[Computing on Encrypted Data (Part 1)]

Homomorphic Encryption from the Standard / Ring LWE Assumptions

Rohit Khera

"We know that corporeal entities such as human beings or physically secured media can maintain secrets. But can disembodied artificial agents, such as programs running on untrusted computers have, keep and use secrets? Apparently, the answer is yes."
- Amit Sahai, UCLA Academic and Cryptographer

The accountant, blindfolded

Consider an accountant that recieves a financial ledger, and is asked to prepare a simple balance statement. Except that she is blindfolded. Is she up to the task? Can she fulfill this assignment?

If we think of the accountant as a metaphor for a program running on an untrusted computer operating on encrypted data, results in theoretical computer science and cryptography tell us that the answer to this question is (resoundingly) - yes .

What We'll Cover

Rings and Homomorphisms

Homomorphic Encryption over the Integers

Standard LWE Encryption

Ring LWE Encryption (R-LWE)

Algebraic Structure of R-LWE Plaintexts

Demo - Homomorphic Eval of AES Function

Ring Homomorphisms

A map f between two rings P and C

$$f : P \to C$$

Such that for elements a and b in P

$$f(a+b) = f(a) + f(b)$$

$$f(a \times b) = f(a) \times f(b)$$

Ring Homomorphisms

By the homomorphism property, if $f : P \to C$ maps

$a \to a'$, $b \to b'$, $c \to c'$

and

$a + b = c$ in P, then $a' + b' = c'$ in C

so that $f^{-1}(c') = c$ (and similarly for multiplication)

Adding and multiplying elements in C implicitly adds and multiplies elements in P

Add and *Mult* are "Turing Complete" in the Ring

Ring Homomorphisms

P = The plaintext "space"

C = The ciphertext "space"

$f(\cdot)$ = The encryption function?!

Question:

Can we construct encryption functions as homomorphisms of rings?

If so, we can construct encrypting homomorphisms that preserve the algebraic structure of the plaintext, and thus evaluate functions (expressed as layered arithmetic circuits comprised of $+$ and $\times$ gates) on the ciphertext that will produce a result, upon decryption, corresponding to the result of the function evaluated directly on the plaintext

Security

We've defined homomorphisms ...

What do we require of encryption functions?

Malleability

HE encryption should be semantically secure

But ...

Is malleable by design

Homomorphic Encryption Over the Integers (DGHV 2011)

Encryption To encrypt a bit $b \in \{0,1\}$, choose $Q \leftarrow_R \mathbf{Dist}(\mathbb{Z}_N)$ and $E \leftarrow_R \mathbf{Dist}(\mathbb{Z}_n)$ with $n << N$. P, a "large" odd number is the secret key.

Then X, an excryption of the bit b is given by

$$X = QP + 2E + b.$$

Decryption To decrypt X, output $(X \pmod P) \pmod 2$.

Is It Homomorphic ?

Encrypt two messages b and b' to get two ciphertexts X and X'

$X = QP + 2E + b$ and $X' = Q'P + 2E' + b'$

Adding the two ciphertexts gives us:

$$X + X' = P(Q + Q') + 2(E + E') + (b + b')$$

Since $E<<P$ and $E'<<P$, decrypting the sum gives us $(((X + X') \bmod \mathrm{P}) \bmod 2) = b + b'$, the sum of the plaintexts

Multiplying the two ciphertexts gives us:

$$X \cdot X' = (QQ'P + 2E'Q + b'Q + 2EQ + bQ')P + 2(2EE' + Eb' + bE') + bb'$$

Since $E<<P$ and $E'<<P$, decrypting the product gives us $(((X \cdot X') \bmod \mathrm{P}) \bmod 2) = b \cdot b'$, the product of the plaintexts
Notice b can only be 0 or 1, so adding two bits is a boolean XOR operation and multiplying them is boolean AND

Encryptions of Zero

For a ciphertext:

$$X = QP + 2E + b,$$

the noise term $QP + 2E$

is called an "encryption of zero".

Encryptions of zero are public keys

(Actually, random subset sums of these are public keys)

Is It Secure ?

