Privacy Preserving Machine Learning

Angel Navia Vázquez
Machine Learning for Data Science (ML4DS) Group

MLG, may 2018

Telling stories: Alice and Bob

- *Alice:* Take this suitcase and please count the money
- *Bob*: Sure! But it is locked. Please, give me key.
- *Alice:* I am sorry. I can´t trust you. You have to count it without opening it.
- *Bob*: Then, it can't be done!

2

What is Privacy Preserving Machine Learning (PPML)?

- Some users have some data
- A computation has to be carried out without revealing the data

- *Alice:* Take these encripted files. Please sum all the numbers in them.
- *Bob*: Sure! Please, give me decryption key.
- *Alice:* I am sorry. I can´t trust you. You have to sum them without decrypting the files.
- *Bob*: Hummmmm… let me think…

3

How to learn while protecting privacy?

- PPML approaches:
  - **Noise perturbation techniques**: an additive or multiplicative noise is included in the data aiming at hiding the original values. WE'RE NOT INTERESTED
  - **Differential privacy**: probabilistic approach that aims at maximizing the information provided by queries from a database while minimizing the chance of identifying individual records. Relies on sanitization techniques:
    - **Input/output perturbation**: add noise to inputs or computation results. WE'RE NOT INTERESTED.
    - **k-anonymization**: verify that at least k elements provide the same attributes. WE'RE NOT INTERESTED.
  - **Federated Machine Learning**: updating a master model with incremental updates from every client owning a subset of the training set. Warning! The model is shared…
  - **Distributed Machine Learning**: the exchanged information is not raw data and does not reveal any information concerning individual users. Warning! The model can be shared…
  - **Partially Homomorphic Cryptography**: Homomorphic encryption schemes guarantee some equivalences between operations performed on the clear and encrypted domain. Warning! some operations may not be supported! Trusted Third Party may be necessary.
  - **Fully Homomorphic Cryptography:** schemes that guarantee the equivalence between any operation performed on the clear and encrypted domain. Complexity? Feasibility?
  - **Secure Multiparty Computation**: every user only gets as information the output of a function on its own data. Warning! Very restrictive, some operations can be very complex / time consuming.

4

Differential privacy

- Privacy definition: With or without including me in the database, my privacy risk should not change much
- Privacy is lost if:
  - An individual record is leaked
  - If a cross-analysis with public data allows to identify individuals
  - A published statistics S allows to better estimate a private value, *e.g.: Alice's height is 2 cm above average height in the population.*

Hospital Patient Data

DOB	Sex	Zipcode	Disease
1/21/76	Male	53715	Heart Disease
4/13/86	Female	53715	Hepatitis
2/28/76	Male	53703	Brochitis
1/21/76	Male	53703	Broken Arm
4/13/86	Female	53706	Flu
2/28/76	Female	53706	Hang Nail

Vote Registration Data

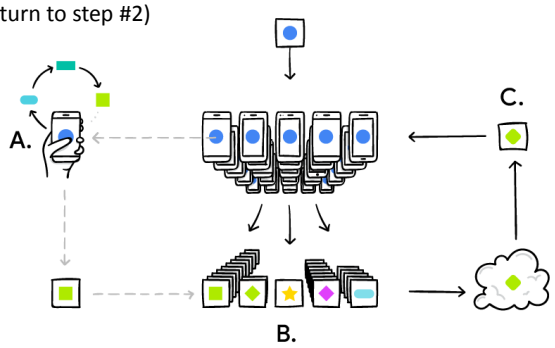
Name	DOB	Sex	Zipcode
Andre	1/21/76	Male	53715
Beth	1/10/81	Female	55410
Carol	10/1/44	Female	90210
Dan	2/21/84	Male	02174
Ellen	4/19/72	Female	02237

Information leak!!!: Andre has heart disease!

