Danger of using fully homomorphic encryption: A look at Microsoft SEAL

Zhiniang Peng of 360 Core Security
@Overdrive 2019

360 INTERNET SECURITY CENTER

Who am I

Zhiniang Peng

Ph.D. in cryptography

Security researcher @Qihoo 360

Twitter: @edwardzpeng

Research areas:

- Software security
- Applied cryptography
- Threat hunting

About the topic

Introduction to homomorphic encryption

Introduction to SEAL

Security pitfalls of SEAL

- CCA attack on BFV
- Data recovery against FPSI
- Circuit privacy of SEAL
- Information leakage
- Countermeasures

Other issues

Conclusion

FORTUNE

L.A. Sues IBM's Weather Company over 'Deceptive' Weather Channel App

By DAVID MEYER January 4, 2019

The Weather Channel's app secretly sucks up users' personal data and uses it for things like targeted marketing and hedge fund analysis, the Los Angeles city attorney has claimed in a lawsuit against The Weather Company, the IBM-owned firm that runs th

The case was first repo
Mike Feuer will hold a
was taking "action aga
allege is egregious beh

LA City Attorney
@CityAttorneyLA
FRIDAY @11am - City Attorney Mik
as we take action against one of Am
for what we allege is egregious beh
#technology

The New York Times

Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret

Dozens of companies use smartphone locations to help advertisers and even hedge funds. They say it's anonymous, but the data shows how personal it is.

By JENNIFER VALENTINO-DeVRIES, NATASHA SINGER, MICHAEL H. KELLER and AARON KROLIK DEC. 10, 2018

Data reviewed by The Times shows over 235 million locations captured from more than 1.2 million unique devices during a three-day period in 2017. Image by U.S.D.A. N.A.I.P.

The millions of dots on the map trace highways, side streets and bike trails — each one following the path of an anonymous cellphone user.

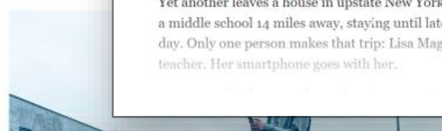
One path tracks someone from a home outside Newark to a nearby Planned Parenthood, remaining there for more than an hour. Another represents a person who travels with the mayor of New York during the day and returns to Long Island at night.

Yet another leaves a house in upstate New York at 7 a.m. and travels to a middle school 14 miles away, staying until late afternoon each school day. Only one person makes that trip: Lisa Magrin, a 46-year-old math teacher. Her smartphone goes with her.

Bloomberg

LA County Sues IBM's Weather Channel for User Location Tracking

By Gerrit De Vynck
January 4, 2019, 11:30 AM PST
Updated on January 4, 2019, 12:12 PM PST

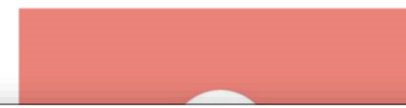


The city of Los Angeles is suing International Business Machines Corp.'s Weather Channel unit, accusing the company of misleading consumers about how their location data was being used.

In a complaint filed Thursday in California state court, the city alleges IBM used detailed location data from users for targeted advertising and to identify consumer trends that might be useful to hedge funds, while at the

WIRED

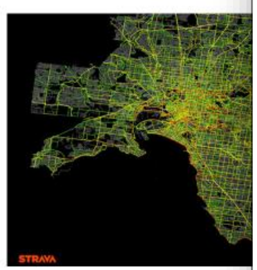
How to stop Google from tracking you and delete your personal data



engadget

Strava begins selling your data points, and no, you can't opt-out [Updated]

Mike Wehner, @MikeWehner
05.23.14



Strava (free) is an extremely popular running and biking app on iPhone, and has long been at or near the top of the fitness app charts. Along with its apps on other platforms including Android and even personal GPS devices, the company has pooled a whole lot of data about your running and cycling habits, which it recently used to create a stunning map of exercise routes around the world. Now the company is using that data as a product of its own, called Strava Metro.

Strava calls Strava Metro a "data service," and it's pitching its massive wealth of user patterns to transportation agencies, city governments, and corporations in a subscription-based format. The data itself, Strava notes, is scrubbed of all identifiable user information, meaning that a company or

news.com.au

How Google is secretly recording you through your mobile, monitoring millions of conversations

The Sun

News Corp Australia Network AUGUST 23, 2017 7:50AM

If you run Android software on your smartphone, Google may have been recording you every day without you knowing. Picture: Getty. Source: Supplied

DID you know that Google has been recording you without your knowledge?

The technology giant has effectively turned millions of its users' smartphones into listening devices that can capture intimate conversations — even when they aren't in the room.

If you own an Android phone, it's likely that you've used Google's Assistant, which is similar to Apple's Siri.

Google says it only turns on and begins recording when you utter the words "OK Google".

But a *Sun* investigation has found that the virtual assistant is a little hard of hearing, reports *The Sun*.

In some cases, just saying "OK" in conversation prompted it to switch on your phone and record around 20 seconds of audio.

Computing on encrypted data

Data leakage become more serious nowadays

- Data security, privacy become a public concern

It will be nice to be able to....

- Encrypt my data before sending to cloud
- Allow the cloud to search/sort/edit the data on my behalf
- Keep the data in cloud in encrypted form
- Without needing to ship it back and forth to be decrypted

