ArchiveSafe LT: Secure Long-term Archiving System

Moe Sabry, Reza Samavi
Introduction and Motivation
Long-Term Secure Archiving

• Every year the amount of digitally stored sensitive information increases significantly.

• Some governmental and legal documents, health and tax records are required to be securely archived for decades to comply with various laws and regulations.

• Regular cryptographic schemes are not guaranteed to stay secure for such long time periods.

• Current solutions rely on information-theoretic techniques e.g.: Multi-server secret sharing.

• They require costly and complicated setup:
  • Private channels for Quantum key distribution (QKD) & One time pads (OTP)
  • Trusted Execution Environments (TEEs)
Gap and Motivation

• **Problem:**
  • Long-Term secure archiving is essential but current solutions are complicated and costly.

  • **Thought:**
    • Is there any other way to prolong the lifespan of standard cryptographic schemes?

  • **Idea:**
    • Robust Combiners!
Robust Combiners

A Robust Combiner is a combination of multiple cryptographic schemes into one so the resulting scheme is robust to the failure of any of the combined ones.
Contributions
Contributions

• We developed *ArchiveSafe LT*, a framework ensuring long-term integrity and confidentiality without the complexity and cost required by the state-of-the-art systems.

• *ArchiveSafe LT* is built on the novel idea of utilizing a pool of computationally-secure schemes to build robust combiners to secure the data.

• *ArchiveSafe LT* provides significant performance improvement and cost reduction compared to the currently available systems.
Related Work

• LINCOS (2017)\(^1\) utilizes proactive secret sharing and information-theoretic hiding commitments for integrity and authenticity protection.

• PROPYLA (2018)\(^2\) enables partial data integrity and authenticity checks. Utilizes oblivious random access machine to hide access patterns.

• ELSA (2018)\(^3\) introduces more efficient data integrity and authenticity checks.

• SAFE (2020)\(^4\) Utilizes a trusted execution environment (TEE) provider to perform secret shares generation.

ArchiveSafe LT Framework
Framework Overview

- **ArchiveSafe LT** defines an archive as a group of data files.

- The framework implements six operations to cover the archive life cycle:
  - Initialize()
  - Update()
  - EvolveIntegrity()
  - EvolveConfidentiality()
  - Retrieve()
  - Delete()
Framework Overview

- Files can be updated, deleted or retrieved individually without processing the whole archive. A unique feature of ArchiveSafe LT.

- When a cryptographic scheme is compromised, the Evolution protocol is initiated to strengthen the combiner by adding an additional secure scheme to it.
Designs Overview

- We present two system designs based on the *ArchiveSafe LT* framework:

<table>
<thead>
<tr>
<th></th>
<th>ASLT-D1</th>
<th>ASLT-D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td><em>Untrusted</em> or incapable storage provider</td>
<td><em>Trusted</em> and capable storage provider</td>
</tr>
<tr>
<td>Processing</td>
<td>Data owner does all processing</td>
<td>Storage provider performs evolution processes</td>
</tr>
</tbody>
</table>

12
Design 1 - Evolution

**Data Owner**

Policy: (\((\Pi_1,k_1), (\Pi_2,k_2),...\))

Map: ((File_1,L_3,fcode_1),
    (File_2,L_5,fcode_2),...)

Integrity Object: Tree root \((r_T)\)

**Storage Provider**

Archive: ((fcode_1,C_1),
    (fcode_2,C_2),...)

Integrity Object: Internal nodes \((nodes)\)

File codes to be evolved

Archived Files + \((nodes)\)

Evolve

Verify

Evolved Files + \((nodes')\)

Store

\(\Pi\): A symmetric encryption scheme

\(k\): secret key

\(L\): Number of evolutions

\(fcode\): File ID
Design II - Evolution

Data Owner
- **Policy**: \((\Pi_1, k_1), (\Pi_2, k_2), \ldots\)
- **Map**: \(((\text{File}_1, L_3, \text{fcode}_1), (\text{File}_2, L_5, \text{fcode}_2), \ldots)\)
- **Integrity Object**: Tree root \((r_T)\)

Storage Provider
- **Archive**: \(((\text{fcode}_1, C_1), (\text{fcode}_2, C_2), \ldots)\)
- **Public Enc.**: \((\Sigma, k_{pub}, k_{priv})\)
- **Integrity Object**: Internal nodes (\(nodes\))

- \(\Sigma\), \(k_{pub}\)
- \(\Sigma(\text{File codes to be evolved} + (\Pi_1, k_1), k_{pub})\)
- \(r'_T + nodes'\)

Evolve
Verify
Store

\(\Pi\): A symmetric encryption scheme
\(k\): secret key
\(\Sigma\): An asymmetric encryption scheme
\(k_{pub}\): Public key
\(k_{priv}\): Private key
\(L\): Number of evolutions
\(\text{fcode}\): File ID
Security Proofs - Tamarin

- To ensure no adversarial scenario is missed, we utilized an automatic prover (Tamarin*) for the confidentiality and integrity security proofs.

- Limitation: We modeled up to two evolution processes.

