Privacy Preserving Protocol for Iris Recognition Based on Somewhat Homomorphic Encryption

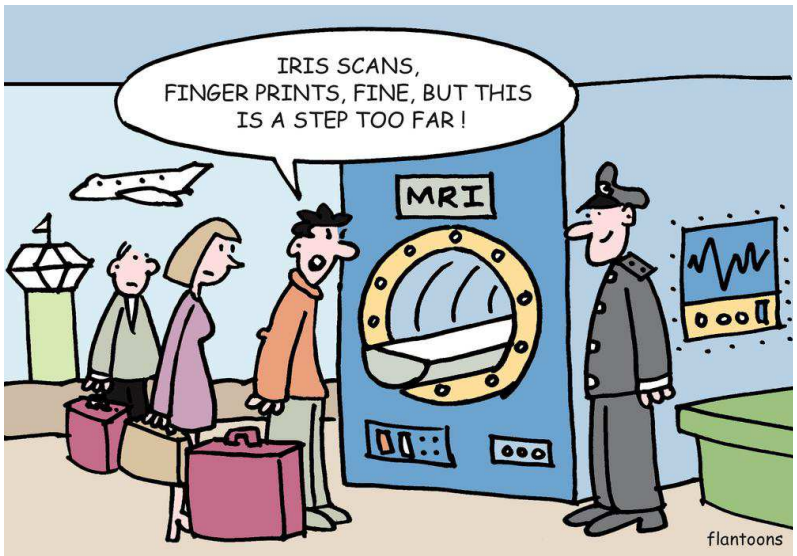
Giulia Droandi, Riccardo Lazzeretti, Mauro Barni, Luca Chiantini

Departement of Information Engineering and Mathematics,
University of Siena, ITALY