Computing on encrypted data

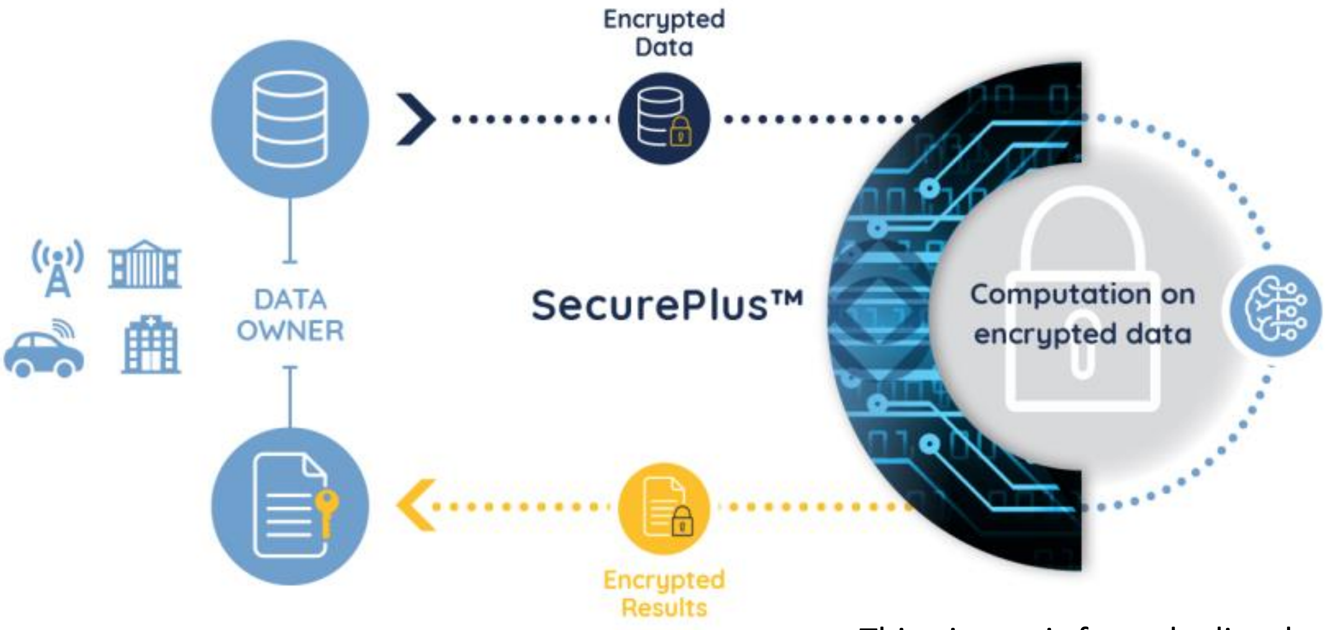
It will be nice to be able to….

Encrypt my query to the cloud

While still allowing the cloud the process them

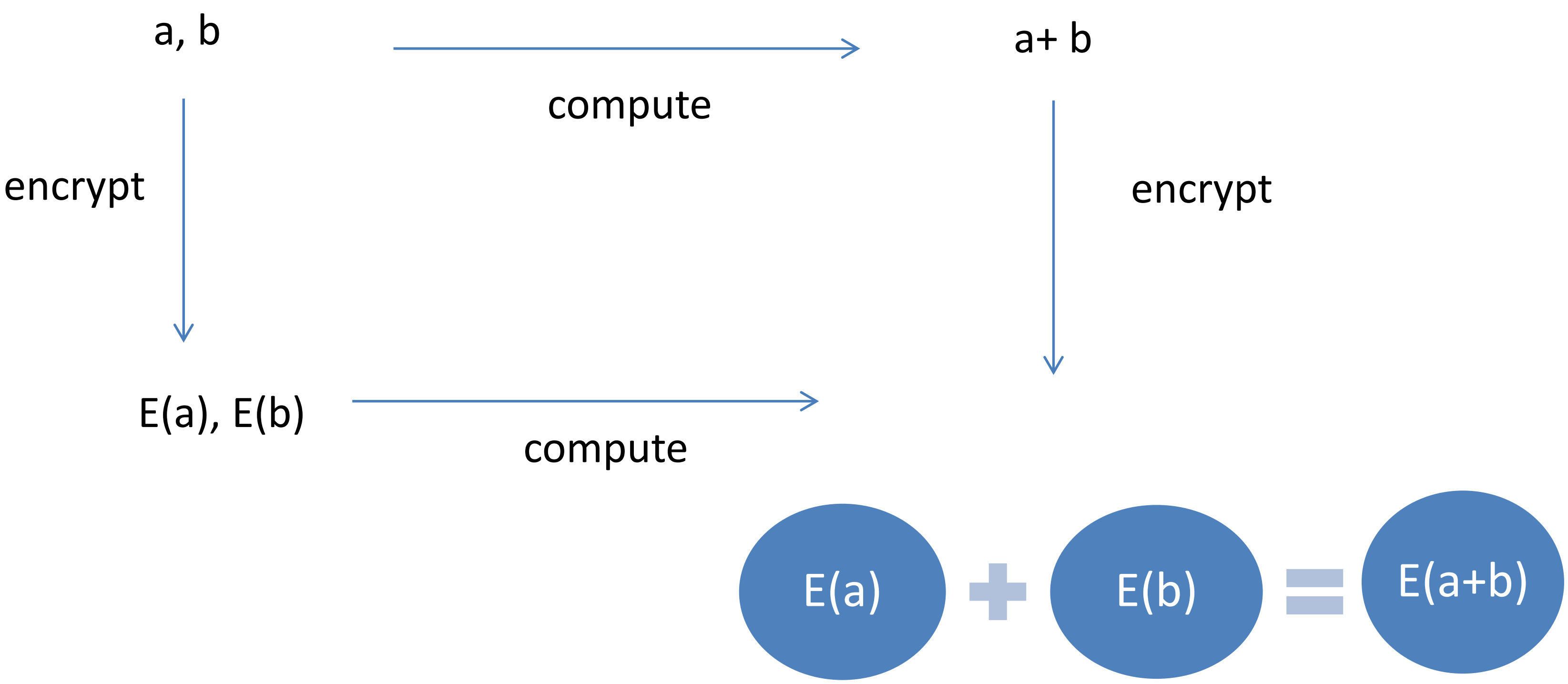
Cloud returns encrypted answers

that I can decrypt



This picture is from duality.cloud

Homomorphic addition



Pure RSA support homomorphic addition!

Homomorphic multiplication

$$
\begin{tikzcd}
a, b \arrow[rr, "\text{compute}"'] \arrow[d, "\text{encrypt}"'] & & a \times b \arrow[d, "\text{encrypt}"] \\
E(a), E(b) \arrow[rr, "\text{compute}"'] & & {}
\end{tikzcd}
$$

$$E(a) \times E(b) = E(ab)$$

Pure Elgamal support homomorphic multiplication!

