Data Authentication

Ashish Kundu
cs.purdue.edu/~ashishk

CS 426
Oct 26, 2009

Outline

- Message Integrity
- Message Authentication
- Complex Messages
- Merkle Hash Technique
- Structural Signatures

Message Integrity

- Message $M \in \{0,1\}^*$

- $S \rightarrow R$:
  - $M, H(M)$
  - $H(M)$ can be used to verify if integrity of $M$ is preserved or not.
  - However, does not guarantee where $M$ comes from?

- $S \rightarrow \text{Attacker} \rightarrow R$
  - $M', H(M')$
  - $R$ cannot know whether it received the message that was supposed to come from $S$.

Message Authentication

- MAC, HMAC
  - Rely on symmetric key encryption

- Digital signature (public key)
  - Sign $M$: $\text{sig}(M)$
  - Does it work?
  - Modify $M$ and $\text{sig}(M)$ such that $\text{sig}(M')$ is a valid signature of $M'$.
    - Unpadded RSA, Malleable encryption
  - Always sign the hash of a message $H(M)$
    - $\text{Sign} H(M)$: Computationally infeasible to modify $H(M)$ in a valid way.
Complex Messages

- Set of messages
  - Ex: Online poker, online chess, online games, etc.
- Multi-set: multiple occurrence of the same element
- List: has an order between elements in a multi-set
  - Ex: Arrays, Blocks of data
- Trees
  - XML, Website, Webpage

Merkle Hash Technique (MHT)

Case: Only leaf nodes have contents.

\[
\begin{align*}
\text{e: } & H(e) & \text{f: } & H(f) \\
\text{c: } & H(c) & \text{d: } & H(H(e), H(f))
\end{align*}
\]

Case: Only leaf nodes have contents.
Merkle Hash Technique (MHT)

Case: Only leaf nodes have contents.

Merkle Hash Technique (MHT)

Case: All nodes have contents.

Extent of Leakage in MHT

Shared subtree
**Extent of Leakage in MHT**

- **Direct Leakages**
  1. Merkle hash (MH) of the root
  2. MH of each sibling of a received node
  3. Hash of the content of each ancestor
  4. MH of a child of each ancestor

- **Indirect (Inferred) Leakages**
  I. $(1) \Rightarrow$ if received subtree is (or is not) the complete tree (Comparison attack)
  II. $(2) \Rightarrow$
    a. Arity of the tree
    b. Number of children of each ancestor
  III. $(2)$ or $(3)$ or $(4) \Rightarrow$ If a hidden subtree same as a received subtree (Comparison attack)

**Motivating Example**

Leakages – How significant/sensitive?
Motivating Example: Semantic Leaks

Problem

Leakages

Problem Definition: Signature for Trees

- Guarantee integrity of any subtree $S$ in a tree $T$, while preventing the following information leakages

Problem Definition: Signature for Trees

- Guarantee integrity of any subtree $S$ in a tree $T$, while preventing the following information leakages

1) Signature of a node $y$ in $T$ but not in $S$

2) Existence of node $y$ in $T$ but not in $S$

3) Structural relationships not present in the subtree

4) For ordered trees, structural order between a node $x$ in $S$ and $y$ in $T$ but not in $S$
## Contributions

- Structural Signatures for Trees

- Provably binding
  - Integrity verification of trees – content, structure

- Provably hiding
  - Prevents leakage

- More efficient than Merkle hash technique

## Some Simple Observations

### Structural Signatures
Some Simple Observations

- A (rooted) tree can be represented by the post-order and pre-order sequences of its vertices [Das et al, ISPP’94]

- A (rooted) binary tree can be represented by the in-order and post-order or pre-order sequences of its vertices [Kamakoti & Pandu Rangan, IPL’92]

Traversal Numbers

- Post-order numbers
- Pre-order numbers

Randomized Traversal Numbers

- Traversal numbers are not secure:
  - Uniform distance, [1…n]
Randomized Traversal Numbers

- Traversal numbers are not secure:
  - Uniform distance, [1…n]
- Randomized traversal numbers
  - Random distance, Random numbers
  - Order is preserved

Randomized Traversal Numbers

- Traversal numbers are not secure:
  - Uniform distance, [1…n]
- Randomized traversal numbers
  - Random distance, Random numbers
  - Order is preserved
- Randomized Post-order Numbers (RPON)
- Randomized Pre-order Numbers (RRON)
- Randomized In-order Numbers (RION)

Randomized Traversal Numbers

- Technique-1: Sort N random numbers, N = number of nodes
- Technique-3: Order-preserving encryption
  - Rakesh Agarwal et al.
- Technique-2: R(i) = R(i-1) + w(i), w(i) = random
  - R(0) = w(0)
  - Is it secure?