(ciphertext $X = 2E_i + Q_iP + \mathrm{b}$)

An easy problem:

Find the greatest common divisor (GCD) of two or more integers:

Given many Q_iP output P

Is It Secure ?

(ciphertext $X = 2E_i + Q_iP + b$)

A harder problem:

Find the "approximate" greatest common divisor (GCD) of two or more integers:

Given many $2E_i + Q_iP$ **output** P

$2E_i + Q_iP$ is the noise term added to our message b

Is It Secure ?

For $E_i \sim O(\theta)$

$Q_i \sim O(\theta^5)$ and

$P \sim O(\theta^2)$

where θ is a security parameter

best known attacks on approximate GCD require 2^θ time.

Noise

Consider the sum of two ciphertexts:

$$X + X' = P(Q + Q') + 2(E + E') + (b + b')$$

Since this decrypts as

$$(((X + X') \bmod \mathrm{P}) \bmod 2) = b + b'$$

We need to ensure that the noise term $2(E + E')$ remains small relative to P so that it does not reduce mod P. Otherwise, decryption will not produce the right result

Standard Learning with Errors (LWE) (BV 2011, BGV 2011)

Parameters: q such that $gcd(q, 2) = 1$

Secrey Key: $\mathbf{s}$ uniformly chosen from $\mathbb{Z}_q^n$

Public Key: Linear polynomials $\{f_i(x_1, ..., x_n\}$ such that $f_i(\mathbf{s}) = 2e_i$, where $e_i << q$

Encrypt: Select $g(x_1, ..., x_n)$ as a random subset sum of the $\{f_i(x_1, ..., x_n\}$. Output a ciphertext that encrypts a message m as $c(x_1, ..., x_n) = m + g(x_1, ..., x_n)$

Decrypt: Evaluate the ciphertext polynomial at the secret key $\mathbf{s}$: $m + g(\mathbf{s}) = m + 2e$. Then reduce mod 2

Is It Homomorphic ?

For $+$ and $\times$, we just add and multiply ciphertext polynomials of the form $c(x_1, ..., x_n) = m + g(x_1, ..., x_n)$.

The noise polynomials $g(x_1, ..., x_n)$ are encryptions of zero, anything they multiply is also an encryption of zero.

$$c_1 \times c_2 = (m_1 + g_1(x_1, ..., x_n)) \times (m_2 + g_2(x_1, ..., x_n))$$

Security of LWE Encryption

Roughly speaking, security is based on ability to efficiently distinguish between the following "distributions":

1: Polynomials that evaluate to something small and even at the secrey key

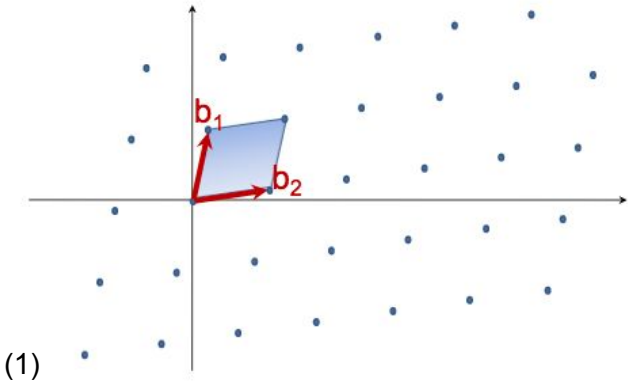
2: Polynomials that are uniformly sampled from the ring

Distinguishing between the above is equivalent to a very well studied problem in the area of lattices

Hard Problems in Lattices - Shortest Vector Problem

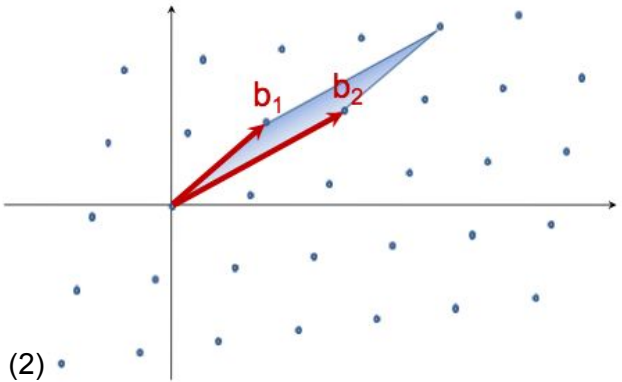
Basis of a lattice: Set of linearly independent vectors that generate the lattice

"Good" Basis



(1)

"Bad" Basis



(2)

Shortest Vector Problem: Given a bad basis for a high dimensional lattice, what is the shortest vector in the lattice?