Fully homomorphic encryption

$$\begin{tikzcd}
x1, \ldots, xn \arrow[r, "\text{compute}"'] \arrow[d, "\text{encrypt}"'] & F(x1,\ldots,xn) \arrow[d, "\text{encrypt}"] \\
E(x1),\ldots, E(xn) \arrow[r, "\text{compute}"'] & \boxed{F(E(x1),\ldots, E(xn))=E(f(x1,\ldots xn))}
\end{tikzcd}$$

Pure Elgamal support homomorphic multiplication!

Protecting Data via Encryption
A famous metaphor

1. Put your gold in the locked box
2. Keep your key
3. Let the jewelry worker work on it through a glove box
4. Unlock the box and get the result

Applications of HE

Outsourcing computation

Machine learning on encrypted data

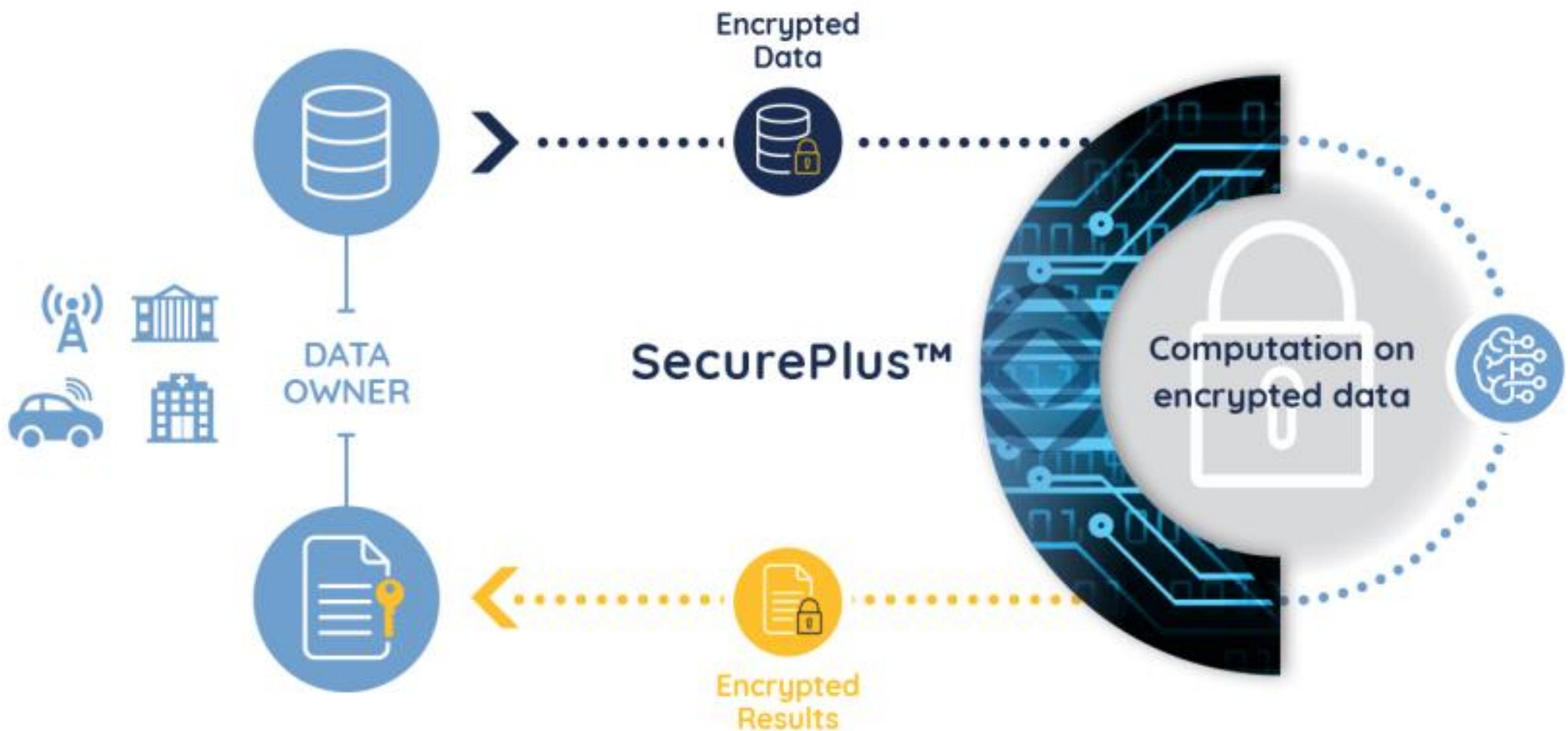
Private cloud storage+computation service

There are two kinds of applications:

private data, public function

private data, private function

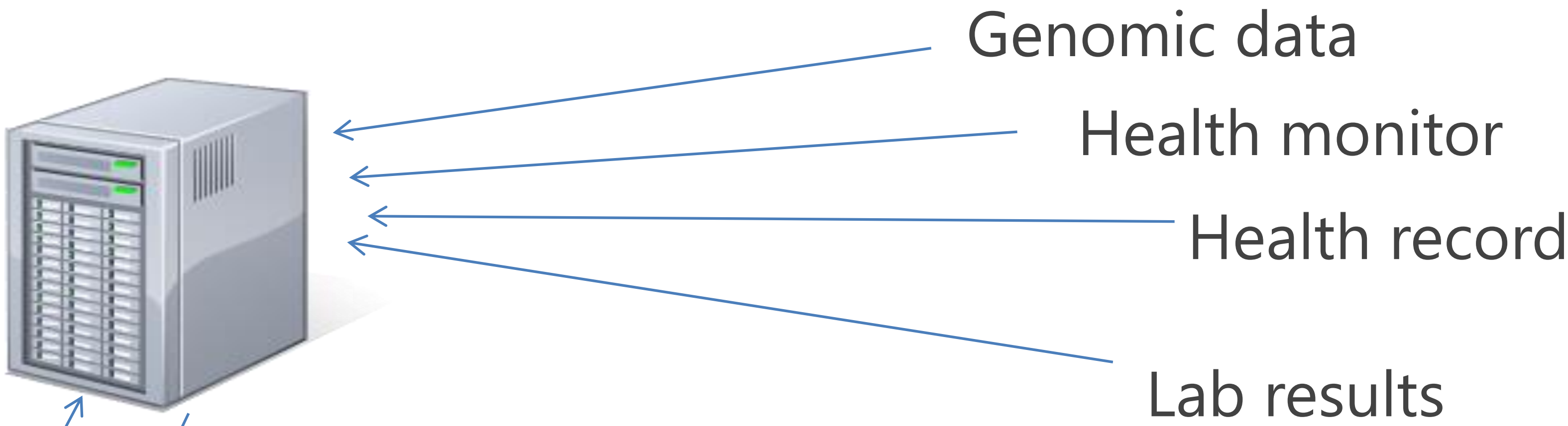
Private data, Public function



This picture is from duality.cloud

Data should be kept secret.
The function f can be public.