March 17, 2014

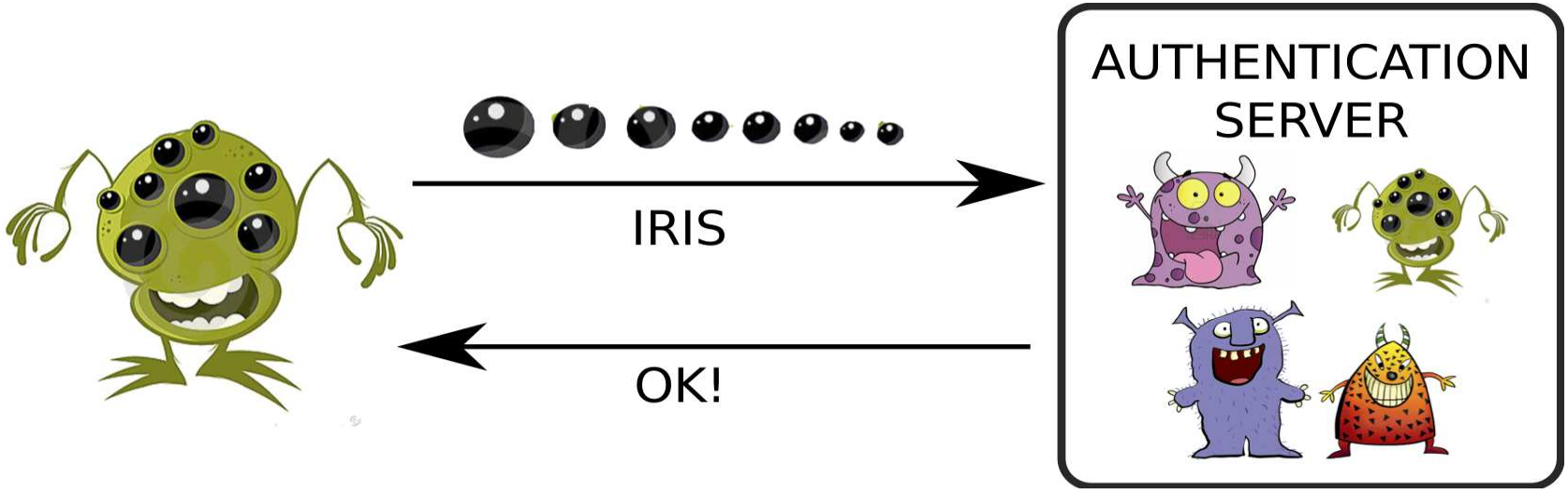
Motivation

Biometric signals often used in access control systems



Motivation

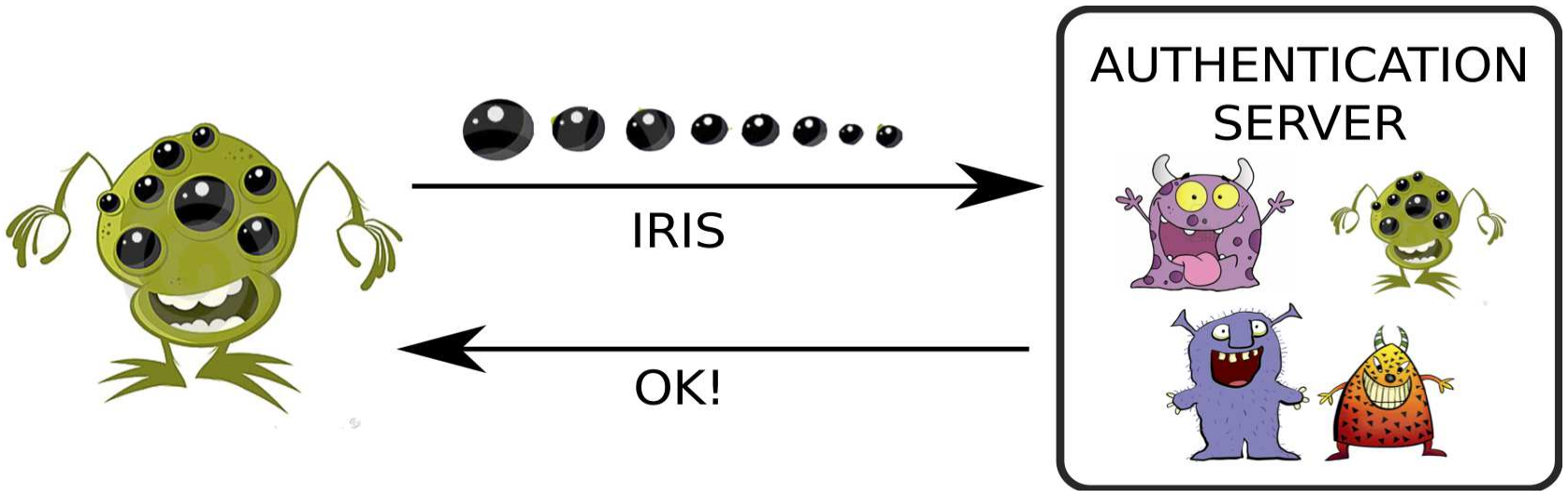
Biometric signals often used in access control systems



Need of privacy protection of both the biometric gallery and the probe

Motivation

Biometric signals often used in access control systems



Need of privacy protection of both the biometric gallery and the probe

Our solution: Somewhat Homomorphic encryption

What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.

What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.