(1)(2) Craig Gentry, Fearful Symmetry: Can We Solve Ideal Lattice Problems Efficiently, Workshop on Lattices with Symmetry, Aug 2013, UCI

Ring Learning with Errors (RLWE) (BGV 2011)

We now describe a scheme that is similar to Standard LWE

Encypting polynomials still evaluate to smeven in the secrey key, But:

They're drawn from Integer Subrings of "Cyclotomic Number Fields"

$\rightarrow$ Message space no longer a single bit, can be any element in the ring

Choose rings where elements are degree n binary polynomials

We can now encrypt bit strings!

Integer Chinese Remainder

For $n_1, ..., n_k$ pairwise coprime and for any $a_1, ..., a_k$, there is an x s.t.

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

Algebraically, this epxresses the ring $\mathbb{Z}_n$ as a Cartesian product of the the k subrings.
We will use a plynomial equivalent of this for Ring LWE

Ring Structure and Parallel Ops

(Maybe rework this slide)
Consider the ring:
$\mathbb{A} = \mathbb{Z}[X]/\langle \Phi_m(X) \rangle$ where m is a parameter and $\Phi_m(X)$ is the $m'th$ cyclotomic polynomial

The plaintext space is defined as $\mathbb{A}_2 = \mathbb{A}/2\mathbb{A}$
$\Phi_m(X)$ is chosen to factor into r degree d irreducible polynomials mod 2

$$\Phi_m(X) = F_1(X) \cdot F_2(X) \cdots F_r(X) \bmod 2$$

Chinese remaindering gives a structure theorem for the the ring $\mathbb{A}_2$

$$\mathbb{A}_2 \cong \mathbb{A}_2[X]/F_1(X) \otimes \mathbb{A}_2[X]/F_2(X) \otimes \cdots \otimes \mathbb{A}_2[X]/F_r(X)$$

$$\cong \mathbb{F}_{2^d} \otimes \cdots \otimes \mathbb{F}_{2^d}$$

So that $\mathbb{A}_2$ is isomorphic to r copies of $\mathbb{F}_{2^d}$ (fields of d bit strings).

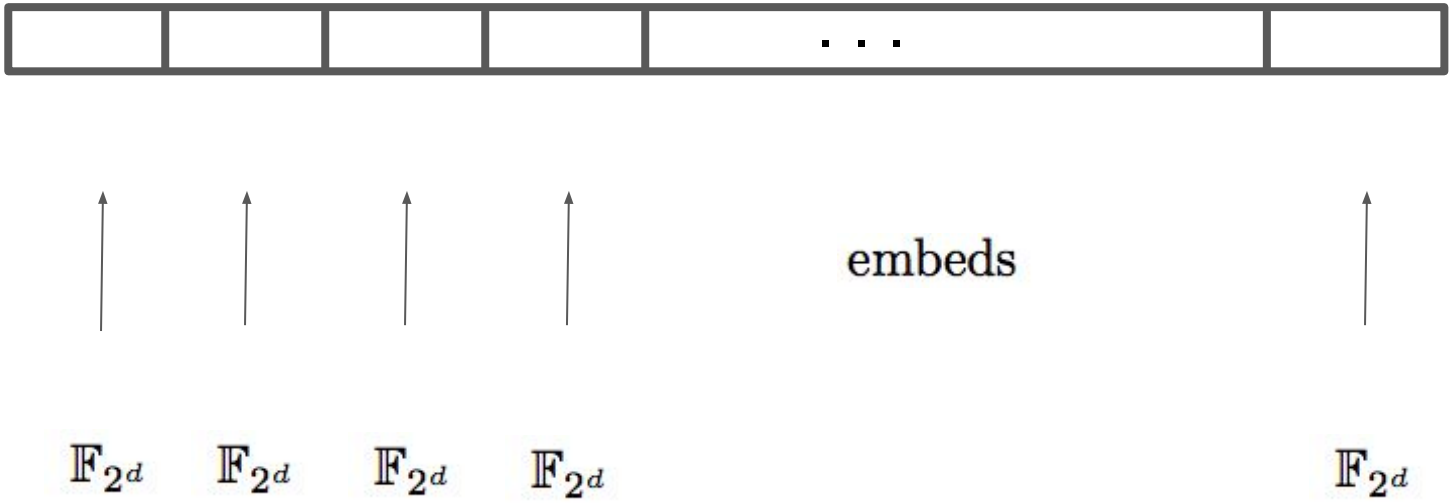
The subrings of $\mathbb{A}_2$ are therefore d bit binary strings!