5

Federated Machine Learning

- *Google's Federated Averaging* algorithm (MacMahan et al.):
  1. Construct some master model with weights **w**.
  2. Download your master model to each of your clients.
  3. For each client, compute an updated, personalized set of weights $\mathbf{w}_k$ using local training data — this is done in parallel and offline. Data does not leave the client.
  4. Now, update **w** with the weighted average of all $\mathbf{w}_k$.
  5. (Return to step #2)



6

Federated Machine Learning (II)

- Useful when:
  - Users have Privacy Protected Data
  - Data cannot leave the users' facilities
  - Computation is decentralized
- The algorithms must be robust against:
  - Non IID data
  - Asynchronous updates
  - Unbalanced data
  - Massive distribution
  - Dynamic datasets
  - Other restrictions: *in mobiles, only operate when plugged in, connected to wifi, etc.*
- We must minimize:
  - The amount of information to be transmitted
  - The communication cost
  - The local computation (low cost processors)

7

Federated Machine Learning (III)

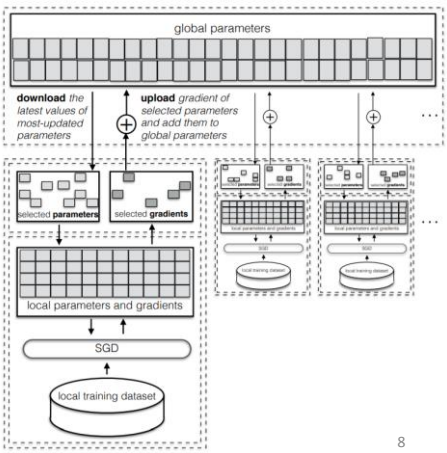
- Complex models better preserve the privacy:
  - *E.g.: in linear language model on sparse data, the nonzero gradients reveal the typed words… the less explicable, the better…*
- Deep Learning experiments

[Shokri'15]

- SSGD Selective Stochastic Gradient Descent: helps to better generalize
- Even sharing 1% of the parameters improves with respect to standalone learning

	SGD	0.1	0.01	0.001	Standalone
MNIST, CNN	0.9917	0.9914	0.9871	0.9645	0.9316
SVHN, CNN	0.9299	0.9312	0.8986	0.7481	0.8182

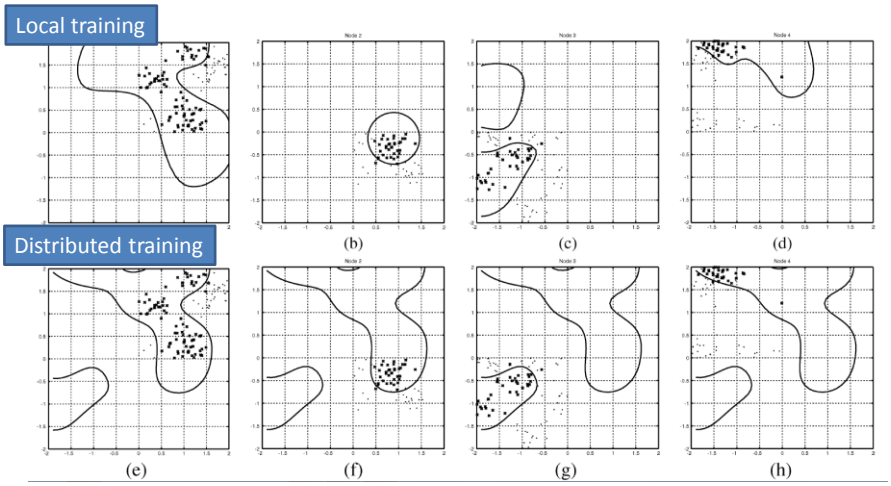
	SGD	0.1	0.01	0.001	Standalone
MNIST, MLP	0.9810	0.98	0.9707	0.9171	0.8832
SVHN, MLP	0.8476	0.8394	0.7833	0.6542	0.5136



8

Distributed Machine Learning

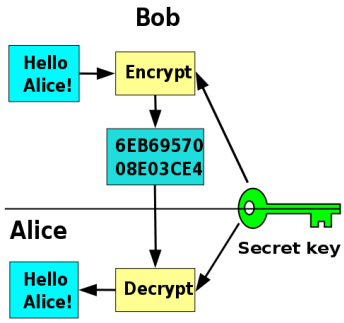
- Data is located in different nodes
- Partial computations are locally done and shared (K^TK, K^Ty)
- The final model is adjusted. e.g: Distributed SVM training [Navia'06]



9

Alice and Bob are back!

