Cryptographic Solutions for Data Integrity in the Cloud

David Mandell Freeman

Stanford University, USA

Stanford Computer Forum
2 April 2012
Homomorphic encryption allows users to delegate computation while ensuring secrecy.
Homomorphic encryption allows users to delegate computation while ensuring secrecy.

\[ \text{pk} \]

\[ \text{The Cloud} \]

\[ c_i = \text{encryption of } i\text{th score} \]

\[ c = \text{encryption of mean} \]

Validity: \[ c \] decrypts to the correct mean.

Security: no adversary can obtain any info about scores.

Length efficiency: \[ c \] is short.

Privacy: decrypted mean reveals nothing else about data.

David Mandell Freeman  
Data Integrity in the Cloud
Homomorphic encryption allows users to delegate computation while ensuring secrecy.

$c_i =$ encryption of $i$th score
Homomorphic encryption allows users to delegate computation while ensuring secrecy.

\[ c_i = \text{encryption of } i\text{th score} \]

<table>
<thead>
<tr>
<th>Student</th>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>( c_1 )</td>
</tr>
<tr>
<td>Becky</td>
<td>( c_2 )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>( c_k )</td>
</tr>
</tbody>
</table>

pk encrypted grades

The Cloud

sk
Homomorphic encryption allows users to delegate computation while ensuring secrecy.

$c_i = \text{encryption of } i\text{th score}$

<table>
<thead>
<tr>
<th>Student</th>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>$c_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>$c_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>$c_k$</td>
</tr>
</tbody>
</table>

$pk \xrightarrow{\text{encrypted grades}} \text{The Cloud}$

$sk \xleftarrow{\text{mean?}}$
Homomorphic encryption allows users to delegate computation while ensuring secrecy.

\( c_i = \text{encryption of } i\text{th score} \)

\( c = \text{encryption of mean} \)

<table>
<thead>
<tr>
<th>Student</th>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>( c_1 )</td>
</tr>
<tr>
<td>Becky</td>
<td>( c_2 )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>( c_k )</td>
</tr>
</tbody>
</table>

Validity: \( c \) decrypts to the correct mean.

Security: no adversary can obtain any info about scores.

Length efficiency: \( c \) is short.

Privacy: decrypted mean reveals nothing else about data.
**Homomorphic encryption** allows users to delegate computation while ensuring **secrecy**.

The Cloud

<table>
<thead>
<tr>
<th>Student</th>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>$c_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>$c_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>$c_k$</td>
</tr>
</tbody>
</table>

$c_i =$ encryption of $i$th score  \hspace{1cm} c =$ encryption of mean

- **Validity**: $c$ decrypts to the correct mean.
- **Security**: no adversary can obtain any info about scores.
- **Length efficiency**: $c$ is short.
- **Privacy**: decrypted mean reveals nothing else about data.
Homomorphic signatures allow users to delegate computation while ensuring integrity.
Homomorphic signatures allow users to delegate computation while ensuring integrity.
Homomorphic signatures allow users to delegate computation while ensuring integrity.

The Cloud

$sk$

$\sigma = \text{signature on } (\text{grades}, 87.3, \text{mean})$

$\sigma = \text{signature on } (\text{grades}, 91, \text{Adam})$
Homomorphic signatures allow users to delegate computation while ensuring integrity.

\[ \sigma_1 = \text{signature on } \langle \text{grades}, 91, \text{“Adam”} \rangle \]

The Cloud

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>( \sigma_1 )</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>( \sigma_2 )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>( \sigma_k )</td>
</tr>
</tbody>
</table>
Homomorphic signatures allow users to delegate computation while ensuring integrity.

The Cloud

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>$\sigma_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>$\sigma_2$</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>$\sigma_k$</td>
</tr>
</tbody>
</table>

$\sigma_1 = \text{signature on } (\text{“grades”}, 91, \text{“Adam”})$
Homomorphic signatures allow users to delegate computation while ensuring integrity.

\[ \sigma_1 = \text{signature on } (\text{“grades”, 91, “Adam”}) \]

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>( \sigma_1 )</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>( \sigma_2 )</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>( \sigma_k )</td>
</tr>
</tbody>
</table>
Homomorphic signatures allow users to delegate computation while ensuring integrity.

