Attribute-Based Signatures for Unbounded Languages from Standard Assumptions

Yusuke Sakai (AIST, Japan)
Shuichi Katsumata (AIST, Japan / U. Tokyo, Japan)
Nuttapong Attrapadung (AIST, Japan)
Goichiro Hanaoka (AIST, Japan)
Our Contribution

• Propose attribute-based signature scheme for *Turing machines*
  – A key-policy variant
  – The policy is described by a Turing machine (TM)
  – The attribute is an input to a TM

The scheme allows policies that accept *unbounded* inputs!
Agenda

- Attribute-Based Signatures
- Security Requirement
- Certificate Approach
- Idea 1: History of Computation
- Idea 2: Locality of Rewriting
- Overview of the Scheme
- Conclusion
Attribute-Based Signatures (ABS)

\[ P \leftrightarrow \text{sk}_P \]

\[ \text{sk}_P, \text{sk}_{P'} \]

\[
\begin{array}{cccccc}
    & a & b & c & d & e \\
\end{array}
\]

\[ q \]
Attribute-Based Signatures

\[ \sigma \leftarrow \text{AttrSign}(pp, sk_p, M, x) \]
Attribute-Based Signatures

\[ \sigma \text{ is made by someone whose policy } P \text{ satisfy } P(x) = 1 \]
Agenda

• Attribute-Based Signatures
• **Security Requirement**
• Certificate Approach
• Idea 1: History of Computation
• Idea 2: Locality of Rewriting
• Overview of the Scheme
• Conclusion
Anonymity

Cannot tell who made \( \sigma \) among signers who satisfy \( P(x) = 1 \)
Unforgeability

Cannot make valid $\sigma$ if $P(x) = 0$

$sk_P, M, x, \sigma$

$sk_{P'}$
Agenda

• Attribute-Based Signatures
• Security Requirement
• Certificate Approach
• Idea 1: History of Computation
• Idea 2: Locality of Rewriting
• Overview of the Scheme
• Conclusion
Certificate Approach (1/2)

Each signer receives \textit{a signature on his policy}

\[
\begin{align*}
\sk_P &= \theta_P = \text{Sign}(\msk, \ P) \\
\sk_{P'} &= \theta_{P'} = \text{Sign}(\msk, \ P')
\end{align*}
\]
Certificate Approach (2/2)

\[ \text{sk}_P = \theta_P = \text{Sign}(\text{msk}, P) \]

\[ \text{sk}_{P'} = \theta_{P'} = \text{Sign}(\text{msk}, P') \]

Prove knowledge of \((P, \theta)\):
(1) \(\text{Verify}(P, \theta) = 1\)
(2) \(P(x) = 1\)
Difficulty

Prove knowledge of \((P, \theta_p)\):
(1) \(\text{Verify}(P, \theta_x) = 1\)
(2) \(P(x) = 1\)

• How to prove the complex condition \(P(x) = 1\)
  – Remind that \(P\) is a Turing machine

• General zero-knowledge is inefficient, so we will \textit{decompose} the statement
Agenda

• Attribute-Based Signatures
• Security Requirement
• Certificate Approach
• Idea 1: History of Computation
• Idea 2: Locality of Rewriting
• Overview of the Scheme
• Conclusion
Idea: History of Computation

• While a TM’s computation is complex, the computation proceeds sequentially
• The computation defines a sequence of “snapshots” of the machine

\[ q_0 \rightarrow w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_4 \rightarrow w_5 \]
Idea: History of Computation

• While a TM’s computation is complex, the computation proceeds sequentially.
• The computation defines a sequence of “snapshots” of the machine.
Idea: History of Computation

• While a TM’s computation is complex, the computation proceeds sequentially
• The computation defines a sequence of “snapshots” of the machine
Idea: History of Computation

• While a TM’s computation is complex, the computation proceeds sequentially
• The computation defines a sequence of “snapshots” of the machine
Implement the Certificate Approach

• Using the sequence of the snapshot \((s_1, ..., s_T)\) we can rephrase the proof as follows:

  **Prove knowledge of \((s_1, ..., s_T)\):**

  (1) \(s_i \rightarrow s_{i+1}\) follows the transition function

• To enforce validity of transition, the KGC signs on all possible valid transition:

\[
\theta[s,s'] \leftarrow \text{Sign}(msk, (s,s'))
\]

\[\forall s \rightarrow s': \text{valid transition}\]
Signing Every Possible Transition

Prove knowledge of \((s_0, s_1, \theta_1)\):

\[ \text{Verify}(\text{vk}, (s_0, s_1), \theta_1) = 1 \]
Signing Every Possible Transition

Prove knowledge of \((s_0, s_1, \theta_1)\):
Verify\((vk, (s_0, s_1), \theta_1) = 1\)

Prove knowledge of \((s_1, s_2, \theta_2)\):
Verify\((vk, (s_1, s_2), \theta_2) = 1\)
Prove knowledge of \((s_1, \ldots, s_T, \theta_1, \ldots, \theta_T)\):

(1) \(\text{Verify}((s_{i-1}, s_i), \theta_i) = 1\)
Main Difficulty

Prove knowledge of \((s_1, \ldots, s_T, \theta_1, \ldots, \theta_T)\):

(1) Verify\(((s_{i-1}, s_i), \theta_i) = 1\)

- Possible pairs of snapshots are infinitely many,
  - since snapshots have unbounded lengths
- We further decompose this condition
Agenda

- Attribute-Based Signatures
- Security Requirement
- Certificate Approach
- Idea 1: History of Computation
- Idea 2: Locality of Rewriting
- Overview of the Scheme
- Conclusion
Configuration

- A snapshot is encoded into a single string, \textit{configuration}, consisting of (1) the content of the tape interleaved with (2) the state symbol $q$ – the position of $q$ encodes the position of the head.

\[ w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ ... \ w_1 \ w_2 \ q \ w_3 \ w_4 \ w_5 \ ... \]
Locality of Rewriting

- Each symbol in a new configuration is determined by neighbors in the old configuration.
- Four neighbors are sufficient for any case.

