gondree-Peterson13]
- *Malware-resistant authentication* [Ateniese-Faonio-K.-Katz13]

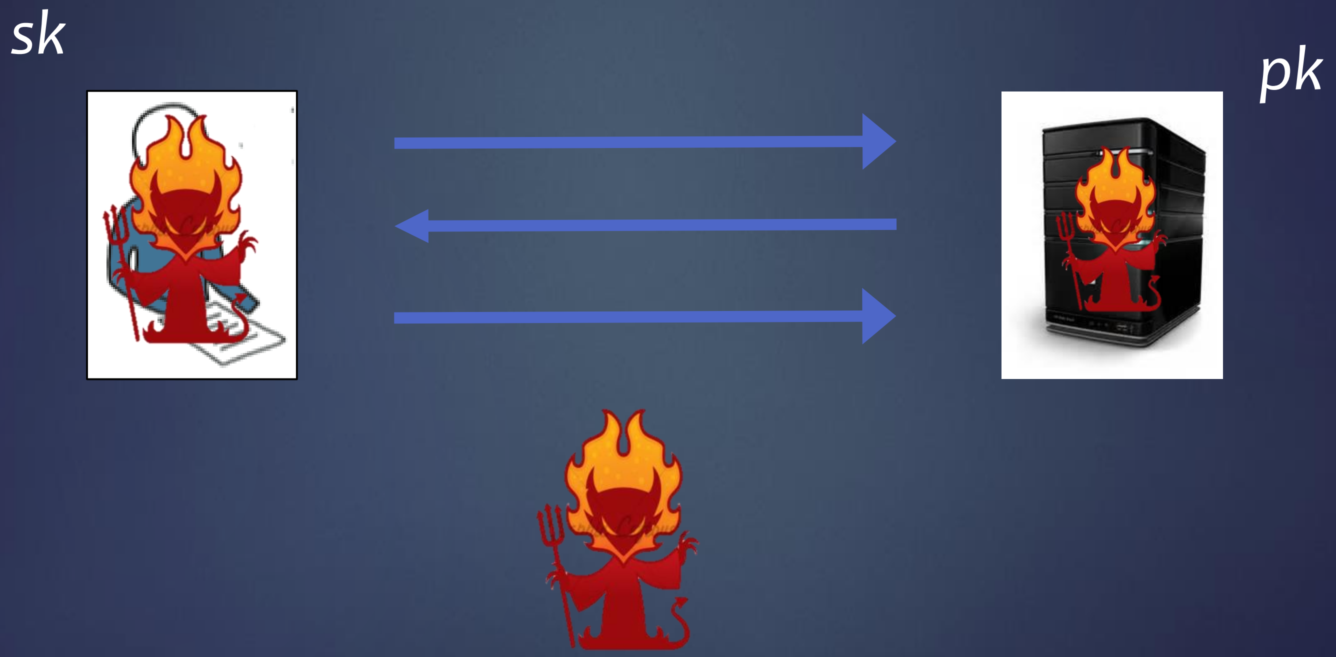
# Identification



$pwd$

$H(pwd)$

# Identification Schemes



*sk*

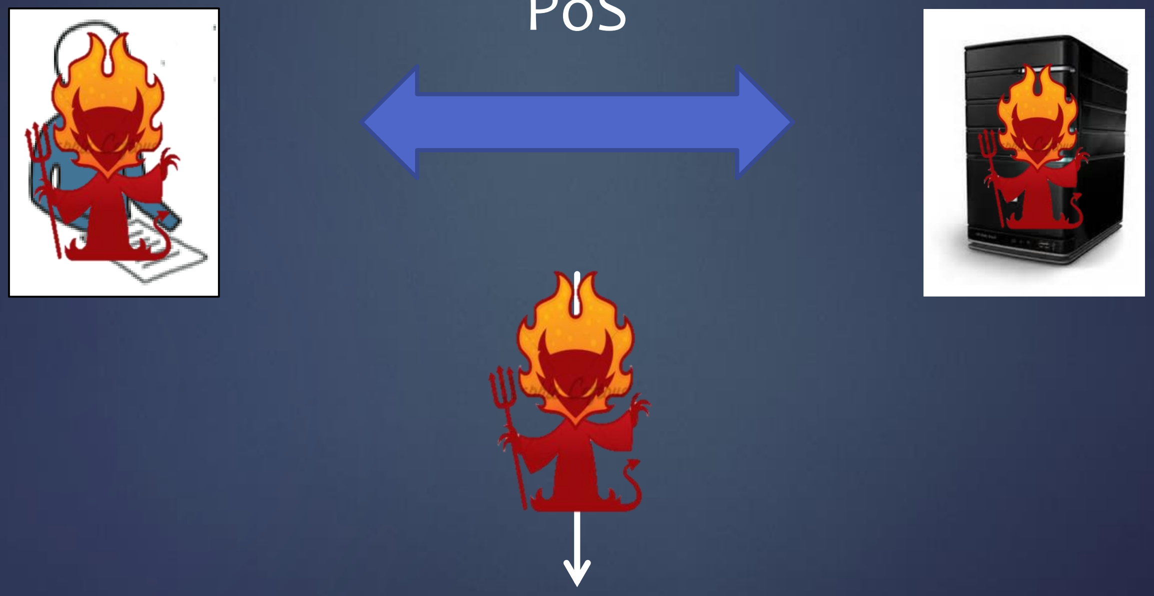
*pk*

# Bounded Retrieval Model

- High-level idea
  - $\mathcal{A}$ can recover $\lambda$ bits of secret key
  - Make secret key larger than $\lambda$ bits
  - Efficiency independent of secret key size
- Concretely
  - 20GB secret key
  - Long time needed for $\mathcal{A}$ to recover 20GB w/o detection
  - Scheme efficiency independent of key size