So we can encode distinct strings in the $\mathbb{F}_{2^d}$
We can now do SIMD $+$ and $\times$ in $\mathbb{A}_2$ - implicitly and in parallel, this $+$ and $\times$ in the $\mathbb{F}_{2^d}$ rings

Ring Structure and Parallel Ops

$$\mathbb{A} = \mathbb{Z}[X]/\langle \Phi_m(X) \rangle$$
$$\mathbb{A}_2 = \mathbb{A}/2\mathbb{A}$$

$\mathbb{A}_2$, the "aggregate" plaintext space



Representing Bytes

We have seen that the aggregate plaintext space $\mathbb{A}_2$ is isomorphic to r copies of $\mathbb{F}_{2^d}$ (fields of d bit strings).

So how can we represent bytes?

We do so by embedding F_{2^8} into the aggregate ring inorder to represent arrays of bytes.
WE can do this by selecting $\Phi_m(x)$ such that it splits into factors of degree d in the extension field such that 8 divides d

Permuting Data: Field Automorphisms

$$\mathbb{A} = \mathbb{Z}[X]/\langle \Phi_m(X) \rangle$$
$$\mathbb{A}_2 = \mathbb{A}/2\mathbb{A}$$

$\mathbb{F}_{2^d}$ $\mathbb{F}_{2^d}$ $\mathbb{F}_{2^d}$ $\mathbb{F}_{2^d}$ $\mathbb{F}_{2^d}$ $\mathbb{F}_{2^d}$

Bootstrapping

Bootstrapping is a way to manage accumulated noise during ciphertext evaluation.
The main concept here is that decrpytion reduces the noise to zero!
(Probably skip this whole slide or greatly simplify)

The ciphertext Ψ_1 encrypts a message π under a public key pk_1, and a secret key sk_1 associated with pk_1. The key sk_1 is then encrypted under a second public key pk_2. Denote the encryption of the jth bit of sk_1 under pk_2 as $\bar{sk}_{1j}$. Define a 're-encryption' procedure for the ciphertext Ψ_1 as the following -
$Recrypt(pk_2, D_\varepsilon, \bar{sk}_1, \Psi)$:

1) $\bar{\Psi}_1 \leftarrow Encrypt(pk_2, \Psi_1 j)$

2) $\Psi_2 \leftarrow Evaluate(pk_2, \mathbf{Dec}, \bar{sk}_1, \bar{\Psi}_1)$

Where **Dec** denotes the HE decryption. The algorithm takes the ciphered message Ψ_1 (which is an encryption of the message π under pk_1), and encrypts it under the "outer" encryption key pk_2 to give the ciphertext $\bar{\Psi}_1$. The decryption circuit is then homomorphically evaluated with the ciphered decryption secret $\bar{sk}_1$ (the secret key sk_1 encrypted under pk_2) and the "doubly" encrypted message ciphertext $\bar{\Psi}_1$. This latter step removes the 'inner' encryption of the message π under the key pk_1, thereby yielding the ciphertext Ψ_2 which is just the encryption of π under pk_2

Circuits vs. RAM Machines

Homomorphic Evaluation of AES Encryption / Decryption

AES Overview

AES-128 consists of 10 applications of a keyed "round" function on 16 bytes of data

The round function operates on the data as a 4×4 matrix of bytes called the "state"

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

AES Round Function

The round function is comprised of the following operations:

AddKey

SubBytes

ShiftRows

MixColumns

AES Round Function

AddKey: XOR the round key with the data

SubBytes:

