Distillation Codes and DoS Resistant Multicast Authentication (NDSS 2004)

Chris Karlof, Naveen Sastry @ UC Berkeley
Yaping Li, J. D. Tygar @ UC Berkeley
Adrian Perrig @ CMU

2006.04.19
Presenter: JeeHyun Hwang

Acknowledgement: Slides were originally provided by Chris Karlof
Multicast authentication problem

- Security goals
  - Packet authenticity
  - DoS resistance
- Threat model
  - Injection
  - Modification
  - Dropping
  - Eavesdropping
Two multicast authentication protocols

- Two signature amortization schemes based on erasure codes
  - SAIDA (Oakland 2002)
    Signature Amortization using the Information Dispersal Algorithm
  - Pannetrat-Molva (NDSS 2003)

- Properties of SAIDA and Pannetrat-Molva
  - Guarantee authenticity of stream
  - Low overhead (12-23 bytes per packet)
  - Robust to packet loss

- Problem: receivers vulnerable to DoS caused by injection attacks
Outline

- Erasure codes & pollution attacks
- Our solution: Distillation codes
- Distillation codes for multicast authentication
Erasure codes

Data

Encode

Parameters: (n,t)
- n encoding symbols
- t symbols needed for recovery

Encoding symbols

1
2
3
4
5
6
7
8

Lossy network

Receiver 1

3
4
5
7
8

D

Receiver 2

1
2
4
5
6
7
8

D

Decode

X

X

X

X
Pollution attacks

Decoding with invalid symbols results in an error
Naive solution: Sign every symbol
Signatures are expensive

- Generation must be cheap (real time streams)
- Verification must be cheap (real time streams, DoS)
- Overhead must be small (tens of bytes)
- No known signature satisfies all these requirements
Naive alternative: Amortize signature
Guess and check

Receiver: incorrect guess

Receiver: correct guess
Guess and check: analysis

- **Attack factor** \( f = \frac{\text{attack traffic}}{\text{valid traffic}} \)
  - Attack factor 5: 50 packets are injected per legitimate 10 packets.

- \( O((f+1)^t) \) erasure decodings and verifications needed to reconstruct the valid data

- \((n,t) = (128,64), f = 1 \rightarrow \text{at least } 2^{64} \text{ operations} \)

Separating good from bad is difficult
Erasure codes & pollution attacks
Our solution: Distillation codes
Distillation codes for multicast authentication
Distillation Codes: Overview

Receiver

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

network

Decode → D → Verify

Decode → ? → Verify

Decode → ? → Verify

Decode → ? → Verify

Decode → ? → Verify

Decode → ? → Verify

Verify → X

Verify → X

Verify → X

Verify → X
Accumulators

Accumulate

Generate witnesses

Recover

Accumulators for partitioning

Recover operation

Accumulate

w1

w2

w3

w4

w5

w6

w7

w8

w1

A

w1

A

w2

A

w2

A'

w

w

1

2

and

in some partition \( P \)

not in \( P \)

9
A fast accumulator using Merkle Hash Trees

Set accumulator value: root
Witness: siblings on root to leaf path. Size= O( log n)
Distillation Codes: Encoding

- Sign
- Erasure encode
- Generate witnesses

D → D → 

1: w_1
2: w_2
3: w_3
4: w_4
5: w_5
6: w_6
7: w_7
8: w_8

network
Distillation Codes: Decoding

1. Recover
   - 1
     - w₁
   - A₁

2. Recover
   - 2
     - w₂
   - A₁

3. Recover
   - 4
     - w₄
   - A₂

4. Recover
   - 5
     - w₅
   - A₃

- Erasure
- Decode
- Verify

- D

- ?
- Verify

- ?
- Verify

- ?
- Verify
### Distillation code performance

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Erasure codes (with signatures)</th>
<th>Distillation Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>1 erasure encoding 1 signature generation</td>
<td>1 erasure encoding 1 signature generation 2n hashes</td>
</tr>
<tr>
<td>Decoding</td>
<td>(O(f^n)) erasure decodings (O(f^n)) signature verifications</td>
<td>2f+1 erasure decodings 2f+1 signature verifications ((f+1) \cdot n \cdot \log(n)) hashes</td>
</tr>
<tr>
<td></td>
<td>Assume ((n,n/2)) erasure code</td>
<td></td>
</tr>
</tbody>
</table>

Assume \((n,n/2)\) erasure code
Outline

- Erasure codes & pollution attacks
- Our solution: Distillation codes
- Distillation codes for multicast authentication
Multicast authentication using SAIDA

- Given packets: \( p_1, p_2, p_3, p_4, \ldots, p_n \)
  
  signature string \( S = h(p_1) \| h(p_2) \| h(p_3) \| h(p_4) \| \cdots \| h(p_n) \| \text{Sign( )} \)

- Receiver can authenticate any packet with \( S \)
- How to deliver signature string?
  - Can’t send \( S \) with each packet: too big
  - Amortize cost by including piece with each packet
Multicast authentication using SAIDA

**Sender:**

1) Break stream into blocks of n packets

2) Compute a signature string and erasure encode

**Receiver:**

1) Erasure decode sig. string
2) Authenticate packets

Recall pollution attack: can’t reconstruct signature string
Distillation codes fix SAIDA

**Sender:**

1) Break stream into blocks of n packets

\[ p_1 \quad p_2 \quad p_3 \quad p_4 \quad \cdots \quad p_n \]

2) Compute a signature string and distillation encode

\[ S \]

**Receiver:**

1) Distillation decode sig. string

2) Authenticate packets

\[ X \quad X \quad \cdots \quad X \]

Use distillation codes instead of erasure codes
Packet authenticity: block signature

Computational DoS resistance: distillation codes

Tolerant to loss: distillation codes

See paper for proofs of security

Also applicable to Pannetrat-Molva

Overhead:

\[(n,t) = (128, 64)\]

RSA-1024 signature

→ SAIDA overhead: 22 bytes

Distillation code overhead: 43 bytes

Total Overhead: 65 bytes per packet
Performance Experiment

- Transmitted 4 Mb/s stream
  (128,64) erasure code
  RSA-1024 signatures
- Receiver: 2.4 GHz Pentium 4, 1GB RAM
- Worst case attack
- Varied attack factor $f = 0..10$
Performance

Graph showing the Receiver Processor Utilization (%) against the Attack Factor. The utilization increases as the attack factor increases, with fluctuations at certain points.
Conclusion

- Distillation codes extend erasure codes for malicious environments
- Distillation codes protect SAIDA and Pannetrat-Molva against DoS
- Applicable to other multicast and distributed storage protocols
Question