Randomized Traversal Numbers

- Technique-1: Sort N random numbers, N = number of nodes
- Technique-3: Order-preserving encryption
  - Rakesh Agarwal et al.
- Technique-2: R(i) = R(i-1) + w(i), w(i) = random
  - R(0) = w(0)
  - Is it secure?
Randomized Traversal Numbers

Notations
- Tree: T(V, E)
- x, y, z ∈ V
- p_x : RPON of x
- r_x : RRON of x
- C_x : Content of x
- | : concatenation operation
- h: one-way collision-resistant cryptographic hash

Structural Signatures
- Structural Position of a Node ‘x’:
  \[ S_x = (p_x, r_x) \]
- Structural Signature of a Tree ‘T’:
  \[ G_T = h(S_1 | C_1 | S_2 | C_2 | \ldots | S_i | C_i | \ldots | S_n | C_n) \]
- Structural Signature of a Node ‘x’:
  \[ G_x = h(G_T | S_x | C_x) \]

Signing a Tree
- Signing the tree by Alice:
  \[ \text{Signs the tree } T \]
  \[ \text{Shares the } T \text{ with distributors (D).} \]
**Signing a Tree** \( T(V, E) \)

1. Compute the post-order and pre-order numbers for each node in \( T \).

2. For each node \( x \) in \( V \): transform the post-order and pre-order numbers into randomized post-order and pre-order numbers, such that
   - for unordered trees, RPON's and RRON's among the siblings do not need to preserve any order;
   - for ordered trees, RPON's and RRON's for all nodes, need to preserve the order.

3. Assign \((p_x, r_x)\) to \( x \) as its structural position \( S_x \).

4. Compute the structural signature of the tree \( T \), \( G_T \):
   \[ G_T = h(S_1 | C_1 | S_2 | C_2 | \ldots | S_i | C_i | \ldots | S_n | C_n) \]

5. For each node \( x \) in \( V \): compute the signature, \( G_x \):
   \[ G_x = h(G_T, S_x, C_x) \]

**Sharing the Tree**

- D sends a subtree S to Bob
  - Two strategies for sending S
    - Complete structure of S
    - Only nodes of S
  - Signature of tree \( T \): \( G_T \)
  - Signatures of nodes in S
  - Digest: \( \text{Digest}_s = h(G_1 | G_2 | \ldots | G_{|S|}) \)
Sharing the Tree

- D sends a subtree $S$ to Bob
  - Two strategies for sending $S$
    - Complete structure of $S$
    - Only nodes of $S$
  - Signature of tree $T$: $G_T$
  - Signatures of nodes in $S$
  - Digest $= h(G_1 | G_2 | \ldots | G_s)$.

- Bob verifies integrity of $S$.

Integrity

Sharing Complete Structure of $S$

Bob receives: $R(V_r, E_r)$.
1. Verifies the certificate of each node in $V_r$
2. Verifies content integrity
3. Verifies structural integrity
4. If all nodes are valid, Bob computes Digest$_r$
   a. If (Digest$_r$ = Digest$_s$) then no node is dropped.

Integrity Verification for Content

1. For each node $y$ in $V_r$,
   a. temp = $h(G_T | S_y | C_y)$
2. If temp is same as $G_y$, then the integrity of the node $y$ is verified.
Structural Integrity

Integrity Verification for Structural Relations

1. Carry out a pre-order traversal on R.
2. Let x be the parent of z;
   a. if ((p_x ≤ p_z) or (r_x ≥ r_z)), then parent-child relationship between x and z is incorrect.

Sharing only the Nodes of S

Bob receives: V_r

1. Verify the certificate of each node in V_r
2. Verify the integrity of content of each node.
3. Reconstructs the subtree R(V_r, E_r) using Structural Positions
   a. Any relation (edge) with an invalid node: invalid.
4. If all nodes are valid, Bob computes Digest_r
   a. If (Digest_r = Digest_s) then no node is dropped.

Security Analysis

- Provably Binding
  - Structural signatures detect violation(s) to integrity of content and structure.
Security Analysis

- Provably Binding
  - Structural signatures detect violation(s) to integrity of content and structure.