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>$\sigma_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>$\sigma_2$</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>$\sigma_k$</td>
</tr>
</tbody>
</table>

$\sigma_1 = \text{signature on} \ (\text{"grades"}, 91, \text{"Adam"})$

$\sigma = \text{signature on} \ (\text{"grades"}, 87.3, \text{"mean"})$
What properties do we want the derived signature $\sigma$ to have?

\[
\sigma = \text{signature on ("grades", 87.3, "mean")}
\]
What properties do we want the derived signature $\sigma$ to have?

$\sigma = \text{signature on ("grades", 87.3, "mean")}$

- **Validity**: $\sigma$ authenticates 87.3 as the mean, and that the mean was computed correctly.
What properties do we want the derived signature $\sigma$ to have?

$\sigma =$ signature on

(“grades”, 87.3, “mean”)

1. **Validity**: $\sigma$ authenticates 87.3 as the mean, and that the mean was computed correctly.

2. **Unforgeability**: no adversary can produce a $\sigma^*$ that authenticates a different mean.
What properties do we want the derived signature $\sigma$ to have?

$\sigma = \text{signature on} $

("grades", 87.3, "mean")

1. **Validity**: $\sigma$ authenticates 87.3 as the mean, and that the mean was computed correctly.

2. **Unforgeability**: no adversary can produce a $\sigma^*$ that authenticates a different mean.

3. **Length efficiency**: $\sigma$ is short.
Properties of Homomorphic Signatures

What properties do we want the derived signature $\sigma$ to have?

$\sigma = \text{signature on} \ (\text{“grades”}, 87.3, \text{“mean”})$

1. **Validity**: $\sigma$ authenticates 87.3 as the mean, and that the mean was computed correctly.

2. **Unforgeability**: no adversary can produce a $\sigma^*$ that authenticates a different mean.

3. **Length efficiency**: $\sigma$ is short.

4. **Privacy**: $\sigma$ reveals nothing about data other than the mean.
Trivial Solution: Verify the Whole Database

The Cloud

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>$\sigma_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>$\sigma_2$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>$\sigma_k$</td>
</tr>
</tbody>
</table>

Bob authenticates database, then computes mean himself.

$\sigma = \{\text{all scores and all signatures}\}$
Trivial Solution: Verify the Whole Database

Signed grades:

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>91</td>
<td>$\sigma_1$</td>
</tr>
<tr>
<td>Becky</td>
<td>73</td>
<td>$\sigma_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Kevin</td>
<td>84</td>
<td>$\sigma_k$</td>
</tr>
</tbody>
</table>

Bob authenticates database, then computes mean himself.

$\sigma = \{\text{all scores and all signatures}\}$

Does not have length efficiency and privacy properties:
- $\sigma$ is as big as entire database.
- Bob learns the whole database
Messages \((m_1, \ldots, m_k)\) grouped together into *files*.

- \(\text{Sign}_{sk}(m_i) \rightarrow \text{signature } \sigma_i \text{ on } i\text{th message}\)
- \(\text{Eval}_{pk}(\sigma_1, \ldots, \sigma_k, f) \rightarrow \text{signature } \sigma \text{ on } f(m_1, \ldots, m_k)\)
- \(\text{Verify}_{pk}(m, \sigma, f) \rightarrow \text{accept/reject}\)
Nontrivial Solution: Move Work to the Cloud

Messages \((m_1, \ldots, m_k)\) grouped together into *files*.

- \(\text{Sign}_{sk}(m_i) \rightarrow \) signature \(\sigma_i\) on \(i\)th message
- \(\text{Eval}_{pk}(\sigma_1, \ldots, \sigma_k, f) \rightarrow \) signature \(\sigma\) on \(f(m_1, \ldots, m_k)\)
- \(\text{Verify}_{pk}(m, \sigma, f) \rightarrow \) accept/reject

**Correctness:** Verify accepts if \(m = f(m_1, \ldots, m_k)\).

**Security goal:** no adversary can authenticate \((m', f)\) for \(m' \neq f(m_1, \ldots, m_k)\).

**Privacy goal:** if \(f(m_1, \ldots, m_k) = f(\hat{m}_1, \ldots, \hat{m}_k) = m\), no one can tell which data set \(\sigma\) was derived from.
Messages \((m_1, \ldots, m_k)\) grouped together into files.