(step t: \( w_1, w_2, q, w_3, w_4, w_5 \))

(step t+1: \( w_1, q', w_2, w'_3, w_4, w_5 \))
The General Cases

• Each cell will be determined by the four neighbors in the old configuration.
Enforcing Validity of Transition

• To enforce validity of transition
  KGC signs on every valid 5-tuple:
  $$\theta[w_1, w_2, w_3, w_4, u]$$
  $$\leftarrow \text{Sign}(\text{msk}, (w_1, w_2, w_3, w_4, u))$$

• The signer proves the knowledge of signature
  for every symbol in the new configuration

old:  \[ w_1 \quad w_2 \quad q \quad w_3 \quad w_4 \quad w_5 \]

new:  \[ w_1 \quad q' \quad w_2 \quad w'_3 \quad w_4 \quad w_5 \]
Enforcing Validity of Transition

- To enforce validity of transition, KGC signs on every valid 5-tuple:
  \[ \theta[w_1, w_2, w_3, w_4, u] \]

- The signer proves the knowledge of \( \theta \) for every symbol in the new configuration:

Prove knowledge of \((w_1, w_2, q, w_3, q', \theta_1)\): Verify(vk, \((w_1, w_2, q, w_3, q'), \theta_1\)) = 1

old: \( w_1 \quad w_2 \quad q \quad w_3 \quad w_4 \quad w_5 \)

new: \( w_1 \quad q' \quad w_2 \quad w'_3 \quad w_4 \quad w_5 \)
Enforcing Validity of Transition

- To enforce validity of transition, KGC signs on every valid 5-tuple:

\[ \theta[w_1, w_2, w_3, w_4, u] \]

- The signer proves the knowledge of \( \theta \) for every symbol in the new configuration:

\[ \text{Prove knowledge of } (w_2, q, w_3, w_4, w_2, \theta_2) : \]

\[ \text{Verify(vk, } (w_2, q, w_3, w_4, w_2), \theta_2) = 1 \]

old: \( w_1 \quad w_2 \quad q \quad w_3 \quad w_4 \quad w_5 \)

new: \( w_1 \quad q' \quad w_2 \quad w'_3 \quad w_4 \quad w_5 \)
Enforcing Validity of Transition

• To enforce validity of transition KGC signs on every valid 5-tuple:

\[ \theta[w_1, w_2, w_3, w_4, u] \]

• The signer proves the knowledge of \( \theta \) for every symbol in the new configuration:

Prove knowledge of \( (q, w_3, w_4, w_5, w'_3, \theta_3) \):

\[ \text{Verify}(vk, (q, w_3, w_4, w_5, w'_3), \theta_3) = 1 \]
Agenda

• Attribute-Based Signatures
• Security Requirement
• Certificate Approach
• Idea 1: History of Computation
• Idea 2: Locality of Rewriting
• Overview of the Scheme
• Conclusion
Putting All Together

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$w'_5$</th>
<th>$q_2$</th>
<th>$w_6$</th>
<th>$w_7$</th>
<th>$w_8$</th>
<th>$w_9$</th>
<th>$w_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Proves the knowledge of signatures on the neighbors (quadratic in running time of TM)
• Every symbol is hidden as a witness

$\text{Verify}((w'_5, q_2, w_6, w_7, w'_5), \theta) = 1$
The Scheme

• Setup:
  – crs $\leftarrow$ CRSGen($1^k$), (vk, sk) $\leftarrow$ SigKg($1^k$)

• KeyGen:
  – for every valid 5-tuple $(w_1, w_2, w_3, w_4, u)$:
    • $\theta_{[w_1, w_2, w_3, w_4, u]} \leftarrow$ SigSign(sk, $(w_1, w_2, w_3, w_4, u)$)

• Sign: \{w_{i,j}\}_{i,j}: 2D arrangement of configurations
  – $\pi_{i,j} \leftarrow$ Prove(crs, $(w_{i-1,j-2}, w_{i-1,j-1}, w_{i-1,j}, w_{i+1,j}, w_{i,j}, \theta)$)

• Verify: for all (i,j) verify $\pi_{i,j}$
Main Theorem

**Theorem** If the non-interactive proof system is witness-indistinguishable and extractable, the signature scheme is unforgeable, the proposed scheme is anonymous and unforgeable.

Instantiate this with GS proofs in SXDH setting and structure-preserving signatures.

**Theorem** If SXDH assumption holds, the proposed scheme satisfies anonymity and unforgeability.
### Efficiency

<table>
<thead>
<tr>
<th>Signing key length</th>
<th>Signature length</th>
<th>Verification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(</td>
<td>\Gamma</td>
<td>^4)$</td>
</tr>
</tbody>
</table>

$|\Gamma|$: The size of the tape alphabet  
$T$: The running time of the TM

- The scheme is reasonably efficient!
Agenda

• Attribute-Based Signatures
• Security Requirement
• Certificate Approach
• Idea 1: History of Computation
• Idea 2: Locality of Rewriting
• Overview of the Scheme
• Conclusion
Summary

• Proposed attribute-based signature scheme for *unbounded languages (Turing machines)*
  – Uniform model of computation as the policy
  – No bound on the sizes of both TMs and attributes
  – Can be instantiated from the SXDH assumption in bilinear groups