1) Represent each byte in the data as a degree 7 binary polynomial

2) Inverse the polynomial

3) Replace each byte with it's inverse

4) Multiply the replaced byte, now represented as an 8 bit vector, by a simple 4×4 matrix of zeros and ones

5) Put the result back in the state

AES Round Function

ShiftRows :

Rotate entries in the ith row of 4×4 state matrix

$i - 1$ places to the left

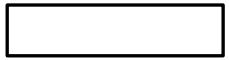
MixCols :

Multiply each column in the state by a fixed matrix

(multiplication is in the "Rijndael Galois Field")

What Does it Mean to Homomorphically Evaluate the AES Function?

Homomorphic AES Encryption



= unencrypted data

= HE encrypted data

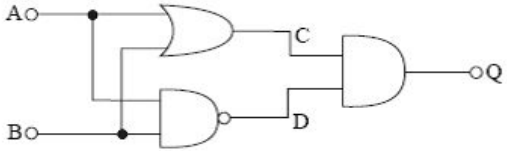
SK_AES = AES Secret Key

= AES Encrypted Data

= Doubly Encrypted Data

= HE Encrypted AES Secret Key

Homomorphic AES Encryption



HE “Circuit” Representation of AES Encryption Function

\+ SK_AES + DATA

= DATA

= HE encrypted data

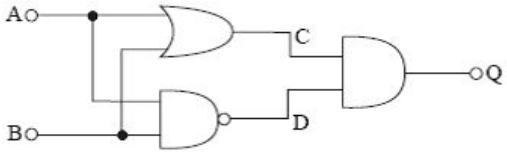
= AES Encrypted Data

= Doubly Encrypted Data

SK_AES = AES Secret Key

SK_AES = HE Encrypted AES Secret Key

Homomorphic AES Decryption



HE “Circuit” representation of AES Decryption Function

\+ SK_AES + DATA

= DATA

= HE encrypted data

= AES Encrypted Data

= Doubly Encrypted Data

SK_AES = AES Secret Key

SK_AES = HE Encrypted AES Secret Key

Example Cost - Homomorphic Eval of SubBytes Operation

3 Mult and 4 automorphisms for inverting a byte

7 more automorphisms for the affine matrix transform

Homomorphic AES -

2012 Performance (GHS 2012)

256 GB RAM

54 AES Blocks (864 bytes)

36 hours for the entire 10 AES rounds

Amortized Throughput: About 40 minutes to encrypt 16 bytes

Described by Gentry, Halevi, Smart

Homomorphic Evaluation of the AES Circuit

Crypto 2012 (Update 2015)

Demo

What is the performance 4 years later?

Demo

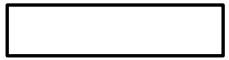
Utilize so called “Gentry - Halevi - Smart Optimizations” (GHS 2011)

GNU Multiprecision lib

Shoup’s Number Theory lib (NTL)

Haveli and Shoup’s Homomorphic Encryption lib (HElib)