Disease prediction



private data, public function

- All data uploaded to the server encrypted under patient’s public or private key
- Cloud operates on encrypted data and returns encrypted predictive results

Private data , Private function

This picture is from duality.cloud

Both data and the model should be kept secret.

Circuit privacy:

An additional requirement in many FHE applications is that the evaluated ciphertext should also hide the function f.

Microsoft

SEAL

Simple Encrypted Arithmetic Library

Quick Background

Homomorphic Encryption library from Microsoft Research

First version released in 2015; currently at version 3.2

Available at https://GitHub.com/Microsoft/SEAL (MIT license)

Developed in standard C++

Implements BFV and CKKS schemes

Simple and easy to use

Comes with detailed examples

Performance of SEAL

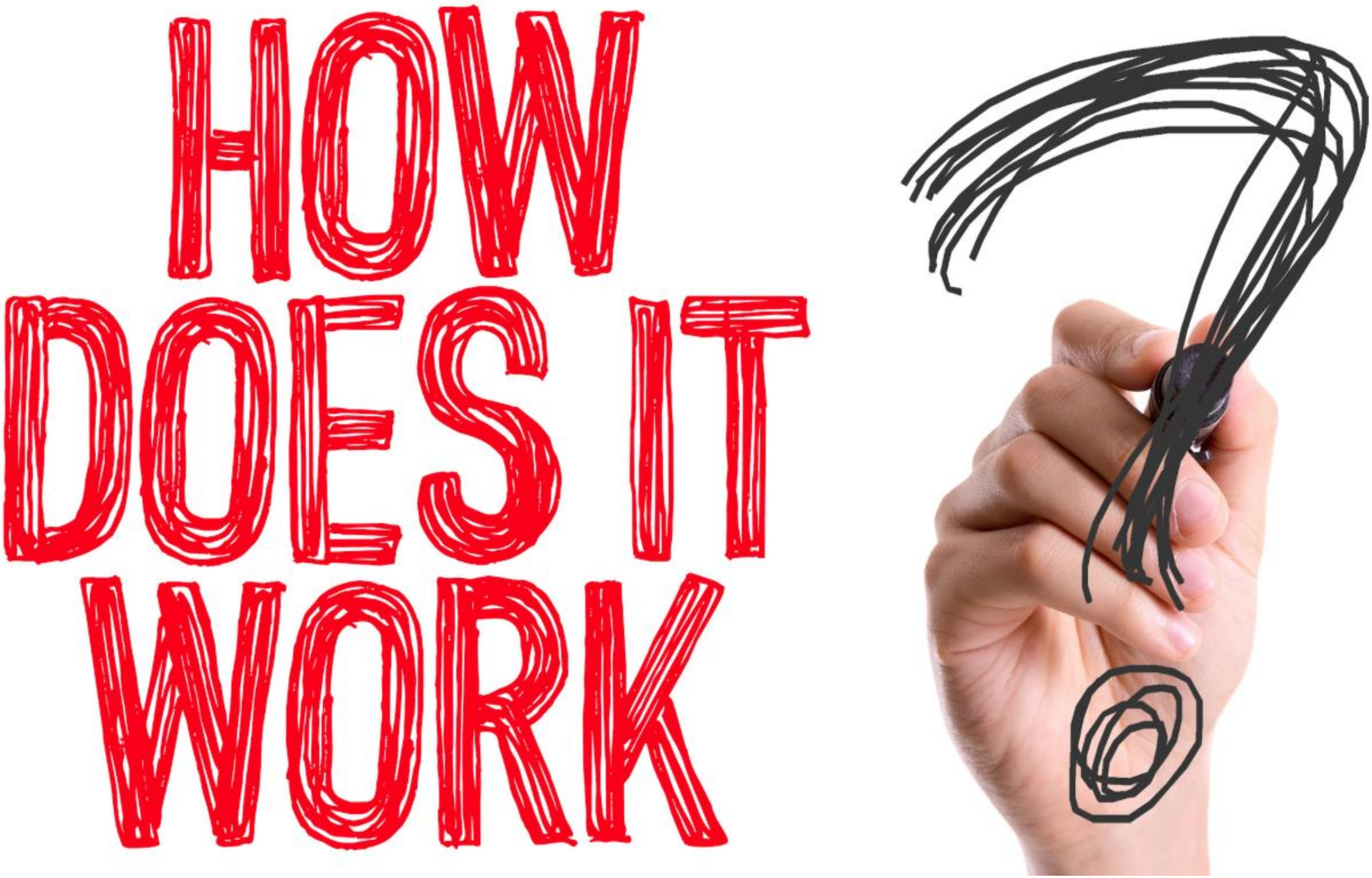
CryptoNets (2016)

- **MNIST handwritten digit classification**
- **60,000 encrypted predictions per hour; 16 per second**
- **99% accuracy**
- **Today can be done probably 100-1000x faster**

Our experiments

- **Logistic regression prediction**
  - **10,000 pieces of data in 5 minutes**
  - **300 times slower than using sklearn directly on plaintext**
- **Summing 100,000,000 random floats between 0 and 100**
  - **860 ms; 8.1x message expansion rate: sizeof(encrypted)/sizeof(plain)**

Seems reasonable


HOW
DOES IT
WORK

Ring-LWE

Ring $R=Z_q[x]/(x^n+1)$

Given:

a_1, $b_1=a_1 \cdot s+e_1$

a_2, $b_2=a_2 \cdot s+e_2$

…

a_k, $b_3=a_k \cdot s+e_k$

To make it simple to understand
you can think all of these are integers

Find: s

s is random in R

e_i are "small" (distribution symmetric around 0)

Decision Ring-LWE

Ring $R=Z_q[x]/(x^n+1)$

Given:

a_1, b_1

a_2, b_2

…

a_k, b_k

Question: Does there exist an s and "small"

$e_1, \ldots, e_k$ such that $b_i=a_i \cdot s+e_i$

or are all b_i uniformly random in R?