- \(\text{Sign}_{sk}(m_i) \rightarrow \) signature \(\sigma_i\) on \(i\)th message
- \(\text{Eval}_{pk}(\sigma_1, \ldots, \sigma_k, f) \rightarrow \) signature \(\sigma\) on \(f(m_1, \ldots, m_k)\)
- \(\text{Verify}_{pk}(m, \sigma, f) \rightarrow \) accept/reject

**Correctness:** Verify accepts if \(m = f(m_1, \ldots, m_k)\).

**Security goal:** no adversary can authenticate \((m', f)\) for \(m' \neq f(m_1, \ldots, m_k)\).

**Privacy goal:** if \(f(m_1, \ldots, m_k) = f(\hat{m}_1, \ldots, \hat{m}_k) = m\), no one can tell which data set \(\sigma\) was derived from.

**Linearly homomorphic** signatures:

- messages \(m_i\) are vectors.
- functions \(f\) are linear combinations.
What are homomorphic signatures good for?

<table>
<thead>
<tr>
<th>$f$</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td>Mean, Fourier transform, Network coding, Regression models (time series data)</td>
</tr>
</tbody>
</table>
What are homomorphic signatures good for?

<table>
<thead>
<tr>
<th>$f$</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Mean</td>
</tr>
<tr>
<td>functions</td>
<td>Fourier transform</td>
</tr>
<tr>
<td></td>
<td>Network coding</td>
</tr>
<tr>
<td></td>
<td>Regression models (time series data)</td>
</tr>
<tr>
<td>Subsets</td>
<td>Message redaction</td>
</tr>
</tbody>
</table>
What are homomorphic signatures good for?

<table>
<thead>
<tr>
<th>$f$</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td>Fourier transform</td>
</tr>
<tr>
<td></td>
<td>Network coding</td>
</tr>
<tr>
<td></td>
<td>Regression models (time series data)</td>
</tr>
<tr>
<td>Subsets</td>
<td>Message redaction</td>
</tr>
<tr>
<td>Polynomials</td>
<td>Standard deviation &amp; higher moments</td>
</tr>
<tr>
<td></td>
<td>Regression models (general data)</td>
</tr>
</tbody>
</table>
What are homomorphic signatures good for?

<table>
<thead>
<tr>
<th>$f$</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td>Mean, Fourier transform, Network coding, Regression models (time series data)</td>
</tr>
<tr>
<td>Subsets</td>
<td>Message redaction</td>
</tr>
<tr>
<td>Polynomials</td>
<td>Standard deviation &amp; higher moments, Regression models (general data)</td>
</tr>
<tr>
<td>Arbitrary circuits</td>
<td>Non-linear estimators and regression, Data mining (decision trees, SVM, etc.)</td>
</tr>
</tbody>
</table>
Application: Least Squares Fits
For a data set \( \{(x_i, y_i)\}_{i=1}^{k} \), the degree \( d \) least squares fit is the polynomial

\[ f(x) = c_0 + c_1 x + \cdots + c_d x^d \]

that “best” approximates the observed \( y \) values.
For a data set \( \{(x_i, y_i)\}_{i=1}^{k} \), the degree \( d \) least squares fit is the polynomial

\[
f(x) = c_0 + c_1 x + \cdots + c_d x^d
\]

that “best” approximates the observed \( y \) values.

\[
(X^T X)^{-1} X^T \vec{y}
\]

\( X \) = matrix of \( x \) values,
\( \vec{y} \) = vector of \( y \) values.
For a data set \( \{(x_i, y_i)\}_{i=1}^{k} \), the degree \( d \) least squares fit is the polynomial

\[
f(x) = c_0 + c_1 x + \cdots + c_d x^d
\]

that “best” approximates the observed \( y \) values.

U.S. population by year

\[
y = f(x) = c_0 + c_1 x + c_2 x^2
\]
Least Squares Fits

For a data set \( \{(x_i, y_i)\}_{i=1}^k \), the degree \( d \) least squares fit is the polynomial

\[
f(x) = c_0 + c_1 x + \cdots + c_d x^d
\]

that “best” approximates the observed \( y \) values.

