HEAT Project Blog

The blog for the EU H2020 funded project HEA T on Homomorphic Encryption

Wednesday, July 1, 2015

Yet Another Somewhat Homomorphic Encryption (YASHE) Scheme

In this blog post we outline the main operations in the YASHE fully homomorphic encryption (FHE) scheme. For the exact details see the paper introducing YASHE.

López-Alt, Tromer, and Vaikuntanathan proposed a (multi-key) FHE scheme based on the work by Stehlé and Steinfeld in which a provably secure version of NTRUEncrypt is presented with security based on standard problems in ideal lattices. The FHE scheme from López-Alt, Tromer, and Vaikuntanathan needs to make an additional assumption relating to the uniformity of the public key the so-called decisional small polynomial ratio (DSPR) assumption, to allow homomorphic operations and remain semantically secure. Brakerski introduced an approach to limit the noise growth during homomorphic operations via a tensoring technique.

In this paper, Yet Another Somewhat Homomorphic Encryption (YASHE) scheme is introduced which incorporates the best of these techniques. YASHE avoids the DSPR assumption by using the techniques described by Brakerski and construct a new fully homomorphic encryption scheme from the Stehlé and Steinfeld version based on standard lattice assumptions and a circular security assumption.

Besides this theoretical advantage, YASHE has other attractive properties

1. This new scheme is scale-invariant: this means it avoids the modulus-switching technique of Brakerski, Gentry and Vaikuntanathan.
2. The ciphertext consists of only a single ring element (as in this paper) as opposed to the two or more ring elements for schemes based purely on the ring learning with errors (RLWE) assumption.

In the following I will describe the more practical variant of YASHE (denoted YASHE' in the paper). Note that this practical variant YASHE' does need the DSPR assumption.

In order to show how YASHE works we need to fix some parameters. Selecting secure parameters is a difficult task by itself for which tools have been generated (see for instance our previous blog post). For this post I simply assume correct and secure parameters have been chosen (but see the paper for some example parameters). Given the security parameter λ, fix a positive integer d and modulus q that determine $R = \mathbb{Z}[X]/(\Phi_d(X))$, and t with $1 < t < q$, and distributions χ_{key}, χ_{err} on R.

The message space is $R/tR = (\mathbb{Z}[X]/(\Phi_d(X)))/(t(\mathbb{Z}[X]/(\Phi_d(X))))$.

The function $P_{w,q}$ is a generalization of the PowersofTwo and $D_{w,q}$ is a generalization of BitDecomp from this paper. Instead of a radix-2 representation these functions use a radix-w system. These function take a single ring element and output $\ell_{w,q} = \lfloor \log_w(q) \rfloor + 2$ ring elements. The choice of w is important since it allows for a trade-off between efficiency and error growth.

The Heat Project

Labels

lattice-based cryptography publication ADOC Introduction Smart Grid homomorphic encryption job lattice-based crypto optimization post-quantum cryptography standardization

G+1 0

Contributors

- Adrian Waller
- Anamaria Costache
- Antoine Joux
- CryptoExperts
- Fre Vercauteren
- Ilia Iliashenko
- JS
- Jean Claude Bajard
- Joppe Bos
- Marc Vauclair
- Nigel Smart
- Pascal Paillier
- Srinivas Vivek
- Vincent Zucca
- Wouter Castryck

Blog Archive

- ► 2017 (2)
- ► 2016 (11)
- ▼ 2015 (22)

Key generation. KeyGen $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$

1. Sample $f', g \leftarrow \chi_{\text{key}}$ and let $f = [tf' + 1]_q$.
2. If f is not invertible modulo q, choose a new f'.
3. Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tgf^{-1}]_q$.
4. Sample $\vec{e}, \vec{s} \leftarrow \chi_{\text{err}}^{\ell_{w,q}}$, compute $\vec{\gamma} = [P_{w,q}(f) + \vec{e} + h \cdot \vec{s}]_q \in R^{\ell_{w,q}}$.
5. Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) = (h, f, \vec{\gamma})$.