Demo



= unencrypted data

= HE encrypted data

= AES Encrypted Data

= Doubly Encrypted Data

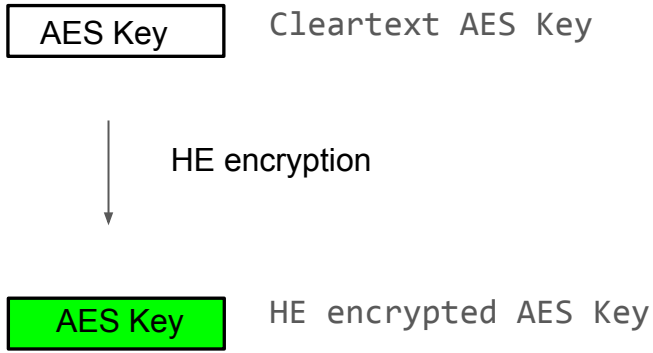
SK_AES = AES Secret Key

= HE Encrypted AES Secret Key

Demo

STEP 1: Homomorphic AES Encryption -

HE Encrypt the AES Key

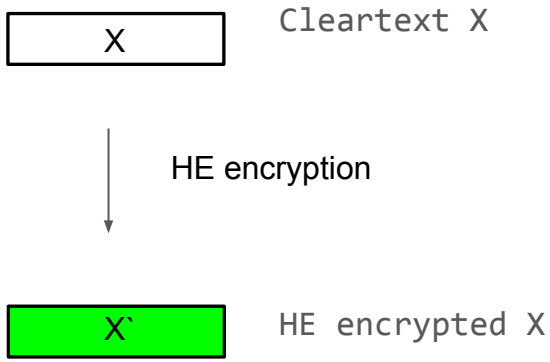



```
//Randomly Generate AES Key

BytesFromGF2X(aesKey.data(), rnd, aesKey.length());

// Encrypt the AES key under the HE key

hAES.encryptAESkey(encryptedAESkey, aesKey, publicKey)
```

Demo

STEP 1: Homomorphic AES Encryption -

HE Encrypt The Plaintext X

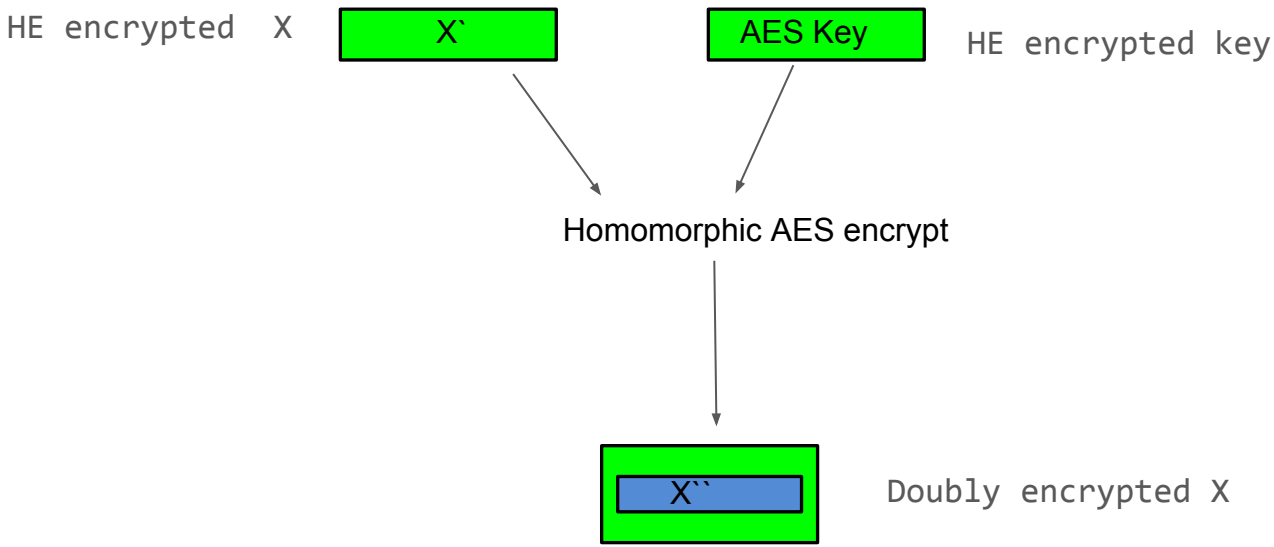



```
// Generate random bytes

BytesFromGF2X(ptxt.data(), rnd, ptxtLen);

//He encrypt plaintext, i.e. outout X`=Enc_HE(X)

hAES.homAESenc1(encrypted, ptxt);
```

Demo

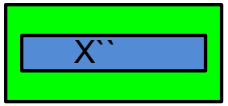
STEP 1: Homomorphic AES Encryption -

Doubly encrypt X



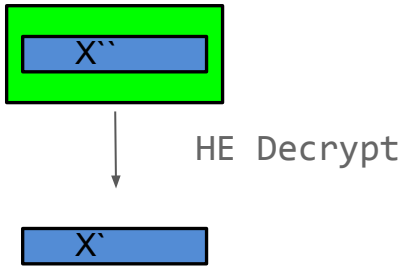

```
hAES.homAESenc2(doublyEncrypted, encrypted, encryptedAESKey);
```