U.S. population by year

\[
y = f(x) = c_0 + c_1 x + c_2 x^2
\]

Formula:

\[
(c_0, c_1, c_2) = (X^t X)^{-1} X^t \tilde{y}
\]

\( X \) = matrix of \( x \) values,

\( \tilde{y} \) = vector of \( y \) values.
Authenticating a least-squares fit

U.S. population by year

$y = f(x)$

$= c_0 + c_1 x + c_2 x^2$

Formula:

$(c_0, c_1, c_2) = (X^t X)^{-1} X^t \vec{y}$

$X = \text{matrix of } x \text{ values,}$

$\vec{y} = \text{vector of } y \text{ values.}$
Authenticating a least-squares fit

U.S. population by year

\[ y = f(x) = c_0 + c_1 x + c_2 x^2 \]

Formula:

\[
(c_0, c_1, c_2) = (X^t X)^{-1} X^t \vec{y}
\]

\( X = \) matrix of \( x \) values,
\( \vec{y} = \) vector of \( y \) values.

For time series \( x \) values, \( \vec{c} \) is linear function of \( y \) values.
Authenticating a least-squares fit

U.S. population by year

\[ y = f(x) = c_0 + c_1 x + c_2 x^2 \]

Formula:

\[ (c_0, c_1, c_2) = (X^t X)^{-1} X^t \vec{y} \]

For time series \( x \) values, \( \vec{c} \) is linear function of \( y \) values.

- Census bureau stores signed population counts on server using linearly homomorphic signature.
- Server can authenticate coefficients of least-squares fit.
State of the art

How can we compute on encrypted or authenticated data?

<table>
<thead>
<tr>
<th>$\mathcal{F}$</th>
<th>Hom. encryption</th>
<th>Hom. signatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polynomials</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arbitrary circuits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State of the art

How can we compute on encrypted or authenticated data?

<table>
<thead>
<tr>
<th>$F$</th>
<th>Hom. encryption</th>
<th>Hom. signatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td>[GM82], [B88], [P99], others</td>
<td></td>
</tr>
<tr>
<td>Subsets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polynomials</td>
<td>[BGN05], [GHV10]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(quadratic)</td>
<td></td>
</tr>
<tr>
<td>Arbitrary circuits</td>
<td>[G09], [DGHV10], [BV11], [BGV12], [B12]</td>
<td></td>
</tr>
</tbody>
</table>
How can we compute on encrypted or authenticated data?

<table>
<thead>
<tr>
<th>( \mathcal{F} )</th>
<th>Hom. encryption</th>
<th>Hom. signatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear functions</td>
<td>[GM82], [B88], [P99], others</td>
<td>[KFM04], [CJL06], [ZKMH07], [BFKW09], [GKKR10], [BF11a], [F12]</td>
</tr>
<tr>
<td>Subsets</td>
<td></td>
<td>[JMSW02], [ABC+12], others</td>
</tr>
<tr>
<td>Polynomials</td>
<td>[BGN05], [GHV10] (quadratic)</td>
<td>[BF11b] (bounded degree)</td>
</tr>
<tr>
<td>Arbitrary circuits</td>
<td>[G09], [DGHV10], [BV11], [BGV12], [B12]</td>
<td></td>
</tr>
</tbody>
</table>
Recent Advances in Homomorphic Signatures
Improved security for \textit{linearly} homomorphic signatures:

2. Systems resistant to a stronger adversary.

Achieved via a generic construction from existing (ordinary) signatures satisfying certain properties.

- Single framework gives many homomorphic schemes.
- Users can choose based on security/efficiency tradeoffs.
Security paradigm:
Show that any adversary that can forge a signature can be used to solve a computational problem believed to be hard.

E.g.:
- **Factoring**: given $N = pq$, find $p, q$.
- **RSA**: given $(N, x, e)$, find $x^{1/e} \mod N$.
- **Strong RSA**: given $(N, x)$, find some $(e, x^{1/e} \mod N)$.
Security paradigm:
Show that any adversary that can forge a signature can be used to solve a computational problem believed to be hard.

E.g.:

- **Factoring**: given $N = pq$, find $p, q$.
- **RSA**: given $(N, x, e)$, find $x^{1/e} \mod N$.
- **Strong RSA**: given $(N, x)$, find some $(e, x^{1/e} \mod N)$.

