

EVE: An Implementation of an Efficient Homomorphic Encryption Scheme on Integer Vectors

Angel Yu
Wai Lok Lai
James Payor

May 6, 2015

Introduction to Homomorphic Encryption

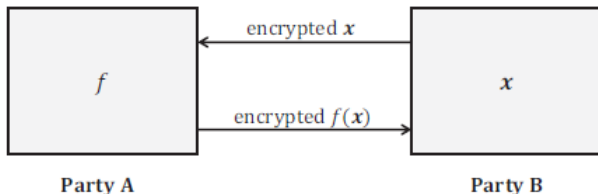


Figure: Most common usage of homomorphic encryption schemes

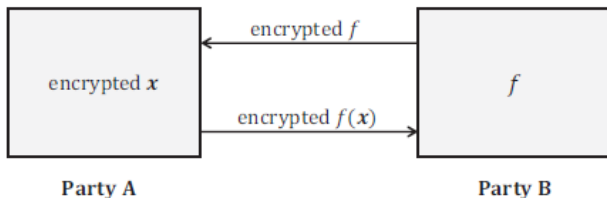


Figure: EVE, the homomorphic encryption scheme considered in our project

Overview of EVE: Supported Operations

Fundamental Operations:

- Encrypt: $\mathbf{c} = E_S(\mathbf{x})$
- Decrypt: $\mathbf{x} = D_S(\mathbf{c})$
- Switching secret keys from S to S'

Supported Operations on Integer Vectors:

- Addition of two vectors: $\mathbf{x}_1 + \mathbf{x}_2$
- Linear transformation: $G\mathbf{x}$
- Weighted inner product of two vectors: $\mathbf{x}_1^T H \mathbf{x}_2$

Can compose these operations to support arbitrary integer polynomials

Details of Fundamental Operations

Encrypt \mathbf{x} with secret key S :

- Choose \mathbf{c} such that $S\mathbf{c} = w\mathbf{x} + \mathbf{e}$

Decrypt \mathbf{c} with S :

- $\mathbf{x} = \left\lceil \frac{S\mathbf{c}}{w} \right\rceil$

Key switching from S to $S' = [I, T]$:

- Want $S'\mathbf{c}' = S\mathbf{c}$
- Want key-switch matrix M such that $S'M = S + E$
- $M = \begin{pmatrix} -TA + S + E \\ A \end{pmatrix}$ for random matrix A , random noise matrix E
- Then $\mathbf{c}' = M\mathbf{c}$

Details of the Three Supported Operations

Addition: $\mathbf{x}' = \mathbf{x}_1 + \mathbf{x}_2$:

- Let $\mathbf{c}' = \mathbf{c}_1 + \mathbf{c}_2$
- So $S\mathbf{c}' = w(\mathbf{x}_1 + \mathbf{x}_2) + (\mathbf{e}_1 + \mathbf{e}_2)$

Linear Transformation: $\mathbf{x}' = G\mathbf{x}$

- Note $GS\mathbf{c} = wG\mathbf{x} + G\mathbf{e}$
- Hence $E_S(\mathbf{x}) = E_{GS}(G\mathbf{x})$
- So $\mathbf{c}' = \mathbf{c}$ and decrypt with $S' = GS$

Details of the Three Supported Operations (con'd)

Weighted Inner Product: $\mathbf{x}' = \mathbf{x}_1^T H \mathbf{x}_2$

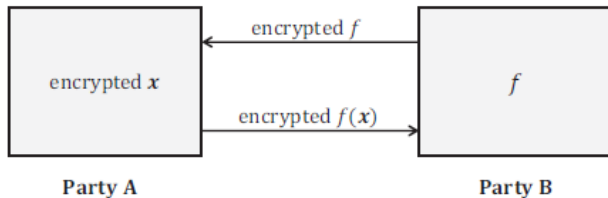
- Let $S' = \text{vec}(S^T H S)^T$
- Let $\mathbf{c}' = \left\lceil \frac{\text{vec}(\mathbf{c}_1 \mathbf{c}_2^T)}{w} \right\rceil$
- Note $(S \mathbf{c}_1)^T H (S \mathbf{c}_2) = w^2 (\mathbf{x}_1^T H \mathbf{x}_2) + (w \mathbf{x}_1^T H \mathbf{e}_2 + w \mathbf{e}_1^T H \mathbf{x}_2 + \mathbf{e}_1^T H \mathbf{e}_2)$
- So $S' \mathbf{c}' = w (\mathbf{x}_1^T H \mathbf{x}_2) + \mathbf{e}'$

Secrecy in the Scheme

- Clients (Party B) calculate key-switch matrices based on operation f
- Server (Party A) performs the operation without knowing f
- Relies on key-switch matrices being indistinguishable from random

Learning with Error Problem (LWE)

Given S and M , solving $S'M = S + E$ to find S' is hard



Applications

- In each application, we have:
 - ▶ Server with data (can be encrypted with S , known to the client)
 - ▶ Client that wants to learn a function of the data
- Effective when server has a lot of data, and results are small

Applications

- Search

- ▶ Server has (encrypted) feature vectors for our data
- ▶ Client wants to score each item to rank them

- Classification

- ▶ Can run any polynomial classifier on the server's data (e.g. naive bayes, SVMs with polynomial kernels)

- Feature extraction

- ▶ We can generalize classification to give a low-dimensional representation of data vectors, which will conserve bandwidth over simply querying all the files.

- We implemented the scheme, and two applications:
 - ▶ Private search on encrypted data
(using TF-IDF relevance on common words)
 - ▶ Spam classification (using a naive-bayes model)
- Server has encrypted word counts of 3500 common words for 200 Enron emails
- Server can't learn our queries, or even distinguish between each type

Live Demo!



Conclusions

- In our demo, scheme is slow due mainly to our lack of optimization.
- No overhead in addition
- Multiplicative overhead in linear transformations and inner products equal to the number of bits involved.
 - ▶ (Note as given, inner products is slow, but we can combine this with a linear transformation step to reduce the work.)
- This scheme compares well to fully homomorphic encryption by limiting scope of computation.
 - ▶ Benchmarks give that HElib achieves 500 multiplications per second for small integers, orders of magnitude worse than our slowdown.

Questions?



Speak now, or forever hold your peace.

Overview of EVE (Any questions?)

Operations:

- Switching secret keys from S to S' : $S'M = S + E$
- Addition of two vectors: $S(\mathbf{c}_1 + \mathbf{c}_2) \approx w(\mathbf{x}_1 + \mathbf{x}_2)$
- Linear transformation: $GS\mathbf{c} = wG\mathbf{x}$, $GS \rightarrow S'$
- Weighted inner product of two vectors: $\mathbf{c}_1^T S^T H S \mathbf{c}_2 \approx w^2 \mathbf{x}_1^T H \mathbf{x}_2$

Applications:

- Anonymous search
- Classification with hidden models
- Computing hidden polynomials of encrypted data