- Symmetric key cryptography
  - A single (secret) key is used to encrypt and decrypt
  - The only method known until 1976
  - Key management is a problem:
    - How to securely send the key to Alice?
    - An increasing number of keys is needed (square with the number of network members)

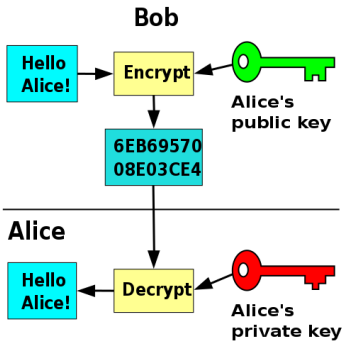


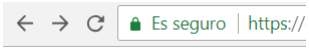
- Asymmetric or public key cryptographic system is proposed in 1976: W. Diffie and M. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory. IT-22 (6): 644–654. 1976.

10

Public Key cryptography

- The encription key is public
- The decription key is private
- Both keys are related, but it is unfeasible to obtain the private key from the public key
- The public key can be freely distributed
- Diffie and Hellman did not find the complete system, only the protocol for key sharing
- 1978: **R**ivest, **S**hamir and **A**dleman defined a working system, currently known as the RSA algorithm.
- Used in most popular digital signature systems: SSL/TLS, VPNs, etc.





11

Homomorphisms

- Algebra definition:
  - structure-preserving map between two algebraic structures of the same type (two groups, two rings, etc.)
- Example:
  - G_1: Real numbers are a group for addition
  - G_2: Positive real numbers are a group for multiplication
  - The exponential function $f(x) = e^x$ is a homomorphism between them:
  - $f(x+y) = e^{x+y} = e^x e^y$ (addition in G_1 is equivalent to multiplication in G_2
- Ring homomorphism: map between rings that preserves the addition, multiplication, and the multiplicative identity
  - $f(a + b) = f(a) + f(b)$ for all a and b in R
  - $f(ab) = f(a)\, f(b)$ for all a and b in R
  - $f(1_R) = 1_S$

12

Homomorphic encryption

- **Partially homomorphic encryption PHE** (either sum or product are supported)
  - Unpadded RSA (multiplicative)
  - ElGamal (multiplicative)
  - Goldwasser–Micali (additive)
  - Benaloh (additive)
  - Paillier (additive)
- **Somewhat homomorphic encryption SHE**:
  - Also known as (XOR, AND)-homomorphic encryption
  - (XOR, AND) defines a Turing-complete system, we can compute any function on encrypted bits
  - Allows many additions and *some* multiplications
- **Fully homomorphic encryption FHE**
  - Supports both sum and product
  - It preserves ring structure (A, +, •))
  - Examples:
    - Gentry's cryptosystem [Gentry09]
    - Cryptosystem over the integers
    - 2nd generation

13

Paillier cryptosystem

- Paillier cryptosystem exploits the fact that certain discrete logarithms can be computed easily
- Powers of a a multiplicative subgroup $G = \{\ldots, b^{-3}, b^{-2}, b^{-1}, 1, b^1, b^2, b^3, \ldots\}$ allow an easy computation of $\log_b a$ *(e.g.* $\log_b\ b^3 = 3$*)*
- *Binomial Theorem:*

$$(1+n)^x = \sum_{k=0}^{x} \binom{x}{k} n^k = 1 + nx + \binom{x}{2} n^2 + \text{higher powers of } n$$