BFV key pair

SecretKeyGen():

sample secret key $s \in \chi$

Over-simplified!

You can think all these are integers

PublicKeyGen(s):

sample $a \in R_q$, $e \in \chi$

$pk_0 = -(a \cdot s + e)$ $\quad$ $pk_1 = a$

Ring-LWE pair

s cannot be recovered

BFV encryption

Encrypt(m): sample $u \in R_q$, $e_1\ e_2 \in \chi$

$c_0 = pk_0 \cdot u + e_1 + \Delta \cdot m,$ $\qquad c_1 = pk_1 \cdot u + e_2$

Replace pk with $(-(a \cdot s + e), a)$

$c_0 = -(a \cdot s + e) \cdot u + e_1 + \Delta \cdot m,$ $\qquad c_1 = a \cdot u + e_2$

$c_0 = \underline{-w \cdot s} + e_1 + e \cdot u + \Delta \cdot m\ ,$ $\qquad c_1 = \underline{w} + e_2$

Decision Ring-LWE pair (cannot be distinguished with random value)
Message m is encrypted with a random pad

Ciphertext can be consider as a polynomial:

$f(x) = c_0 + c_1 * x$

BFV decryption

Decrypt(c):

$f(x)=c_0+c_1 \cdot x$

substitute x with s

$f(s)=c_0+c_1 \cdot s$

Replace c with $=([-w \cdot s + e_1 + e \cdot u + \Delta \cdot m\,]_q, [w + e_2]_q)$

$f(s)=-w \cdot s + e_1 + e \cdot u + \Delta \cdot m + (w + e_2) \cdot s$

$=\underline{e_1 + e \cdot u + e_2 \cdot s} + \Delta \cdot m$

Much smaller than Δ

Then we can recover m

We can think that:

$f(s)= v + \Delta \cdot m$

where v is much smaller than Δ

Homomorphic addition

Homomorphic addition:

Ciphertext1: $f_1(x)$

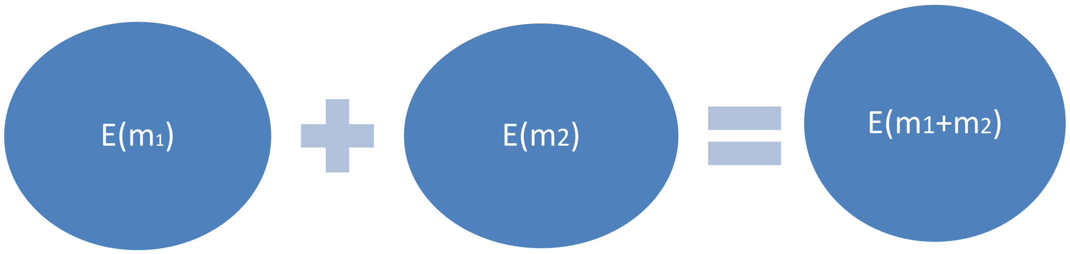
Ciphertext2: $f_2(x)$

Compute: $f_3(x)=f_1(x)+f_2(x)$

We have:

$f_1(s)=v_1+\Delta \cdot m_1$

$f_2(s)=v_2+\Delta \cdot m_2$



Then decrypt $f_3(x)$:

$$f_3(s)=f_1(s)+f_2(s)=v_1+v_2+\Delta \cdot (m_1+m_2)$$

$$=v_3+\Delta \cdot (m_1+m_2)$$

Homomorphic multiplication

Homomorphic multiplication:

Ciphertext1: $f_1(x)$

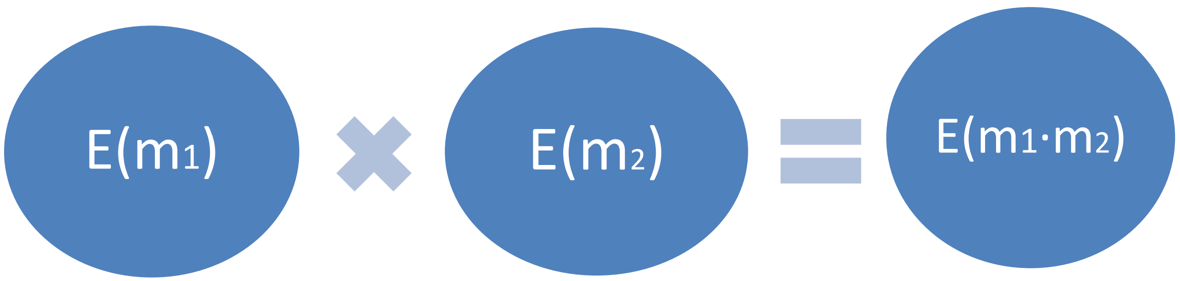
Ciphertext2: $f_2(x)$

Compute: $f_3(x)=f_1(x)*f_2(x)$

We have:

$f_1(s)=v_1+\Delta \cdot m_1$

$f_2(s)=v_2+\Delta \cdot m_2$



Then decrypt f3(x):

$f_3(s)=f_1(s)*f_2(s)=v_1 \cdot v_2+\Delta \cdot (v_1 \cdot m_2+v_2\cdot m_1)+\Delta_2 \cdot m_1 \cdot m_2$

Divide by Δ , we can get:

$f_3(s)/\Delta= v_3+\Delta_2\cdot (m_1 \cdot m_2)$

More about BFV scheme

This is just a over simplified version of BFV

- But enough to make you us understand the problems

For more details about BFV

- Read the paper:
  - Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: CRYPTO 2012 - Volume 7417. pp. 868–886 (2012)
  - Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012)

You can play with BFV

- I write a sage version of simplified BFV
- https://github.com/edwardz246003/danger-of-using-homomorphic-encryption/blob/master/BFV.py

Security of BFV scheme

Encrypt message m to an polynomial f(x)

Decrypt by substitute x with s

get $f(s)=v+\Delta \cdot m$, can recover m easily

Message is “blind” by Ring-LWE pair + noise

Distinguish ciphertext → distinguish Ring-LWE pair

Provable security: IND-CPA → Ring-LWE

Chosen plaintext attack

If someone break IND-CPA, he can break Ring-LWE

Ring-LWE is supposed to be a hard math problem

IND-CCA?