# BRM-ID via PoS [AFKK13]

$sk = f \leftsquigarrow \{0,1\}^k$



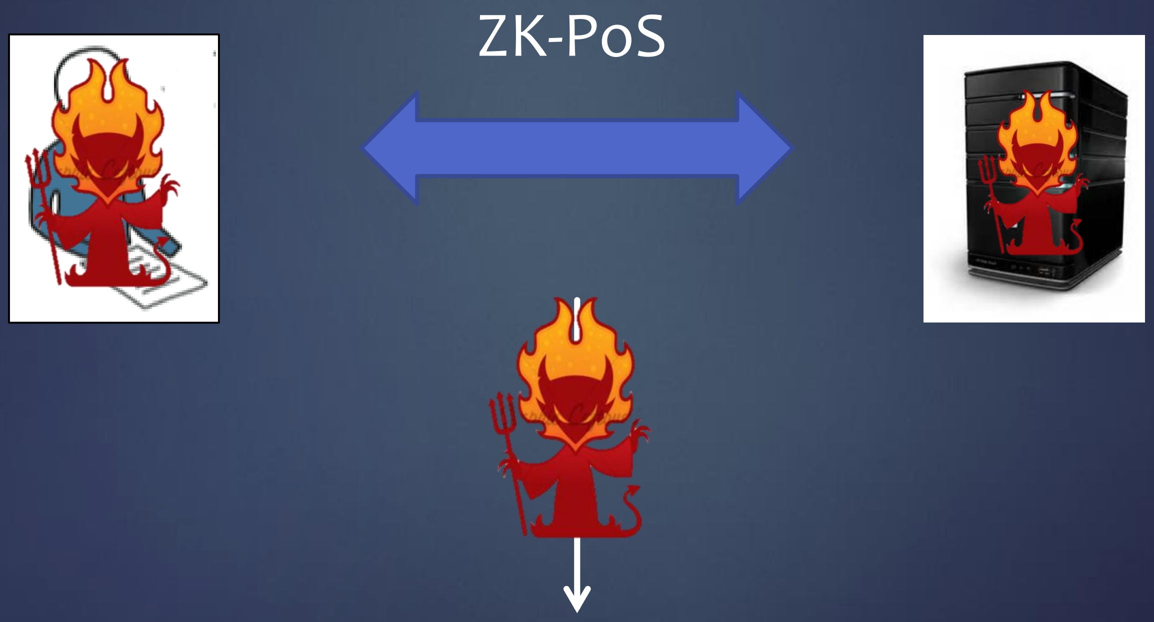
PoS

*st*

$O(1)$

# BRM-ID via PoS [AFKK13]

$sk = f \leftsquigarrow \{0,1\}^k$



ZK-PoS

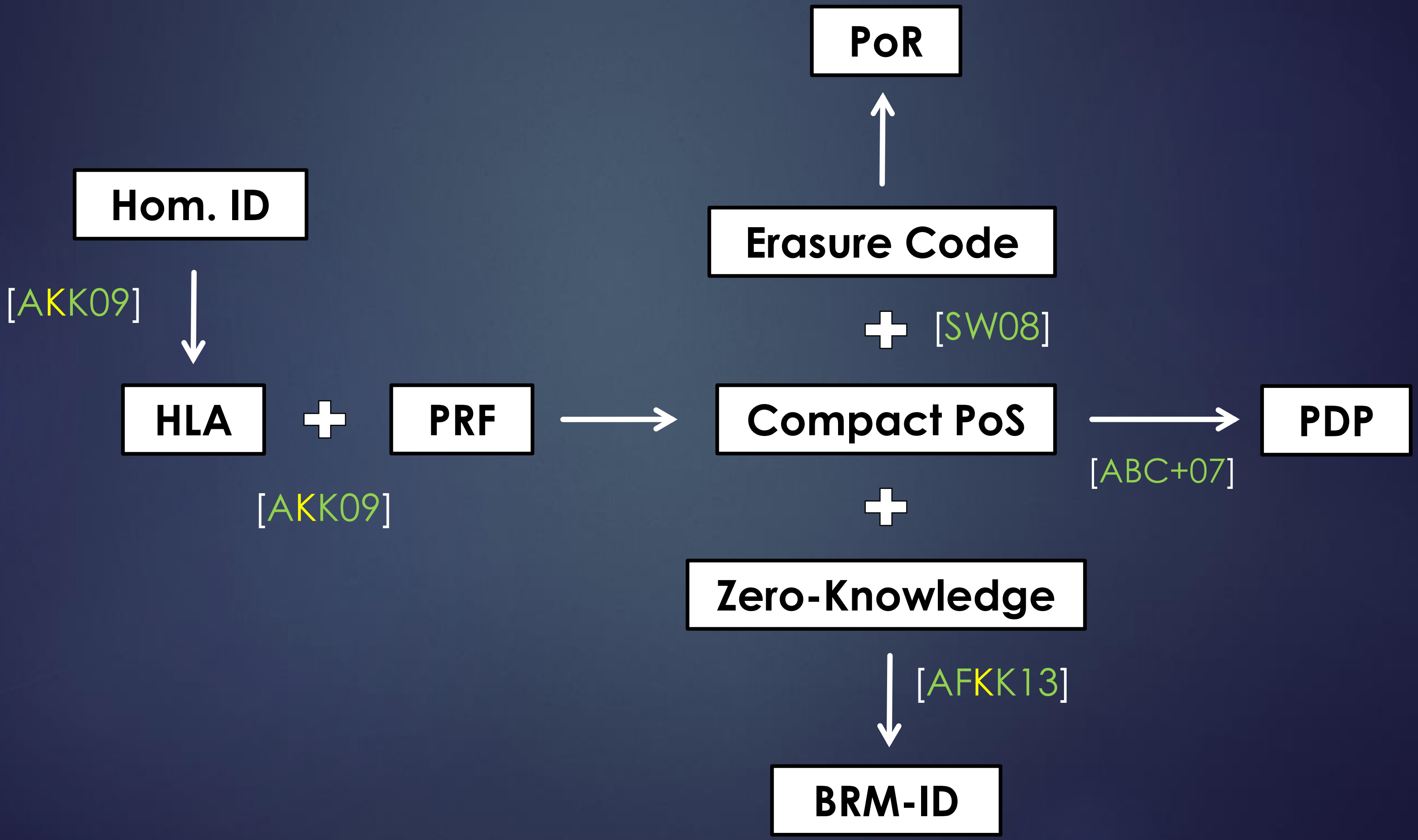
$st$

$O(1)$

# Zero-knowledge PoS

- [Wang-Chow-Wang-Ren-Lou09]
  - Bilinear DH (?)
  - Based on [Shacham-Waters08]
- [Ateniese-Faonio-K.-Katz13]
  - Construction #1: RSA
  - Construction #2: Factoring
  - Based on [ABC07+]
  - Full proof of security

# HLA-Based PoS Design



# BRM-ID

- [Alwen-Dodis-Wichs09]
  - 3 BRM-IDs
  - Based on Okamoto ID scheme
  - Asymptotically less efficient than ours

# Our RSA-Based BRM-ID

[AFKK13]

- Machine #1: PC1-HD
  - Pentium Dual-Core 2.93GHz
  - 2MB L2 cache
  - 2GB DDR2 800MHz of RAM
  - 1TB SATA 6Gb/s rotating hard drive
- Machine #2: PC1-USB
  - Machine #1 + USB drive
- Machine #3: PC2-SSD
  - Intel Xeon 8-Core 2.2GHz
  - 16MB L3 cache
  - 256GB DDR3 1600MHz of RAM
  - RAID 4 512GB SATA SSD hard drives

# Our RSA-Based BRM-ID

[AFKK13]

- 1020 bits of security + 256MB of leakage
    - 348MB secret key
    - PC1-HD: 0.18s
    - PC1-USB: 0.5s
    - PC2-SSD: 0.12s
- 1020 bits of security + 4GB of leakage
    - 5584GB secret key
    - PC1-HD: 2.5-3s
    - PC1-USB: 1s
    - PC2-SSD: 0.12s

# Conclusions

- PoS are interesting in practice
  - Well motivated
  - Different guarantees (PDPs and PORs)
  - Efficient constructions
  - Based on variety of assumptions (RSA, BDH, OWF)
- PoS are interesting in theory
  - Non-trivial security definitions and constructions
  - Interactive proofs, signatures, coding theory
- PoS are useful
  - Integrity, availability, geo-location, malware-resistant authentication

# The End