Hierarchy of problems: solving (1) $\Rightarrow$ solving (2) $\Rightarrow$ solving (3).
(converse not known to be true or false)

Top of hierarchy: stronger security guarantees.
Bottom of hierarchy: more efficient constructions.
Previous constructions of linearly homomorphic signatures:

- [GKKR10]: Based on RSA, requires “ideal” hash function (indistinguishable from random).
- [CFW11]: Based on Strong RSA, uses “real world” hash function (collision-resistant).
Previous constructions of linearly homomorphic signatures:

- **[GKKR10]**: Based on RSA, requires “ideal” hash function (indistinguishable from random).
- **[CFW11]**: Based on Strong RSA, uses “real world” hash function (collision-resistant).

Our construction:

- **[F12]**: Based on RSA, uses “real world” hash function.

Similar results using hierarchy of “Diffie-Hellman” problems.
Contrubution 2: Stronger Adversary

Security model for homomorphic signatures:

\[
\text{Chall.} \quad \text{sk} \quad \text{pk} \quad \rightarrow \quad \text{tuple} \quad (F, i, m_i) \quad \leftarrow \quad \text{sig} \quad \sigma \quad F_i \quad \rightarrow \quad \text{\{\}}
\]

\[
\text{repeat}
\]

\[
\text{Forgery} \quad F^*, m^*, \sigma^*, f \quad \leftarrow \quad \text{Adversary}
\]

 Forgery is a valid signature \(\sigma^*\) on \((F^*, m^*, f)\) with \(m^* \neq f\) (messages in file \(F^*\)).

Original adversary: must query entire files at once.

Stronger adversary: adaptively queries one message at a time from any file.

Our new schemes are secure against the stronger adversary.

David Mandell Freeman

Data Integrity in the Cloud
Contribution 2: Stronger Adversary

Security model for homomorphic signatures:

Chall.\[\text{sk}\] → \{\text{pk}\} → \text{tuple} (\text{F}, i, m_i) ← \text{sig} \sigma \text{F}_i → \}

\[\begin{align*}
&\text{Adversary} \\
&\text{Forgery is a valid signature } \sigma^* \text{ on } (\text{F}^*, m^*, f) \text{ with } m^* \neq f \text{ (messages in file } F^*). \\
\end{align*}\]
Contribution 2: Stronger Adversary

Security model for homomorphic signatures:

Chall. \[\begin{array}{c}
\text{sk} \\
pk
\end{array} \rightarrow \text{Adversary}
\]

 Forgery is a valid signature \(\sigma^*\) on \((F^*, m^*, f)\) with \(m^* \neq f\) (messages in file \(F^*\)).

Original adversary: must query entire files at once.

Stronger adversary: adaptively queries one message at a time from any file.

Our new schemes are secure against the stronger adversary.
Contribution 2: Stronger Adversary Security model for homomorphic signatures:

\[ \text{Chall.} \quad \xrightarrow{pk} \quad \text{Adversary} \]

File \( F = \{m_1, \ldots, m_k\} \)

Forgery is a valid signature \( \sigma^* \) on \((F^*, m^*, f)\) with \( m^* \neq f \) (messages in file \( F^* \)).

Original adversary: must query entire files at once. Stronger adversary: adaptively queries one message at a time from any file. Our new schemes are secure against the stronger adversary.
Contribution 2: Stronger Adversary

Security model for homomorphic signatures:

Chall. \[\text{pk} \rightarrow \text{file } F = \{m_1, \ldots, m_k\} \leftarrow \text{sigs } \sigma_1^F, \ldots, \sigma_k^F \rightarrow\]

Adversary

 Forgery is a valid signature \(\sigma^*\) on \((F^*, m^*, f)\) with \(m^* \neq f\) (messages in file \(F^*\)).

Original adversary: must query entire files at once.

Stronger adversary: adaptively queries one message at a time from any file.

Our new schemes are secure against the stronger adversary.
Contrubution 2: Stronger Adversary Security model for homomorphic signatures:

Chall. \[\text{pk} \rightarrow \text{file } F = \{m_1, \ldots, m_k\} \leftarrow \text{sigs } \sigma_1^F, \ldots, \sigma_k^F \rightarrow \text{Adversary} \]

Adversary

Forgery is a valid signature \(\sigma^*\) on \((F^*, m^*, f)\) with \(m^* \neq f\) (messages in file \(F^*\)).

