ArchiveSafe LT: Secure Long-term Archiving System

Moe Sabry, Reza Samavi
Introduction and Motivation
Long-Term 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 which require costly and complicated implementations:
  • Multi-server secret sharing
  • Quantum key distribution (QKD)
  • One time pads (OTP)
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!
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

• To ensure long-term integrity and confidentiality without the complexity and cost of private channels for QKD, OTP and secret sharing, we developed *ArchiveSafe LT*.  

• *ArchiveSafe LT* is built on the novel idea of utilizing a pool of computationally-secure schemes to build robust combiner for data encryption and integrity verification.  

• *ArchiveSafe LT* provides significant performance improvement and cost reduction compared to the currently available systems.
ArchiveSafe LT Framework
Framework Overview I

- **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 II

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

- When a confidentiality scheme is compromised, EvolveConfidentiality() is initiated to strengthen the combiner by adding an additional secure scheme to it. Same idea for integrity.
Design I - 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**: \(((\text{fcode}_1, C_1), (\text{fcode}_2, C_2), \ldots)\)
- **Integrity Object**: Internal nodes \((nodes)\)

\(\Pi\): A symmetric encryption scheme
\(k\): secret key
\(L\): Number of evolutions
\(\text{fcode}\): File ID

File codes to be evolved

Archived Files + nodes

Verify

Evolve

Evolved Files + nodes’

Store
Design I - 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**: \(\{(\text{fcode}_1, C_1),\)
\(\ (\text{fcode}_2, C_2), \ldots\}\)

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

\(\Pi\): A symmetric encryption scheme
\(k\): secret key
\(L\): Number of evolutions
\(\text{fcode}\): File ID
Security Proofs
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 $\text{fname}$ $\text{fcontents} \ #\text{tchallenge}$.

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

$\implies$

not(Ex $\#tk . K(\text{fcontents}) @ \#tk)$

A valid challenge exists
This file was not retrieved by the adversary before
The adversary does not have the secured archive and broke schemes 1 & 2 at the same time
The adversary does not have the secured archive and broke schemes 2 & 3 at the same time
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
Experimental Evaluation
Evaluation Experiment

• Objectives:
  • We measure the system's performance through an experiment mimicking the evolution of an archive.
  • We benchmark the system performance against the state-of-the-art- systems.

• Scenario:
  • 1992: Initial creation using DES + 3DES and MD2 + MD5.
  • 2001: 1\textsuperscript{st} evolution using AES-128 and SHA-256.
  • 2004: 2\textsuperscript{nd} evolution using AES-192 and SHA-384.
  • 2015: 3\textsuperscript{rd} 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.
Evaluation Experiment Flow

Sample Files Generation

Initial Creation
DES+3DES,MD2+MD5

C = Enc(Enc(d))

1st Evolution
AES-128,SHA-256

C1 = Enc(C)

2nd Evolution
AES-192,SHA-384

C2 = Enc(C1)

3rd Evolution
AES-256,SHA3-512

C3 = Enc(C2)

Final Retrieval

C

1st Retrieval

C1

2nd Retrieval

C2

3rd Retrieval

d

c

d

d
Evaluation Experiment Challenges

• Could not have access to the state-of-the-art environments (QKD & private channels).

• Could not have access to the state-of-the-art experimental data.

• **Approach:**
  • We used the same data point (file sizes) provided in the other systems’ researches for benchmarking.
## Results – Performance & Space

<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>SAFE⁴</th>
<th>ArchiveSafe LT</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation Time</td>
<td>10 Sec.</td>
<td>3.3 Sec. (± 2%)</td>
<td>Improvement increases with larger archive sizes</td>
</tr>
<tr>
<td>Evolution Time</td>
<td>109 Sec.</td>
<td>3.2 Sec. (± 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 SAFE (TEE)

On a 10 MB Archive

---

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)

*Full version: http://

Thank you!