1

0

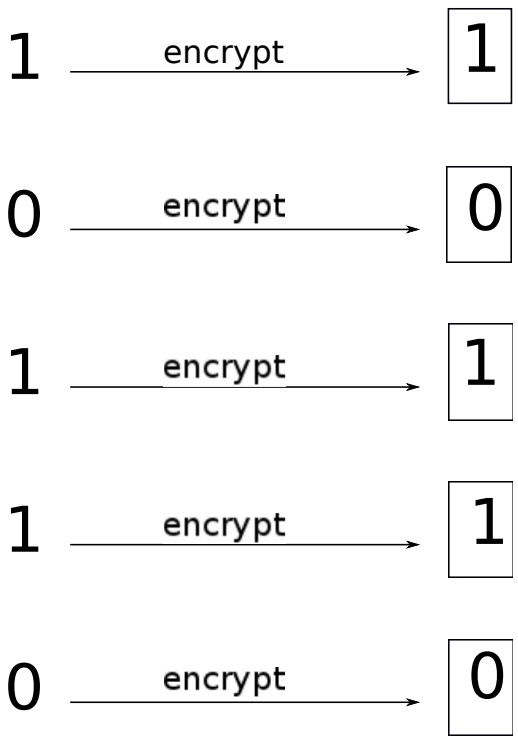
1

1

0

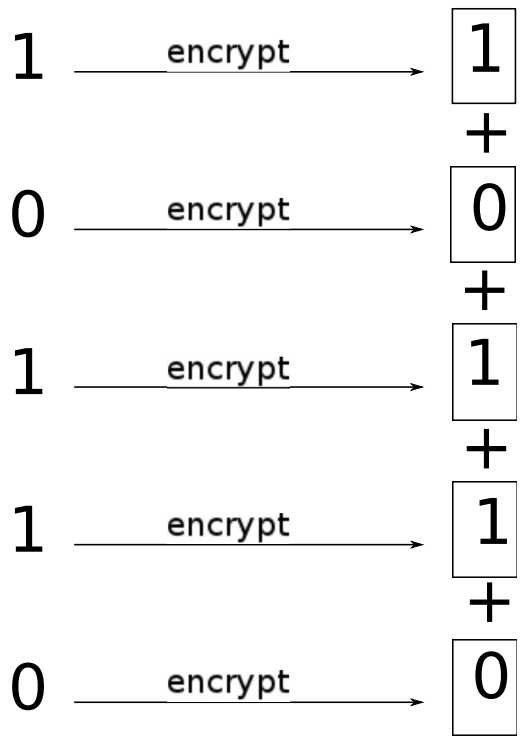
What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.



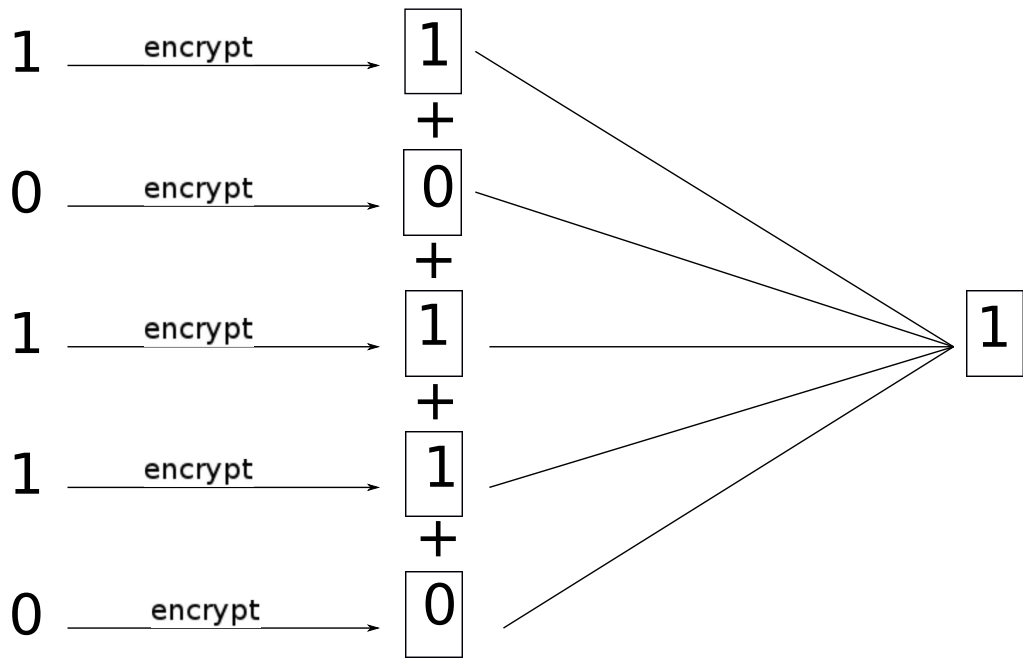
What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.



