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**
  - $\text{Gen}(1^k) \Rightarrow (sk, vk)$
  - $\text{Sign}(sk, m) \Rightarrow \sigma$
  - $\text{Vrfy}(vk, m, \sigma) \Rightarrow b$

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

- **Security**

**UNF**: “given $m$ and $\sigma$, no $A$ can output a valid $\sigma'$ for an element $m' \neq m$”
Communication Channels
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

Erasure code

1 2 3 4

HLA

1 2 3 4

Semi-compact PDP

1 2 3 4

PRF

Compact PDP

1 2 3 4

Semi-compact PoR

1 2 3 4

HLA

1 2 3 4

PRF

Compact PoR

1 2 3 4

EC  EC
Extracting via Linear Algebra

**SOUND**: “there exists an expected poly-time extractor $K$ that extracts the file from any poly-time $A$ 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”

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

- What if \( c_1 \) and \( c_2 \) are not linearly independent?
  - Just pick them at random
- What if \( A \) doesn’t compute inner product?
  - Use HLAs!

Extract \( f \)
1. If \( c_1 \) and \( c_2 \) are lin. Indep.
2. Solve for \( f \) using linear algebra

\[
f = \begin{align*}
\langle c_1, f \rangle \\
\langle c_2, f \rangle
\end{align*}
\]
Syntax

- \( \text{Gen}(1^k) \rightarrow K \)
- \( \text{Tag}(K, f) \rightarrow (t, st) \)
- \( \text{Chall}(1^k) \rightarrow c \)
- \( \text{Auth}(K, f, t, c) \rightarrow \alpha \)
- \( \text{Vrfy}(K, \mu, c, st) \rightarrow b \)

Security

**UNF**: “given \( f \) and \( c \), no \( \mathcal{A} \) can output a valid \( \alpha \) for an element \( \mu \neq (c, f) \)”
Constructing HLAs \[\text{[AKK09]}\]

- 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]

\[ t_i = H_K(i) + f_i \cdot w \]

\[ \alpha = \langle c, (H_K(1), ..., H_K(n)) \rangle + \mu \cdot w \]

\[ C \leftarrow [\mathbb{Z}_p]^n \]

\[ \mu = \langle c, f \rangle \text{ and } \alpha = \langle c, t \rangle \]
Simple HLA

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

- UNF: $\alpha$ proves that $\mu$ 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

\[ t_i = H_K(i) + f_i \cdot w \]

\[ \mathbf{c} \leftarrow [\mathbb{Z}_p^*]^n \]

\[ \mu = \langle c, f \rangle \text{ and } \alpha = \langle c, t \rangle \]

\[ \alpha = \langle t, (H_K(1), \ldots, H_K(n)) \rangle + \mu \cdot w \]
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 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

Erasure code

HLA

Semi-compact PDP

PRF

Compact PDP

HLA

Semi-compact PoR

PRF

Compact PoR
## Constructions

<table>
<thead>
<tr>
<th>Assmpt.</th>
<th>Verif.</th>
<th>ROM</th>
<th>Dyn.</th>
<th>Unbounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ABC07+]</td>
<td>RSA+KEA</td>
<td>public</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>[JK07]</td>
<td>OWF</td>
<td>private</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[SW08]</td>
<td>BDH</td>
<td>public</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>[SW08]</td>
<td>OWF</td>
<td>private</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[APMT09]</td>
<td>OWF</td>
<td>private</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[EKPT09]</td>
<td>Fact</td>
<td>public</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[DVW09]</td>
<td>OWF</td>
<td>private</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[AKK09]</td>
<td>Fact</td>
<td>Public</td>
<td>Yes*</td>
<td>No</td>
</tr>
</tbody>
</table>
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 \rightarrow H(pwd)
Identification Schemes

\[ sk \rightarrow pk \]

- \( sk \)
- \( pk \)
Bounded Retrieval Model

- High-level idea
  - $A$ 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 $A$ to recover 20GB w/o detection
  - Scheme efficiency independent of key size
BRM-ID via PoS [AFKK13]

\[ sk = f \leftarrow \{0, 1\}^k \]

PoS

O(1)
BRM-ID via PoS [AFKK13]

\[ sk = f \leftarrow \{0,1\}^k \]

ZK-PoS

\[ O(1) \]

st
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

Hom. ID \[\text{[AKK09]}\]
  \(\downarrow\)
  HLA \[\text{[AKK09]}\]
    \(+\)
    PRF

Erasure Code \[\text{[SW08]}\]
  \(+\)
  Compact PoS
    \(+\)
    Zero-Knowledge \[\text{[AFKK13]}\]
      \(\downarrow\)
      BRM-ID

PoR
  \(+\)
  [ABC+07]
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