- *Then:*

$$(1+n)^x \equiv 1 + nx \pmod{n^2}$$

$$y = (1+n)^x \bmod n^2$$

$$(y = 1 + nx \pmod{n^2})$$

$$x \equiv \frac{y-1}{n} \pmod{n}.$$

$$L((1+n)^x \bmod n^2) \equiv x \pmod{n}, \qquad L(u) = \frac{u-1}{n}$$

14

Paillier cryptosystem (II)

- **Key generation**:
  - Choose two large prime numbers p and q randomly and independently of each other such that gcd(pq,(p-1)(q-1))=1
  - Compute n=pq and λ = lcm(p-1,q-1)
  - Select random integer g in Z^*n^2
  - check the existence of the following modular multiplicative inverse: $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$
  - The public (encryption) key is (n,g)
  - The private (decryption) key is (λ, μ)
- **Encryption:**
  - Let m be a message to be encrypted where 0 <= m < n
  - Select random r where 0 < r < n (nondeterministic encryption)
  - Compute ciphertext as: $c=(g^m \times r^n) \bmod n^2$
- **Decryption:**
  - Let c be the ciphertext to decrypt, c in Z^*n^2
  - Compute the plaintext message as: $m=L(c^{\lambda} \bmod n^2) \times \mu \bmod n$

15

Paillier homomorphic properties

- The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts

$$D(E(m_1,r_1)) \times E(m_2,r_2) \bmod n^2) = m_1 + m_2 \bmod n$$

- The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts

$$D(E(m_1,r_1)) \times g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n$$

- An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

$$D(E(m_1,r_1))^{m_2} \bmod n^2) = m_1 m_2 \bmod n$$
$$D(E(m_2,r_2))^{m_1} \bmod n^2) = m_1 m_2 \bmod n$$

- An encrypted plaintext raised to a constant k will decrypt to the product of the plaintext and the constant,

$$D(E(m_1,r_1))^k \bmod n^2) = k m_1 \bmod n$$

- Given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.

16

Python example

```
from paillier.paillier import *
priv, pub = generate_keypair(512)
```

<PrivateKey:
798118547958500245259981069765619625628486136783124026485990453392220872813728033186211995518324090070051355953664166
615424331878953264099307183676192082857769585333462739006841720237665186879921732227444670044974240474743823343622275
328057457870259241634233056334083366354464615015820528874187804862924365 1>
<PublicKey:
798118547958500245259981069765619625628486136783124026485990453392220872813746259450778998988157502979179198140 5
347528323252888280515201802857785430853213>

```
x = 3
Y = 5
cx = encrypt(pub, x)
cy = encrypt(pub, y)
```

cx
246412201076731154038001018870106841565814606909735717678669358112537820370610886961033908799214026769403051603 7
050388941325485887713849453957342286518027965018228254436662577890811031885745859841313985730954818431234307273 9
324016605999147562799006820033147344115635251932131316168182737324552612639491410 90L

cy
163965410016549323020073911463271011880696878860861040561316250253147983423731111928193850686920118548853548189 8
226775964079590461908316484601763775640641969640688825242364824752619646480123259784748845452879333921043679273 3
354386977678729706093166702429762725145482584069220181876314585322278921760606900715L

Python example (II)

```
cz = e_add(pub, cx, cy)
```

cz
308743721956558557163990323289550203652837363336018643416848853002649985289658553293037575252852126966976976519883033 2
824514339020469101652356735739842753865924373908934196360464071819330759839646317717301631638297879268716373756768 73
202451578588524167493693647873311640908872331014223482414795776521274552 45L

```
z = decrypt(priv, pub, cz)
z
```
8L

```
cq = e_add_const(pub, cz, 2)
```

cq
336866228544107405684372345216953267869146674574444052817365983881326621398350755863305216417867294096836266102120915 8
296863166093933008688379633639599958420921632753036197345135095767831268515948466260727624820304785468445789089225239 0
852642739418417770605903466796699997142267348199023010959007761169456220L

```
q = decrypt(priv, pub, cq)
z
```
10L

