Integrity Assurance in Resource-Bounded Systems through Stochastic Message Authentication

Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos

Institute for Software Integrated Systems
Department of Electrical Engineering and Computer Science
Data Integrity

• Data integrity: assuring that data cannot be modified in an unauthorized and undetected manner

• Classic, non-resource-bounded example:

  desktop computer  HTTPS  webservice

Not really an issue these days, right?
Example of Data-Tampering

Traffic monitoring: Sensys Networks VDS240

- wireless vehicle detection system based on magnetic sensors embedded in roadways
- insecure communication protocol lacks integrity protection
- attacker may cause disastrous traffic congestions
Message Authentication

- Secret key
- Cryptographic computation
- Message
- Tag

Cryptographic computation is computationally expensive.
Insufficient resources

Limited amount of resources

Some messages are verified

Maximal achievable security

Sufficient resources

Messages are verified

Maximum security

Messages are not verified

Zero security
Stochastic Verification

select randomly which messages to verify

message1
message2

tag1
message1
tag2
m3554ge2

verify

verify

tag1
message1
tag2
m3554ge2

?
Applications

• In many scenarios, suboptimal data acquisition and control is **costly** but **not disastrous**
  • inefficient traffic control
  • incorrect smart-metering
  • ...

• Resource-bounded devices
  • battery-powered devices
  • legacy devices
  • low-performance devices
  • ...

• Comparison to lightweight cryptography
  • we build on well-known and widely deployed cryptographic primitives
  • our system adapts to arbitrary resource bounds
Game-Theoretic Model

“Which messages to verify?”

- Stackelberg security game with a defender and an attacker

Messages

- divided into classes
- messages of class $i$ may cause $L_i$ damage

1. Defender

- chooses verification probabilities $p_i$
- subject to computational budget constraint
  \[ \sum p_i T_i \leq B \]

where $T_i$ is the cost of verifying all messages of class $i$
1. Defender

2. Attacker
   - selects the number \( a_i \) of modified/forged messages for each class \( i \)
   - knows the defender’s strategy (i.e., \( p_i \) for every \( i \))

3. Payoffs
   
   \[
   \begin{align*}
   &\text{outcome:} \\
   \text{attack detected:} & 1 - \prod (1 - p_i)^{a_i} \\
   \text{attacker receives} & \text{punishment } F \\
   \text{attack not detected:} & \prod (1 - p_i)^{a_i} \\
   \text{defender loses /} & \text{attacker gains } \sum a_i L_i
   \end{align*}
   \]
Illustration of the Defender’s Payoff

Defender’s payoff

"region of deterrence"

$F = 0.5, L_1 = 1, L_2 = 3$
Deterrence Strategies

- Deterrence strategy: attacker’s best response is not to modify any messages

**Theorem:** The defender has a deterrence strategy if and only if

\[ B \geq \sum_i \frac{L_i}{L_i + F} T_i \]

and the minimal deterrence strategy is

\[ p_i = \frac{L_i}{L_i + F} \]
Non-Deterrence Strategies

$F = 0.5, L_1 = 1, L_2 = 3$
Continuous Relaxation

- No closed-form solution for the original model
- Continuous relaxation of the model
  - $a_i$ is continuous (i.e., $a_i = 1.5$ means that the attacker modifies one and a half messages)

**Theorem:** Optimal strategy in the continuous relaxation is

$$\frac{L_1}{\ln(1 - p_1)} = \frac{L_2}{\ln(1 - p_2)} = \cdots = \frac{L_C}{\ln(1 - p_C)}$$

$$\sum p_i T_i = B$$
Numerical Example Comparing Strategies

\[ F = 0.5, L_1 = 1, L_2 = 2, L_3 = 3, T_1 = T_2 = T_3 = 1 \]
Numerical Example Comparing Strategies

\[ F = 0.5, L_1 = 1, L_2 = 2, L_3 = 3, T_1 = T_2 = T_3 = 1 \]
Experiments

- Implementation and testing on an ATmega328P microcontroller

- Message authentication tag generation and verification:
  - HMAC (keyed-hash message authentication code)
  - Using the SHA-1 hash function

- Random number generation:
  - Linear-feedback shift register
Experimental Results

![Graph showing running time per message versus probabilities \( \sum p_i \).]
Resource-Bounded Senders

- So far, we have saved computation only at the receiver
- Two-way communication

“Could we also save computation when generating tags?”

- next: stochastic authentication tag generation

Up to 100% saving when receiving + 0% saving when sending up to 50% saving overall
Stochastic Message Authentication

Send a **random** subset of the messages with correct tags

- **Fake tags**
  - indistinguishable from correct tags for the attacker
  - distinguishable from incorrect tags for the receiver
  - computationally inexpensive to generate and verify

Detect modifications to messages with correct tags
Generating and Verifying Fake Tags

- Proof-of-concept algorithms based on the HMAC construction with a Merkle-Damgard hash function

Algorithm 1 MAC tag generation in partial HMAC

1: function GENERATE_TAG(K, m)
2:   rnd ← U(0, 1)
3:   if rnd ≤ p_{class}(m) then
4:     return HMAC(m)
5:   else
6:     return f(f(IV, K ⊕ ipad), m)
7: end function

Algorithm 2 MAC tag verification in partial HMAC

1: function VERIFY_TAG(K, m, t)
2:   t_f ← f( f(IV, K ⊕ ipad), m_1 )
3:   if t = t_f then
4:     return fake
5:   else
6:     t_c ← H((K ⊕ opad) | f(f(...f(t_f, m_2),...,m_n), length padding))
7:     if t = t_c then
8:       return correct
9:     else
10:    return incorrect
11: end if
12: end if
13: end function

- Implementation and testing show substantial savings for both the receiver and sender on an ATmega328P microcontroller
Conclusion

- Stochastic message verification
  - message authentication for resource-bounded devices
  - game-theoretic model for defending against worst-case attackers
  - experimental results confirm computational cost model

- Next: stochastic message authentication tag generation
  - allows saving computation for both sender and receiver
Thank you for your attention!

Questions?