* https://tamarin-prover.github.io/
Tamarin - Model

• **Functions:**
  - KeyGen/2, Lock/3, Unlock/3

• **Equations:**
  - Unlock(schemenum, KeyGen(schemenum, sk),
    Lock(schemenum, KeyGen(schemenum, sk), data)) = data

• **Rules:**
  - **Oracles:** OCorruptKey, OUpdate, OEvolve, ODelete, ORetrieve2/3, OForge2/3.
  - **Challenges:** DistinguishChallenge, IntegrityChallenge.
Tamarin – Confidentiality Lemma

All fname fcontents #tchallenge.

ChallengeStored(fname, fcontents) @ #tchallenge & not(Ex #tr . RetrievedContents(fname, fcontents) @ #tr) & not(
\(\text{(Ex #tga tc1 tc2 . GotArchive(fname, '2') @ #tga & Corrupted('1') @ tc1 & Corrupted('2') @ tc2)}\)
\| (Ex #tga tc2 tc3 . GotArchive(fname, '3') @ #tga & Corrupted('2') @ tc2 & Corrupted('3') @ tc3)))

==> not(Ex #tk . K(fcontents) @ #tk)
Tamarin – Integrity Lemma

All fname layer1 layer2 fcontents #tforgeanswer .
ForgeAnswer(fname, layer1, layer2, fcontents) @ #tforgeanswer

=>
(Ex fname2 #tstored . Stored(fname2, fcontents) @ #tstored)
| (Ex #tc1 #tc2 . Corrupted(layer1) @ #tc1 & Corrupted(layer2) @ #tc2)

A valid challenge exists
Adversary is not presenting a valid file under a different valid file name
The adversary did not broke schemes 1 & 2 at the same time
Evaluation and Results
Evaluation Experiment

- We measure the system's performance through an experiment mimicking the evolution of an archive:
  - **1992**: Initial creation using DES + 3DES and MD2 + MD5.
  - **2001**: 1st evolution using AES-128 and SHA-256.
  - **2004**: 2nd evolution using AES-192 and SHA-384.
  - **2015**: 3rd evolution using AES-256 and SHA3-512.
Evaluation Experiment Setup

• The experiment was performed using HP Z420 (Ubuntu Linux 20.04.3 LTS, 8-core Intel Xeon CPU E5-1620 3.6 GHz, 32 GiB RAM, 1 TB SSD).

• We performed 100 repetitions of the following tasks:
  • 1000 sample files of each size were randomly generated.
  • We measured times for:
    • Initial creation.
    • Evolution.
    • Retrieval.
## Results – Performance & Space I

<table>
<thead>
<tr>
<th></th>
<th>LINCOS(^1), PROPYLA(^2), ELSA(^3)</th>
<th>ArchiveSafe LT</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation Time</td>
<td>55.2 Hrs.</td>
<td>7.7 Hrs. (± 2%)</td>
<td>Improvement increases with larger archive sizes</td>
</tr>
<tr>
<td>Evolution Time</td>
<td>110.4 Hrs.</td>
<td>0.7 Hrs. (± 2%)</td>
<td>Improvement increases with larger archive sizes</td>
</tr>
<tr>
<td>Storage Space</td>
<td>3x</td>
<td>1x</td>
<td>Improvement increases with more shares</td>
</tr>
</tbody>
</table>

*ArchiveSafe LT* time & space utilization compared to other systems

On a 158 GB Archive

## Results – Performance & Space II

<table>
<thead>
<tr>
<th></th>
<th>SAFE&lt;sup&gt;4&lt;/sup&gt;</th>
<th>ArchiveSafe LT</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Creation Time</strong></td>
<td>10 Sec.</td>
<td>3.3 Sec. (± 2%)</td>
<td>Improvement increases with larger archive sizes</td>
</tr>
<tr>
<td><strong>Evolution Time</strong></td>
<td>109 Sec.</td>
<td>3.2 Sec. (± 2%)</td>
<td>Improvement increases with larger archive sizes</td>
</tr>
<tr>
<td><strong>Storage Space</strong></td>
<td>3x</td>
<td>1x</td>
<td>Improvement increases with more shares</td>
</tr>
</tbody>
</table>

*ArchiveSafe LT* time & space utilization compared to SAFE (TEE)

On a 10 MB Archive

Conclusion
Conclusion

• *ArchiveSafe LT* provides long-term integrity and confidentiality using standard cryptographic schemes through a robust combiner.

• Compared to state-of-the-art approaches, *ArchiveSafe LT* reduces cost and complexity and provides better performance and space utilization.

• **Future Work:**
  • Improve *ArchiveSafe LT* efficiency and robustness in supporting long-term integrity.
ArchiveSafe LT: Secure Long-term Archiving System

• A system providing long-term integrity and confidentiality through robust combiners.
• Utilizes standard cryptographic schemes.
• Can be utilized for in-house or outsourced storage.
• Better performance and space utilization than similar systems.

We gratefully acknowledge Dr. Douglas Stebila for many helpful comments and discussions on this paper.

Authors: Moe Sabry (alym2@mcmaster.ca), Reza Samavi (samavi@ryerson.ca)

Paper: https://dl.acm.org/doi/10.1145/3564625.3564635
Artifacts: https://github.com/samavi/pubs/tree/main/ArchiveSafeLT

Thank you!