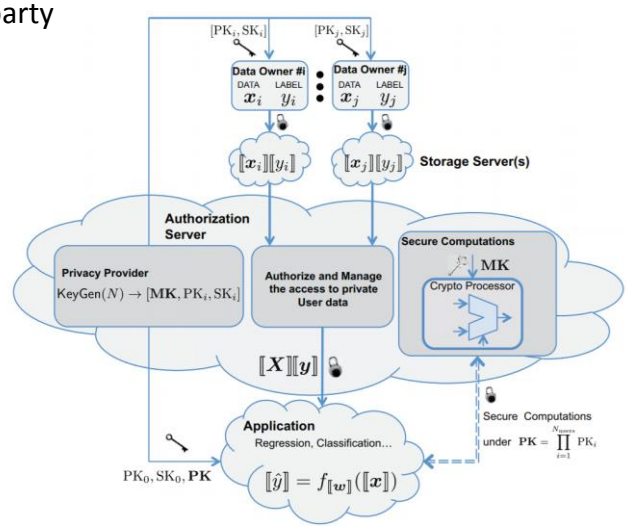
```
cp = e_mul_const(pub, cq, 11)
```

cp
646813598018216077262570768607977562224086941654680088210333908266028369732940115977126870678406078633281335833371196213614 8
479026859601935564043868170959289483897969543379666927154222626517539325443292610580926611431841221608959672924906430960 2
588025034007222673638237184555193156462882653663732090722585 4L

```
p = decrypt(priv, pub, cp)
p
```
110L

Proposed approach #1

- A trusted third party provides the unsupported operations (González17):



Towards FHE: (XOR, AND)-homomorphic encryption with private keys (DGHV'10)

- Choose large odd number p (n^2 bits) -> decryption key (fixed)
- To encrypt bit m:
  - Choose large q (n^5 bits) and small r (n bits), random numbers
  - $c = p \cdot q + 2 \cdot r + m$ (2r is "noise"!). r is needed to avoid recovering the secret key using two encriptions of "0" as GCD($q_1 p$, $q_2 p$)
  - $2 \cdot r + m$ much smaller than p, s.t. $2r + m \pmod 2 = m \pmod 2$
  - ciphertext is close to a multiple of p, s.t. $c \pmod p = (2r + m) \pmod p$
- Decryption:
  - $m \equiv (c \bmod p) \bmod 2$
- Adding (XOR) two encrypted bits ($2(r_1+r_2)$ is "noise"!):

$$c_1 + c_2 = (q_1 + q_2) \cdot p + 2 \cdot (r_1 + r_2) + (m_1 + m_2)$$
$$\hookrightarrow \text{ if } 2 \cdot (r_1 + r_2) + (m_1 + m_2) \text{ much smaller than } p$$
$$\hookrightarrow (c_1 + c_2 \bmod p) \bmod 2 \equiv m_1 + m_2 \pmod 2$$

(XOR, AND)-homomorphic encryption (DGHV'10)

- Multiply (AND) two encrypted bits ($2(r_1 r_2)$ is "noise"!) :

$$
\begin{aligned}
c_1 \cdot c_2 &= \ldots = (q_1 q_2 p + 2q_1 r_2 + q_1 m_2 + 2q_2 r_1 + q_2 m_1)p + \\
&\quad 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2 \\
c_1 q_2 &= q_1 q_2 p + 2q_2 r_1 + q_2 m_1 \\
c_2 q_1 &= q_1 q_2 p + 2q_1 r_2 + q_1 m_2 \\
c_1 \cdot c_2 &= (c_1 q_2 + c_2 q_1 - q_1 q_2 p)p + 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2
\end{aligned}
$$