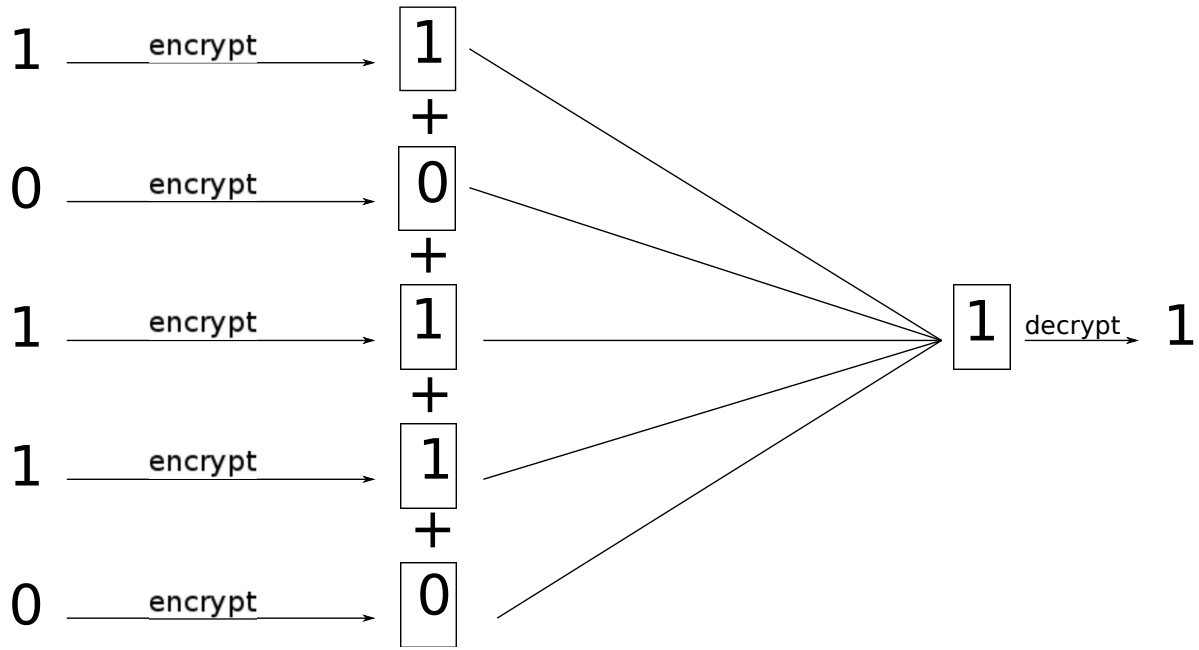
What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.



What is Homomorphic encryption?

It allows to perform operations on plain texts while they are encrypted.



Somewhat and Fully Homomorphic

Somewhat Homomorphic: can perform a limited number of operations.

Fully Homomorphic: can perform a virtually infinite number of operation.

Somewhat DGHV

We extend the Somewhat homomorphic scheme of Van Dick et al.[1] usualy called *DGHV*
In DGHV scheme the message is encrypted by hiding it in the additional noise of a multiple of an odd integer p.

[1]Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010) Fully homomorphic encryption over the integers. Advances in Cryptology - EUROCRYPT 2010, 24-43.

Somewhat DGHV

We extend the Somewhat homomorphic scheme of Van Dick et al.[1] usualy called *DGHV*
In DGHV scheme the message is encrypted by hiding it in the additional noise of a multiple of an odd integer p.

$$c = p \cdot q + 2r + m$$

[1]Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010) Fully homomorphic encryption over the integers. Advances in Cryptology - EUROCRYPT 2010, 24-43.

Somewhat DGHV

We extend the Somewhat homomorphic scheme of Van Dick et al.[1] usualy called *DGHV*
In DGHV scheme the message is encrypted by hiding it in the additional noise of a multiple of an odd integer p.

$$c = p \cdot q + 2r + m$$

Until $c \mod p < p/2$ the message can be recovered easily.

[1] Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010) Fully homomorphic encryption over the integers. Advances in Cryptology - EUROCRYPT 2010, 24-43.

Details

The previous symmetrical scheme can be modified into an asymmetric one.
Let:

- λ be an integer referred to as the security parameter;
- η be the bit length of the secret key p;
- τ be the number of elements composing the public key, each of which has bit length γ;
- ρ and ρ' be respectively the bit lengths of the noise in the public key and in a fresh ciphertext.

Details continue

Keys :

Secret Key : a η-bit random odd integer p.

Details continue

Keys :

Secret Key : a η-bit random odd integer p.

Public Key : τ integers x_i such that $x_i = p \cdot q_i + r_i$. for $i = 0, \ldots \tau$. x_0 is the largest, it is odd and $[x_0]_p$ is even.

Details continue

Keys :

Secret Key : a η-bit random odd integer p.

Public Key : τ integers x_i such that $x_i = p \cdot q_i + r_i$. for $i = 0, \ldots \tau$. x_0 is the largest, it is odd and $[x_0]_p$ is even.

encryption : given $m \in \{0, 1\}$

$$c = m + 2r + 2 \sum_{S \subset \{1, \ldots \tau\}} x_i \mod x_0$$

Details continue

Keys :

Secret Key : a η-bit random odd integer p.