Original adversary: must query entire files at once. Stronger adversary: adaptively queries one message at a time from any file.

Our new schemes are secure against the stronger adversary.
Contributed: Stronger Adversary

Security model for homomorphic signatures:

\[
\text{Chall.} \quad \text{sk} \quad \xrightarrow{pk} \quad \text{Adversary}
\]

\[
\text{file } F = \{m_1, \ldots, m_k\} \\
\text{sigs } \sigma_1^F, \ldots, \sigma_k^F \\
\text{forgery } F^*, m^*, \sigma^*, f
\]

Original adversary: must query entire files at once.
Stronger adversary: adaptively queries one message at a time from any file.

Our new schemes are secure against the stronger adversary.
Contribution 2: Stronger Adversary

Security model for homomorphic signatures:

\[
\text{Chall.} \quad \text{pk} \quad \text{Adversary}
\]

\[
\text{file } F = \{m_1, \ldots, m_k\} \quad \text{sig} \sigma_1^F, \ldots, \sigma_k^F
\]

\[
\text{forgery } F^*, m^*, \sigma^*, f
\]

Forgery is a valid signature \( \sigma^* \) on \((F^*, m^*, f)\) with

\[
m^* \neq f(\text{messages in file } F^*).
\]
Contributions 2: Stronger Adversary

Security model for homomorphic signatures:

\[
\text{Chall.} \quad \text{pk} \quad \text{sk} \quad \text{Adversary}
\]

file \( F = \{m_1, \ldots, m_k\} \rightarrow \text{file} \rightarrow \text{sigs} \rightarrow \text{adversary} \rightarrow \text{forgery} \)

 Forgery is a valid signature \( \sigma^* \) on \( (F^*, m^*, f) \) with

\[ m^* \neq f(\text{messages in file } F^*) \]

- Original adversary: must query entire files at once.
 CONTRIBUATION 2: STRONGER ADVERSARY

Security model for homomorphic signatures:

Chall.  \( \text{pk} \)  \( \text{sk} \)  
\[
\text{file } F = \{ m_1, \ldots, m_k \} \leftarrow \text{sigs} \sigma_1^F, \ldots, \sigma_k^F \rightarrow \text{forgery } F^*, m^*, \sigma^*, f
\]

Forgery is a valid signature \( \sigma^* \) on \((F^*, m^*, f)\) with

\[ m^* \neq f(\text{messages in file } F^*). \]

- Original adversary: must query entire files at once.
- Stronger adversary: adaptively queries one message at a time from any file.

David Mandell Freeman  Data Integrity in the Cloud
Contrubution 2: Stronger Adversary

Security model for homomorphic signatures:

Chall. $\rightarrow$ tuple $(F, i, m_i)$ $\leftarrow$ sig $\sigma_i^F$

Adversary $\rightarrow$ forgery $F^*, m^*, \sigma^*, f$

Forgery is a valid signature $\sigma^*$ on $(F^*, m^*, f)$ with

$m^* \neq f($messages in file $F^*$).

- Original adversary: must query entire files at once.
- Stronger adversary: adaptively queries one message at a time from any file.
**Contribution 2: Stronger Adversary Model for Homomorphic Signatures**

Security model for homomorphic signatures:

Chall. $\rightarrow$ \[\text{pk}\]

\[\text{tuple } (F, i, m_i) \leftarrow \text{sig } \sigma^F_i\]

Adversary $\rightarrow$

\[\text{forgery } F^*, m^*, \sigma^*, f \leftarrow\]

\[\text{repeat}\]

 Forgery is a valid signature $\sigma^*$ on $(F^*, m^*, f)$ with

$$m^* \neq f(\text{messages in file } F^*).$$

- **Original adversary**: must query entire files at once.
- **Stronger adversary**: adaptively queries *one message at a time* from any file.