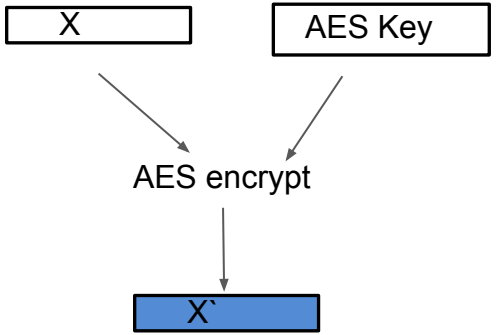
Demo

We now have: 

Doubly encrypted X

Let's do a sanity test:

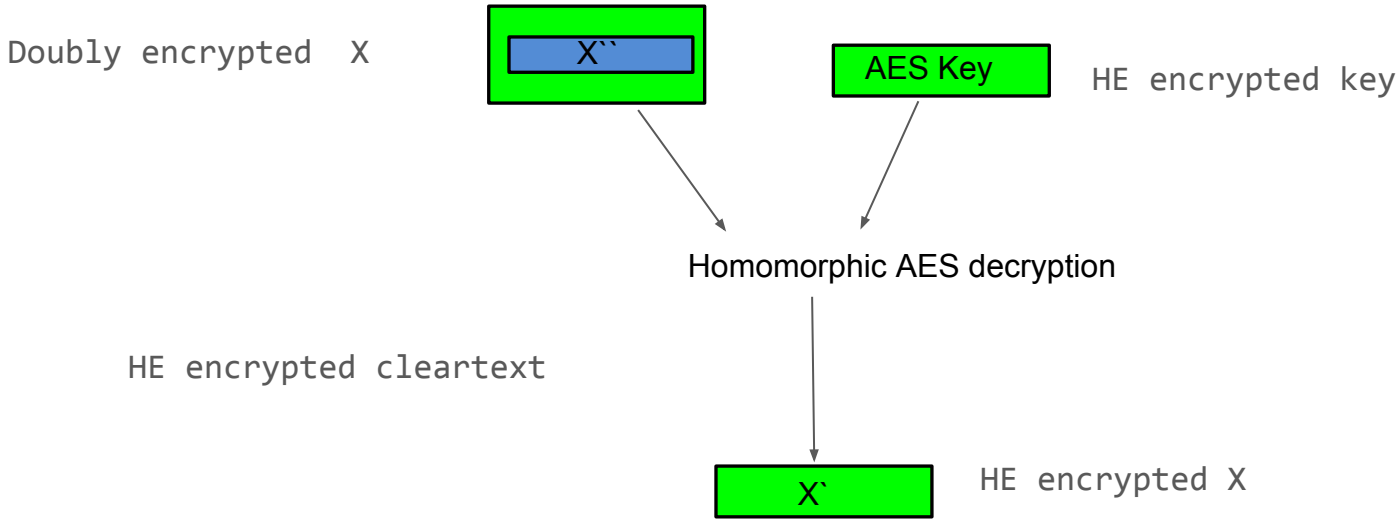
1. 

2. 

Check that the outcome from (1) and (2) are the same

Demo

Homomorphic AES Decryption:

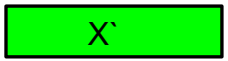


Now Homorphically AES Decrypt (remove “inner” AES encryption):

```
hAES.homAESdec(doublyEncrypted, encryptedAESkey, aesCtxt);
```

Demo

Where do we Stand?:



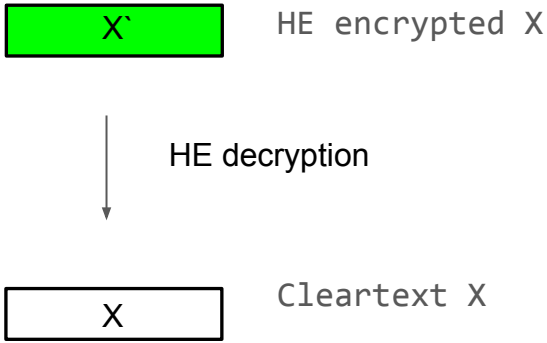
HE encrypted X

We have the plaintext X encrypted under the HE scheme

In theory, we can now do many things with this, like Homomorphic Virus Checking on Encrypted Data!

Demo

Homomorphic Decryption:



Questions