Chosen ciphertext attack

Attacker is given access to a *decryption oracle*

BFV doesn't have IND-CCA security

All practical FHE cannot guarantee IND-CCA

Homomorphic property seems to conflict with CCA

Theoretical research on CCA FHE

Chosen-Ciphertext Secure Fully Homomorphic Encryption

No FHE implementation can guarantee security in the IND-CCA scenario

IND-CCA Scenario

Why need IND-CCA

Attacker **may** be able to ask for a decryption in real scenario

IND-CCA is a standard requirement for normal encryption schemes

Scenarios that require HE often require IND-CCA

Outsourced computation by HE seems in CPA model

Rich data flows between data-owner and cloud

Multi-party's cooperation and data exchange

If certain decrypted data is leaked to the cloud

break the CPA model, need CCA security

One query attack

Suppose attacker can query decryption oracle 1 time

Realistic in many scenarios

Ask to decrypt a malicious ciphertext f(x)

$f(x)=c_0+c_1x$ with $c_0=0$, $c_1=\Delta$

Decryption substitutes x with s

We get: $f(s)=\Delta s$

Then decrypted message equal to s (private key)

Recover private key with only one query

Extremely dangerous

Other FHE schemes face the same problem

Demo of one query attack

```python
def recover_key():
    cc0=0
    cc1=delta
    print "Recover private key successfully:", (s).list()==decrypt([cc0, cc1]).list()
    print "Secret key:",Roundt(s).list()
    print "recovered key",decrypt([cc0, cc1]).list()

s, pk=gen()
rlk, E=gen_rlk(T)
recover_key()
```

```
Recover private key successfully: True
Secret key: [1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
```

Default parameter of SEAL

https://github.com/edwardz246003/danger-of-using-homomorphic-encryption/blob/master/CCA_attack.py

Countermeasures

Never use HE in any scenario where decrypted result may leak to evaluator.

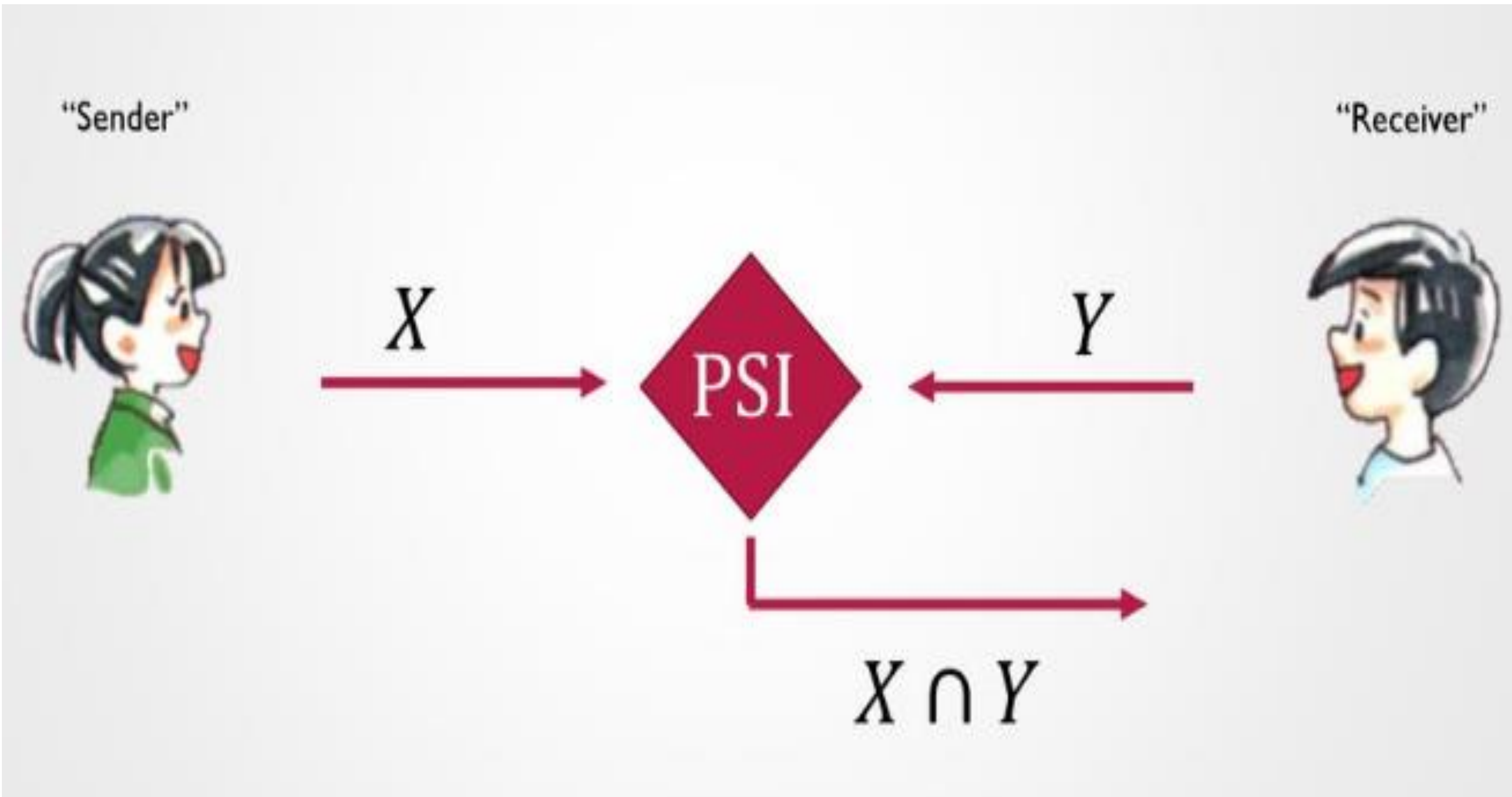
Otherwise, there is no encryption at all.