Public Key : τ integers x_i such that $x_i = p \cdot q_i + r_i$. for $i = 0, \ldots \tau$. x_0 is the largest, it is odd and $[x_0]_p$ is even.

encryption : given $m \in \{0, 1\}$

$$c = m + 2r + 2 \sum_{S \subset \{1, \ldots \tau\}} x_i \mod x_0$$

decryption :$[c \mod p]_2$

Details continue

Keys :

Secret Key : a η-bit random odd integer p.

Public Key : τ integers x_i such that $x_i = p \cdot q_i + r_i$. for $i = 0, \ldots \tau$. x_0 is the largest, it is odd and $[x_0]_p$ is even.

encryption : given $m \in \{0, 1\}$

$$c = m + 2r + 2 \sum_{S \subset \{1, \ldots \tau\}} x_i \mod x_0$$

decryption :$[c \mod p]_2$

evaluate : Given $C(x_1, \ldots x_t)$and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts and return the resulting ciphertext.

Our Cryptosystem

* The schemes proposed thus far work on the integer ring $\mathbb{Z}_2$.
* We propose to work in the ring $\mathbb{Z}_b$ ($b \in \mathbb{Z}$)
* We choose an integer $b = 2^k$ ($k > 0$),
* We modify the original scheme so that it can encrypt and decrypt integer numbers in the interval $[0, b)$.

Our Cryptosystem

* The schemes proposed thus far work on the integer ring $\mathbb{Z}_2$.
* We propose to work in the ring $\mathbb{Z}_b$ ($b \in \mathbb{Z}$)
* We choose an integer $b = 2^k$ ($k > 0$),
* We modify the original scheme so that it can encrypt and decrypt integer numbers in the interval $[0, b)$.
*

$$c = m + p \cdot q + br$$

Negative Numbers

The scheme we are going to propose can also encode negative numbers. Given the base $b = 2^k \Rightarrow$ we can encrypt number in the interval $(-b/2, b/2]$. The decryption function is performed as:

$$([c]_p \mod b)$$

The result is:

positive if $\left[[c]_p\right]_b < b/2$

negative if $\left[[c]_p\right]_b > b/2 \Rightarrow \left[[c]_p\right]_b - b$

In this case the *base* should be twice the maximum integer that needs to be computed.

Keys

The Secret Key: is an odd η-bit integers $p \in \mathbb{Z}$ such that $p \equiv 1($ mod $b)$ in other words $p = n \cdot b + 1$.

Public key: Choose a random set of odd integers $q_i \in \mathbb{Z}$. For all $1 \leq i \leq \tau$ we define $x_i = q_i \cdot p + r_i$ and a noise-free element $x_0 = q_0 \cdot p$ with $q_0 > q_i$ for all i. The public key is $\mathbf{pk} = \{x_0, x_1, \ldots, x_\tau\}$.

Encryption and decryption

`Encrypt`(**pk**, m). Given a integer message $m \in [0, b)$, choose a random integer $r \in (-2^{\rho'}, 2^{\rho'})$ and let S be a sparse subset of indexes in $\{1, \ldots, \tau\}$. The ciphertext c is:

$$c = \left[m + br + b \sum_{i \in S} x_i \right]_{x_0} \tag{1}$$

`Decrypt`(p, c). Given a ciphertext c, the decryption function is:

$$m = [c]_p \mod b, \tag{2}$$

Operations

The integer x_0 of the public key is an exact multiple of p
This choice permits the reduction modulo x_0 after every operation

`Add`$(c_1, c_2, \mathbf{pk})$. To perform addition, given two cipher texts c_1, c_2:

$$c_s = [c_1 + c_2]_{x_0} \tag{3}$$

`Mult`$(c_1, c_2, \mathbf{pk})$ To perform multiplication, given two cipher texts c_1, c_2:

$$c_m = [c_1 \cdot c_2]_{x_0} \tag{4}$$

Parameters

The sets of parameters chosen to test the performance of the stand-alone scheme

Security level	λ	ρ	η	γ	τ
Toy	42	27	1026	150000	158
Small	52	41	1558	830000	572
Medium	62	56	2128	4200000	2110

Table : Parameter sets of the proposed scheme as a function of λ.