Encryption. Encrypt (m, pk)

1. For a message $m + tR$, choose $[m]_t$ as its representative.
2. Sample $s, e \leftarrow \chi_{\text{err}}$.
3. Output the ciphertext $c = [\lfloor q/t \rfloor [m]_t + e + \mathsf{pk} \cdot s]_q \in R$.

Decryption. Decrypt (c, sk)

1. To decrypt a ciphertext c, compute $m = \left[\left\lfloor \frac{t}{q} \cdot [\mathsf{sk} \cdot c]_q \right\rceil\right]_t \in R$.

Key switching. KeySwitch $(\tilde{c}_{\mathsf{Mult}}, \mathsf{evk})$

1. Output the ciphertext $[\langle D_{w,q}(\tilde{c}_{\mathsf{Mult}}), \mathsf{evk} \rangle]_q$.

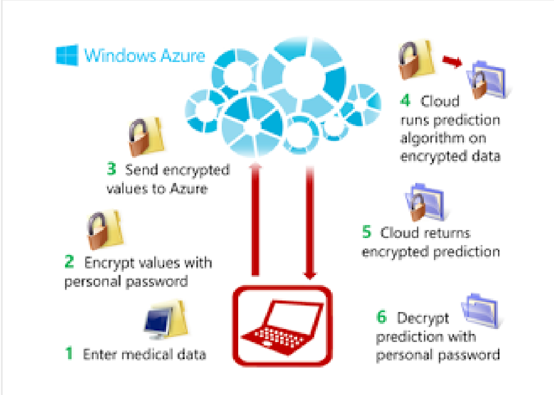
The key switching algorithm transforms an intermediate encryption into a ciphertext that can be decrypted with f itself. The evaluation key is $\mathsf{evk} = [P_{w,q}(f) + \vec{e} + h \cdot \vec{s}]_q$, where $\vec{e}, \vec{s} \leftarrow \chi_{\text{err}}^{\ell_{w,q}}$ are vectors of polynomials sampled from the error distribution χ_{err}. This key is a vector of quasi-encryptions of the secret key f under its corresponding public key. It is required for the homomorphic multiplication operation and is therefore made public. This means, we need to make a circular security assumption, namely that the scheme is still secure even given that evk is publicly known.

Homomorphic addition. Add (c_1, c_2)
Given two ciphertexts $c_1, c_2 \in R$, which encrypt two messages m_1, m_2, their sum *modulo* q, $c_{\mathsf{Add}} = [c_1 + c_2]_q$, encrypts the sum of the messages *modulo* t, $[m_1 + m_2]_t$.

Homomorphic multiplication. Mult (c_1, c_2, evk)
Output the ciphertext

$$c_{\mathsf{Mult}} = \mathrm{KeySwitch}(\tilde{c}_{\mathsf{Mult}}, \mathsf{evk}), \text{ where } \tilde{c}_{\mathsf{Mult}} = \left[\left\lfloor \frac{t}{q} \cdot c_1 \cdot c_2 \right\rceil\right]_q.$$

YASHE in practice
This practical version of YASHE has been used in order to ensure privacy of sensitive medical data. In this work it is shown how to privately conduct predictive analysis tasks on encrypted data using homomorphic encryption. As a proof of concept a working implementation of a prediction service running in the cloud, which takes as input private encrypted health data, and returns the probability for suffering cardiovascular disease in encrypted form. Since the cloud service uses homomorphic encryption, it makes this prediction while handling only encrypted data, learning nothing about the submitted confidential medical data.



In a recent paper in the framework of the HEAT project, a modular hardware architecture for somewhat homomorphic function evaluation using YASHE is presented. In another recent publication, YASHE is implemented to investigate the potential of FPGAs.

Posted by Joppe Bos at 11:46 AM

G+1 +1 Recommend this on Google

No comments:

Post a Comment

Enter your comment...

Comment as: Skanda (Googl ▼ Sign out

Publish Preview Notify me

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)