The decrypted result may leak to evaluator in many scenarios, with or without being noticed.

But how can we make sure there is no leakage?

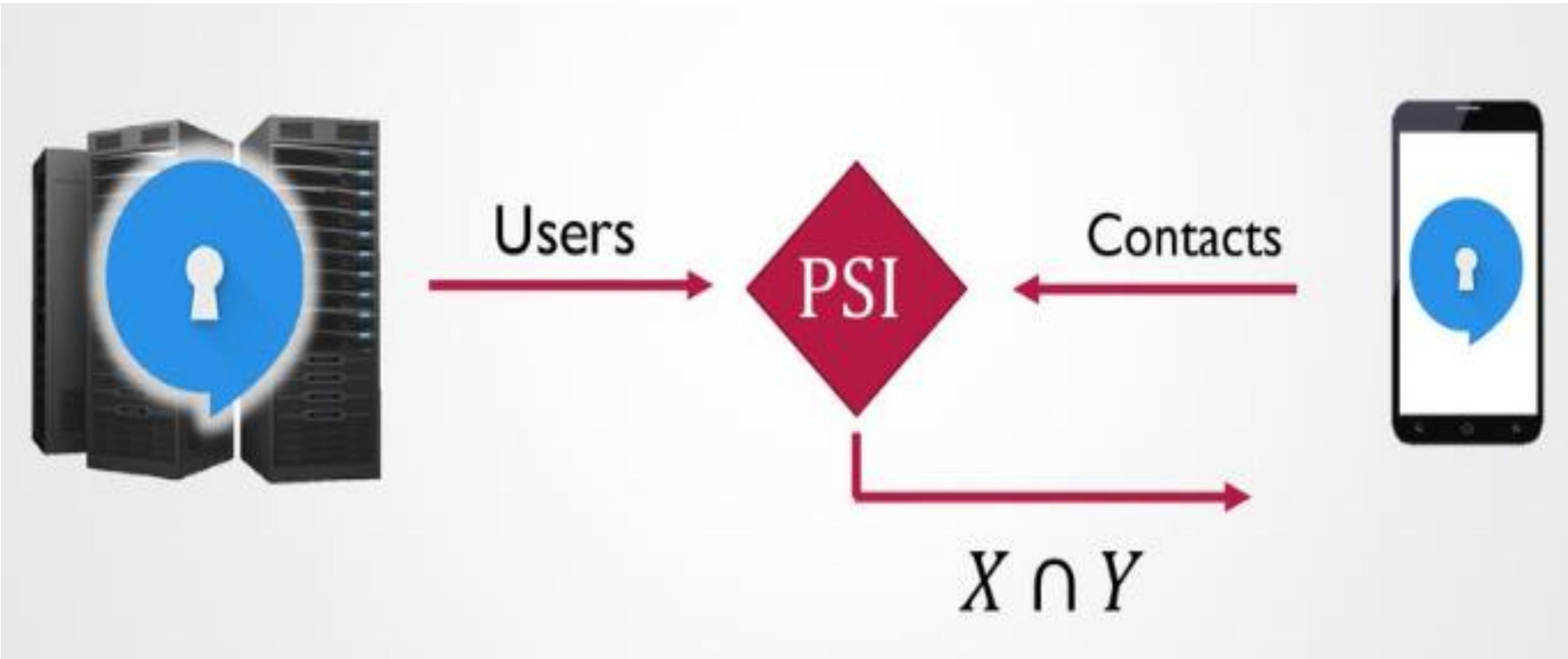
Collaboration with Microsoft SEAL team on building new types of mitigations for this issue.
Should be available shortly in SEAL.

Private Set Intersection (PSI)



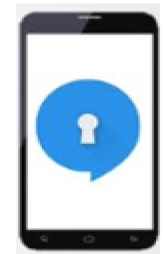
Without leaking anything else

App: Contact discovery



Private contact discovery on E2EE IM (Signal…)

Using HE to build PSI
FPSI in CCS17 (oversimplified)



local database Y

Encrypt(X) with HE

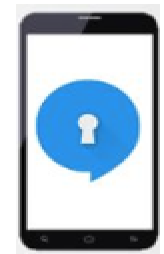
Send Encrypted X to Server

Compute with local database Y,
get the encrypted X∩Y

Send Encrypted X∩Y to Client

Decrypt the result,
Get the X∩Y

CCA attack on this scenario



After client get $X \cap Y$.
He found out that $X \cap Y$ are also using signal.
Then he add them as friends

$X \cap Y$ in plaintext

Information leakage to server.
Server can launch CCA attack!

Lesson learned:

There are always unexpected data flows between data-owner and cloud.
Be extremely careful when using homomorphic encryption.

1 bit leakage

Difference from previous CCA attack

User will check whether the decryption result is 0

So only 1 bit information leakage per query

We can reveal 1 bit of user's private key using this 1 bit information leakage.

Data owner do not need to leak the plaintext to the attack, any 1bit information leakage will lead to 1 bit key leakage

Security of BFV will drop exponentially

1 bit information leakage is Inevitable in real life application

It's very dangerous

Demo of CCA attack (1 bit infoleak)

```
def recover_key(i):
    t1=[0 for _ in range(d)]
    t1[i]=M
    t2=M
    cc0=pk[0]+R(t1)
    cc1=pk[1]+R(t2)
    return (decrypt([cc0,cc1]).list())[i]

s,pk=gen()
rlk,E=gen_rlk(T)
M=delta//4+50
Recoverd_key=[]
for i in range(d):
    Recoverd_key.append(recover_key(i))
print "Recover private key successfully:",Recoverd_key==s.list()
```

```
Recover private key successfully: True
```

Any 1bit information leakage will lead to 1 bit key leakage

https://github.com/edwardz246003/danger-of-using-homomorphic-encryption/blob/master/CCA_attack.py

Another attack

local database Y

Encrypt(X) with HE

Send Encrypted X to Server

Compute with local database Y; get the encrypted X∩Y .
But most of the HE have no circuit privacy.
Other information except X∩Y may also leak.