$\hookrightarrow$ if $2(2r_1 r_2 + \ldots) + m_1 m_2$ much smaller than p

$\quad \hookrightarrow (c_1 \cdot c_2 \bmod p) \bmod 2 \equiv m_1 \cdot m_2 \pmod 2$

- But...
  - Ciphertext grows with each operation: $|c1 + c2| = n + 1$ and $|c1 \cdot c2| = 2 \cdot n$
  - Noise doubles on addition and squares on multiplication… ciphertext may become unrecoverable after some operations.
  - This approach is Somewhat Homomorphic Encryption (SHE)

21

Fully Homomorphic Encryption

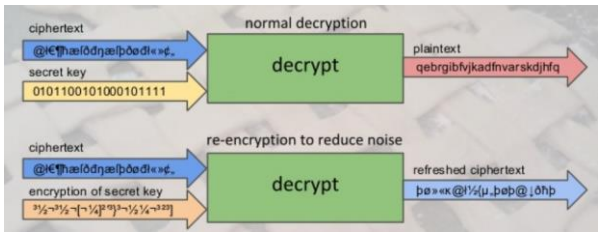
- Gentry's "bootstrapping" theorem:
  - "If an encryption scheme can evaluate its own decryption circuit, then it can evaluate everything"
  - Problem: operations increase noise
  - Decryption reduces noise, but the key is private. How to reduce noise without knowing the private key?
- Steps:
  - Construct a SHE scheme that supports evaluation of D-degree polynomials
  - Squash the decryption procedure and express it as a low degree polynomial (D/2)
  - Apply a bootstrapping transformation to obtain the FHE scheme

22

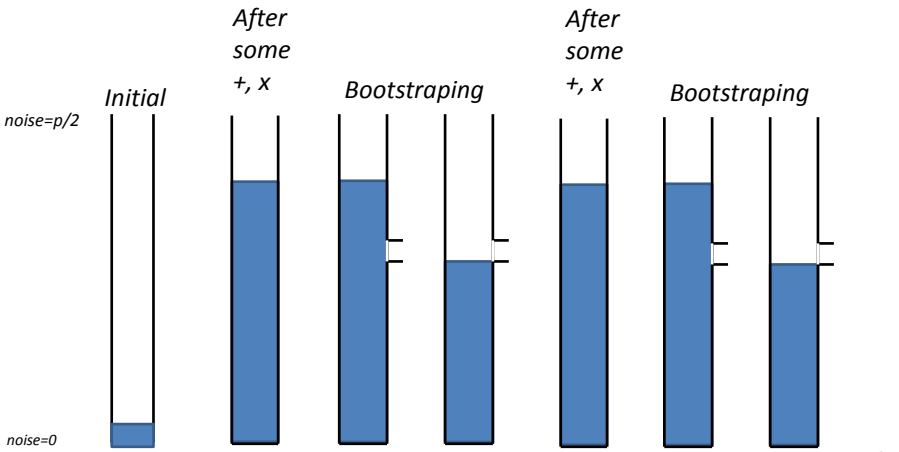
Bootstrapping

RSA&friends:
MANY mult
ZERO add

- How to reduce the noise?:
  - Decryption D(Enc(m), p) reduces noise to zero (but secret key is needed!!!)
  - Use Enc(p) (circular encryption) and homomorphically evaluate the decription Enc(m)* = D(Enc(m), Enc(p))
  - Enc(m)* and Enc(m) have different noise levels
  - Regardless noise level in Enc(m), the noise level in Enc(m)* is fixed!!!