- Provably Hiding
  - Structural signatures do not leak (or leak negligible amount of information about)

Complexity Analysis

- Cost of Signature Generation
  - Time: $O(|V|)$
  - Storage: $O(|V|) = O(|S_x| + |G_x| + |G_T|))$

- Cost of Distribution
  - Substructure
    - Communication: $2|V| - 1$, Storage: $2|V| - 1$
  - Only the Nodes: 50% reduction
    - Communication: $|V|$, Storage: $|V|$

- Cost of Integrity Verification
  - Time: $O(1)$ for each node, structural relation/order.
  - Time: $O(|V_k|)$, for the received subtree
Complexity Analysis

- Cost of Signature Generation
  - Time: $O(|V|)$, Storage: $O(|V|)$ ($= O(|S_x| + |G_x| + |G_T|)$)

- Cost of Distribution
  - Substructure
    - Communication: $2|V| - 1$, Storage: $2|V| - 1$
  - Only the Nodes: 50% reduction
    - Communication: $|V|$, Storage: $|V|$

- Cost of Integrity Verification
  - Time: $O(1)$ for each node, structural relation/order.
  - Time: $O(|V_R|)$, for the received subtree

Performance: Infrastructure

- Trees
  - 0-65535 nodes, 2-ary ordered complete trees, height 1-16.
- Language
  - Java J2SE5.0
- Machine
  - IBM T42 512MB RAM

Signing: Height Vs. Time (Nano-secs)

- Time taken more than MHT: 0.13 seconds
- Negligible: Signed once, used many times.

Signing: No. of Nodes Vs. Time (Nano-secs)
**Integrity: Height Vs. Time (Micro-secs)**

- More efficient than MHT
- Verified by many users: so gain is multiplied.

**Integrity: No. of nodes Vs. Time (Micro-secs)**

- More efficient than MHT
- Verified by many users: so gain is multiplied.

**Dynamic Updates**

- Node/subtree insertion to T: node signature using $G_T$
- Deletion: $G_T$ is not updated
- Edge insertion/deletion

**Dynamic Updates**

- Node/subtree insertion: node signature using $G_T$
  - $O(1)$ per node
- Deletion: no change to $G_T$
  - $O(1)$ per node
- Edge insertion/deletion
  - $O(1)$ per edge
Dynamic Updates

- Node/subtree insertion: node signature using $G_T$
  - $O(1)$ per node
- Deletion: no change to $G_T$
  - $O(1)$ per node
- Edge insertion/deletion
  - $O(1)$ per edge

MHT and related techniques:

- Updates: $O(\log(n))$ per node/edge

Use of $G_T$ (signature of tree $T$) for the updated tree $T'$:

- Advantage: Prevents leakage related to updates
- Drawback: $G_T$ does not reflect updates

Summary

- Stronger security guarantees than MHT and related techniques
  - Hiding
  - Binding
- More efficient
  - Distribution
  - Integrity verification
  - Similar cost for signing process
- Simple – an important security criteria
  - To understand and implement

Summary

- Stronger security guarantees than MHT and related techniques
  - Hiding
  - Binding
- More efficient
  - Distribution
  - Integrity verification
  - Similar cost for signing process
- Simple – an important security criteria
  - To understand and implement

Pervasive Devices/Applications
### Some Applications

- Database integrity & confidentiality
- Index integrity & confidentiality
- XML Signatures: Integrity & confidentiality of XML
- Memory integrity
- Directory signatures

### Related Work: Authentication of Trees

- [Merkle’79] Merkle hash technique (MHT): widely used
  - Leaks content and structural information: $O(\log(n))$, $n$: no. of vertices
  - [Buldas & Laur’07] binding but not hiding;
    - no solution proposed.

- Well-known applications of Merkle hash technique:
  - [Pang & Tan’08] Verifying completeness of relational query answers
  - [Martel et al’04] Search DAGS
  - [Bertino et al’04] Selective & authentic third party XML Dissemination
  - [Devanbu et al’01, ’03] XML authentication

### Related Work: Authentication

- [Merkle’79] Merkle hash technique (MHT): Widely used
  - Leaks content and structural information: $O(\log(n))$, $n$: no. of vertices
  - [Buldas & Laur’07] binding but not hiding;
    - no solution proposed.

- Well-known applications of Merkle hash technique:
  - [Pang & Tan’08] Verifying completeness of relational query answers
  - [Martel et al’04] Search DAGS
  - [Bertino et al’04] Selective & authentic third party XML Dissemination
  - [Devanbu et al’01, ’03] XML authentication

### Related Work: Information Leakages

- [Wang et al, VLDB’05] Structural relations (edges) in XML: Not accessible by a user
  - Does not address leakages due to integrity verification

- Active area of research: last two decades
  - Privacy-preserving databases
    - [Chatvichienchai et al, DEXA’06] [Rastogi et al, VLDB’07] [Wong et al, VLDB’07] [Zhang and Zhao, VLDB’05]
  - [Irwin & Yu, CCS’05] Automated Trust Negotiation
  - [Dodis & Smith, STOC’05] Error correction
  - [Chatvichienchai et al, DEXA’06] Information leakage in updating XML
    - Leakage from integrity verification of XML – not addressed
and questions?

Ashish Kundu
www.cs.purdue.edu/~ashishk