Send Encrypted X∩Y to Client

Client decrypt the result,
Get the X∩Y
Also get other information on Y

More details can be found in the paper

```
ma=randint(0, t-1) # Alice's input
mb=randint(0, t-1) # Bob's input
print "Alice has input", ma
print "Bob has input", mb
#Alice encrypt her input
#Alice keep u, e1 , e2
u=sample_2()
e1=sample_e()
e2=sample_e()
ca=(Roundq(pk[0]*u+e1+delta*ma), Roundq(pk[1]*u+e2))
#Then Alice send the ciphertext ca to bob

#Bob receive ca, and do some homomorphic computation
cab=(Roundq(ca[0]-delta*mb), Roundq(ca[1])) # plus const mb
r=randint(0, t)
print "Bob choose a random factor, r:", r
cab=(Roundq(r*cab[0]), Roundq(r*cab[1])) # mul a random number r
#Bob respond cab back to Alice

result=decrypt(cab)
print "is mb==mb?", result==0

# A semi-honest Alice can recover the r by using
for i in range(t):
    if Roundq(i*ca[1])==Roundq(cab[1]):
        break
r_prime=i
print "Alice recover r':", r_prime
print "Is r' equals to r:", r_prime==r

#Alice can use r to recover Bob's input mb
mb_prime=(ma-(int(result)*inverse_mod(r_prime, t)))%t
print "Alice recover mb' ", mb_prime
print "Is mb' equals to mb:", mb_prime==mb
```

Circuit Privacy of SEAL

SEAL doesn’t provide circuit privacy on default

Addressed in FPSI paper and in old *SEAL manual* (no longer available)

Best practice is “noise flooding”

adding an encrypted 0 to the final result, with “enough” noise

But there is no standard interface of “noise flooding” in SEAL

normal software developer definitely can’t play with the magic

Hardness of providing “noise flooding”

Need to know how much noise is needed, this is also some kind of information we need to protect. :(

All practical FHE lib seems have the circuit privacy problem

There are *ad hoc* workarounds but no generic solution

Solutions require crypto expertise to implement

Countermeasures

An improved PSI protocol is published in CCS18

- https://eprint.iacr.org/2018/787
- Solve the PSI problem
- Non-generic solution

As for circuit privacy of SEAL and HE

- You need a crypto expert to review your implementation
- Professional knowledge on lattice-based crypto
- SEAL team is considering a standard interface to deal with this issue

Coding Information Disclosure in SEAL

HE is working on a polynomial ring based on finite field

- plaintext is integer, float or string
- we need convert them to the ring

IntegerEncoder of SEAL

- Encode an integer to a polynomial
- Many to one mapping
  - Information leakage!

More details can be found in the paper.

I think you don't want the mathematical formula here.

Demo of Encoder

```
m1=IntegerEncoder(2)
m2=IntegerEncoder(2)
c1=encrypt(m1)
c2=encrypt(m2)
result=add(c1,c2)
print decrypt(result)
print IntegerDecoder(decrypt(result))
```

```
2*X
4
```

```
m1=IntegerEncoder(1)
m2=IntegerEncoder(3)
c1=encrypt(m1)
c2=encrypt(m2)
result=add(c1,c2)
print decrypt(result)
print IntegerDecoder(decrypt(result))
```

```
X + 2
4
```

Demo of information leakage of IntegerEncoder in SEAL

We have m1 and m2. We want:

result= m1 + m2

For m1=2 and m2=2, we have

result = 4

For m1=1 and m2=3, we have

result = 4

But the encoded form are different!

2*X != X+2

Attack on millionaire problem:

https://github.com/edwardz246003/danger-of-using-homomorphic-encryption/blob/master/millionaire.py

Countermeasures

Coding problem may also affect other HE lib

- Be careful when using encode functionality provided by HE libraries
- The crypto part have a security proof, but the encoding may not

Don’t use IntegerEncoder in SEAL without understanding security model

- IntegerEncoder is primarily a demonstrative tool
- BatchEncoder and CKKSEncoder don’t have this issue

Other security issues

HE does not provide the security features as the commonly known encryption algorithms

- HE is not an Authenticated Encryption
- Cannot guarantee the integrity of the data
- Attacker can use homomorphic nature of HE to modify ciphertext
- Don’t use HE for storage and data transmission directly
- Consider wrapping HE traffic e.g. inside TLS

We need a standard documentation for HE

- Microsoft is currently leading the development of standard for HE
- Important to include protocol security topics to standard!

Can FHE really compute arbitrary functions?

$$\begin{tikzcd}
x1, \ldots, xn \arrow[r, "\text{compute}"'] \arrow[d, "\text{encrypt}"'] & F(x1,\ldots,xn) \arrow[d, "\text{encrypt}"] \\
E(x1),\ldots, E(xn) \arrow[r, "\text{compute}"'] & \boxed{F(E(x1),\ldots, E(xn))=E(f(x1,\ldots xn))}
\end{tikzcd}$$

Arbitrary function in FHE means arbitrary addition and multiplication here.

Arbitrary addition and multiplication does not mean you can run arbitrary program

You can't do comparison directly

(if branch is not support here)

Update the famous metaphors

1. Put your gold in the locked box

This box should be opaque!

2. Keep your key
3. Let the jewelry worker work on it through a glove box
with eyeshade
4. Unlock the box and get the result

Conclusion

HE is a useful in many scenarios

Its performance is improving and acceptable

HE is not omnipotent

It can not run arbitrary program

HE has many security pitfalls

It is extremely dangerous to use HE without a crypto expert for now.

Community needs to focus on building secure protocols.

Acknowledgements

We would like to thank

Kim Laine of Microsoft Research

Chen Hong of Alibaba Gemini Lab

For their valuable comments and suggestions to the talk

Thanks

360
WWW.360.cn
SAFETY FIRST