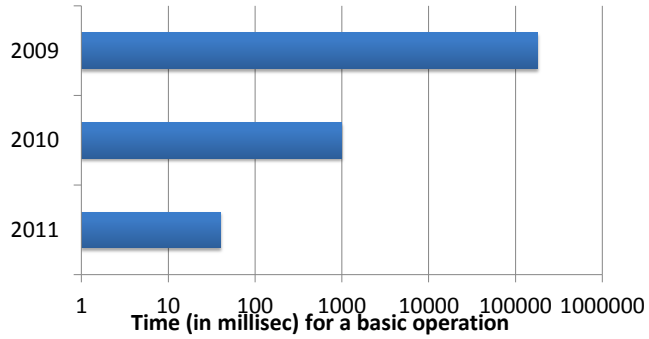


Every Bootstrap stage reduces noise to a fixed level



24

Efficiency improvement over time



25

FHE practical applications

- **Advertising**:
  - a consumer wants product recommendations but without revealing its identity [Jeckmans]
  - A consumer receives encrypted recommendations from a recommender [Armknecht]
  - A consumer wants ro receive contextual adds, but without revealing its position [Naehrig2011]
- **Medical applications**:
  - The patient data is uploaded (encrypted) to a system that analyses it and tracks the users health [Naehrig2011]
- **Data Mining**:
  - Customer data analysis [Yang]
- **Financial privacy**:
  - Outsource computation on private data with private algorithms [Naehrig2011]
- **Image recognition**:
  - Detect illegal images on a hard drive based on a Police database [Bösch]
- **Computation delegation**:
  - Outsource computation to a third party while checking that it was done correctly [Chung]
- **Private internet search**:
  - Retrieve information without revealing the query
- **Spam filtering**:
  - Blacklisting encrypted emails
- **DNA analysis**:
  - Search for DNA markers without revealing DNA itself
- **Electronic voting**:
  - Compute the result of a voting without decrypting the ballots

26

References

- Frederik Armknecht and Thorsten Strufe. **An efficient distributed privacy preserving recommendation system**. In The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2011, pages 65–70. IEEE, 2011.
- Christoph B¨osch, Andreas Peter, Pieter H. Hartel, and Willem Jonker. **SOFIR: securely outsourced forensic image recognition**. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, pages 2694–2698. IEEE, 2014.
- Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. **Improved delegation of computation using fully homomorphic encryption**. In Rabin [52], pages 483–501.
- W. Diffie and M. Hellman. **New Directions in Cryptography**. IEEE Transactions on Information Theory. IT-22 (6): 644–654. 1976.
- González-Serrano F.J, Navia-Vázquez A. and Amor-Martín A. **Training Support Vector Machines with privacy-protected data**. Pattern Recognition. Vol. 72, pp. 93-107, 2017
- Hayes, Brian. (2012). **Alice and Bob in Cipherspace**. American Scientist. 100. 362-367.
- Arjan Jeckmans, Andreas Peter, and Pieter H. Hartel. **Efficient privacy enhanced familiarity-based recommender system**. In Jason Crampton et al., editors, Computer Security – ESORICS 2013, volume 8134 of Lecture Notes in Computer Science, pages 400–417. Springer, 2013.
- McMahan et al. **Communication-Efficient Learning of Deep Networks from Decentralized Data**. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.
- A. Navia-Vázquez, D. Gutiérrez-González, E. Parrado-Hernández, J.J Navarro-Abellán, **Distributed Support Vector Machines**. *IEEE Tr. Neural Networks*, Vol. 17, No. 4, pp. 1091-1097, 2006.
- Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. **Can homomorphic encryption be practical?** In Christian Cachin and Thomas Ristenpart, editors, Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW, pages 113–124. ACM, 2011.
- R. Shokri, V. Shmatikov. **Privacy-Preserving Deep Learning**. CCS'15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310-1321
- C. Gentry. **A fully homomorphic encryption scheme**. PhD Dissertation. Stanford University, 2009
- Dijk, Gentry, Halevi, Vaikuntanathan. **Fully homomorphic encryption over the integers**. EUROCRYPT'10 Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques. pages 24-43, 2010.
- Zhiqiang Yang et al. **Privacy-preserving classification of customer data without loss of accuracy**. In Hillol Kargupta et al., editors, Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, pages 92–102. SIAM, 2005.

27

Thanks for your attention!

28