Our new schemes are secure against the stronger adversary.
“Homomorphic hash”: for fixed $h_1, \ldots, h_n \in \mathbb{Z}_N$, vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n} \pmod{N}.$$ 

- Homomorphic property: $H_{\text{hom}}(\mathbf{v}) \cdot H_{\text{hom}}(\mathbf{w}) = H_{\text{hom}}(\mathbf{v} + \mathbf{w})$. 

[1] Construction: Sign$(\mathbf{v}) = H_{\text{hom}}(\mathbf{v})^{1/e} \pmod{N}$. 

$h_i$ derived from filename $F$ using “ideal” hash function. 

Sign$(\mathbf{v}) \cdot $Sign$(\mathbf{w})$ authenticates $\mathbf{v} + \mathbf{w}$. 

New idea: Tie together signature on $F$ and hash of $\mathbf{v}$: for public $x$ and “real-world” hash function $\eta$, 

Sign$(\mathbf{v}) = (x^{\eta(F)}, H_{\text{hom}}(\mathbf{v})^{1/\eta(F)})$. 

This example derived from [GHR99] signatures; mechanism also applies to [BB04], [W05], [HW09], [LW10],...
Ideas behind the Construction

“Homomorphic hash”: for fixed $h_1, \ldots, h_n \in \mathbb{Z}_N$, vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n} \pmod{N}.$$  

- Homomorphic property: $H_{\text{hom}}(\mathbf{v}) \cdot H_{\text{hom}}(\mathbf{w}) = H_{\text{hom}}(\mathbf{v} + \mathbf{w}).$

**[GKKR10] construction:** $\text{Sign}(\mathbf{v}) = H_{\text{hom}}(\mathbf{v})^{1/e} \pmod{N}.$

- $h_i$ derived from filename $F$ using “ideal” hash function.
- $\text{Sign}(\mathbf{v}) \cdot \text{Sign}(\mathbf{w})$ authenticates $\mathbf{v} + \mathbf{w}$. 
Ideas behind the Construction

“Homomorphic hash”: for fixed $h_1, \ldots, h_n \in \mathbb{Z}_N$, vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n} \pmod{N}.$$ 

- Homomorphic property: $H_{\text{hom}}(\mathbf{v}) \cdot H_{\text{hom}}(\mathbf{w}) = H_{\text{hom}}(\mathbf{v} + \mathbf{w})$.

[GKKR10] construction: $\text{Sign}(\mathbf{v}) = H_{\text{hom}}(\mathbf{v})^{1/e} \pmod{N}$.

- $h_i$ derived from filename $F$ using “ideal” hash function.
- $\text{Sign}(\mathbf{v}) \cdot \text{Sign}(\mathbf{w})$ authenticates $\mathbf{v} + \mathbf{w}$.

New idea: Tie together signature on $F$ and hash of $\mathbf{v}$: for public $x$ and “real-world” hash function $\eta$,

$$\text{Sign}(\mathbf{v}) = \left( x^{1/\eta(F)}, H_{\text{hom}}(\mathbf{v})^{1/\eta(F)} \right).$$
“Homomorphic hash”: for fixed \( h_1, \ldots, h_n \in \mathbb{Z}_N \), vector \( \mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n \), define

\[ H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n} \pmod{N}. \]

- Homomorphic property: \( H_{\text{hom}}(\mathbf{v}) \cdot H_{\text{hom}}(\mathbf{w}) = H_{\text{hom}}(\mathbf{v} + \mathbf{w}) \).

**[GKKR10] construction**: \( \text{Sign}(\mathbf{v}) = H_{\text{hom}}(\mathbf{v})^{1/e} \pmod{N} \).

- \( h_i \) derived from filename \( F \) using “ideal” hash function.
- \( \text{Sign}(\mathbf{v}) \cdot \text{Sign}(\mathbf{w}) \) authenticates \( \mathbf{v} + \mathbf{w} \).

**New idea**: Tie together signature on \( F \) and hash of \( \mathbf{v} \):

for public \( x \) and “real-world” hash function \( \eta \),

\[ \text{Sign}(\mathbf{v}) = \left( x^{1/\eta(F)}, H_{\text{hom}}(\mathbf{v})^{1/\eta(F)} \right). \]

This example derived from [GHR99] signatures; mechanism also applies to [BB04],[W05],[HW09],[LW10],....
Thank you!

Questions?

Comments?

Job Offers?