The sets of parameters used are the same as in Coron et al. paper[2]

[2]Coron, J. S., Mandal, A., Naccache, D., and Tibouchi, M. (2011). Fully homomorphic encryption over the integers with shorter public keys. Advances in Cryptology - CRYPTO 2011, 487-504.

Number of operations

Since the scheme is somewhat Homomorphic it is important to know how many operations we can perform.

$$\mu \leq \frac{\eta - 1}{\log_2 \tau + k + \rho + 2}. \tag{5}$$

base		lambda 42	lambda 52	lambda 62
2	**Practical average**	25.41	27.29	28.54
	theoretical	26.76	28.75	29.94
2048	**Practical average**	19.00	22.02	24.02
	theoretical	22.14	25.05	26.91
4096	**Practical average**	18.12	21.58	24.02
	theoretical	21.22	24.27	26.25

Table : Maximum number of multiplications. We compare theoretical limit with practical. They are shown in Table 1.

Iris

Among all the iris matching protocols proposed in the past, we rely on the one described by Daugman[3].

$$D(q, x_i) = \|(q \oplus x_i)\| < \epsilon$$

The result is a 2048 bit vector

[3] Daugman, J. (2004). How iris recognition works. Circuits and Systems for Video Technology, IEEE Transactions on, 14(1), 21 - 30.

Iris

Among all the iris matching protocols proposed in the past, we rely on the one described by Daugman[3].

$$D(q, x_i) = \|(q \oplus x_i)\| < \epsilon$$

$$D(q, x_i) < \epsilon$$

$$\Downarrow$$

$$D(q, x_i) - \epsilon < 0$$

The result is a 2048 bit vector

[3] Daugman, J. (2004). How iris recognition works. Circuits and Systems for Video Technology, IEEE Transactions on, 14(1), 21 - 30.

Protocol

The protocol can be divided into three parts:
We assume client have already provided Pk in a registration phase

One: The client sends to the server

- A binary probe $(q_1 \dots q_n)$

Protocol

The protocol can be divided into three parts:
We assume client have already provided Pk in a registration phase

One: The client sends to the server

- A binary probe $(q_1 \dots q_n)$

Two: The server

1. Encrypts all database iris vectors with pk
2. Starts computing
3. Sends back a integer vector

Protocol

The protocol can be divided into three parts:
We assume client have already provided Pk in a registration phase

One: The client sends to the server

- A binary probe $(q_1 \dots q_n)$

Two: The server

1. Encrypts all database iris vectors with pk
2. Starts computing
3. Sends back a integer vector

Three: The client, received the encrypted vector, decrypts it by using the secret key and checks if one of the integers would be negative.

Protocol

The protocol can be divided into three parts:
We assume client have already provided Pk in a registration phase

One: The client sends to the server

- A binary probe $(q_1 \dots q_n)$

Two: The server

1. Encrypts all database iris vectors with pk
2. Starts computing
3. Sends back a integer vector

Three: The client, received the encrypted vector, decrypts it by using the secret key and checks if one of the integers would be negative.

if there exists an index I in $\{1 \dots N\}$ for which $D(q, x_I) < \epsilon$
the user is enrolled in the database.

Server Computing

$$D(q, x_i) = \|(q \oplus x_i)\|$$

$\oplus$ is XOR between binary vectors

$\oplus$ is addition modulus b, in our case.

So the result could be either 0, 1, 2

dataBase (D)	1	0	1	1	1	0	0
probe (P)	0	1	1	0	1	0	0

Server Computing

$$D(q, x_i) = \|(q \oplus x_i)\|$$

$\oplus$ is XOR between binary vectors

$\oplus$ is addition modulus b, in our case.

So the result could be either 0, 1, 2

dataBase (D)	1	0	1	1	1	0	0
probe (P)	0	1	1	0	1	0	0
$D + P = A$	1	1	2	1	2	0	0

Server Computing

$$D(q, x_i) = \|(q \oplus x_i)\|$$

In order to transform 2's in zeros, we after perform $(q \cdot x_i)$ where $\cdot$ stands for multiplication.

dataBase (D)	1	0	1	1	1	0	0
probe (P)	0	1	1	0	1	0	0
$D + P = A$	1	1	2	1	2	0	0
$D \cdot P = B$	0	0	1	0	1	0	0

Server Computing

$$D(q, x_i) = \|(q \oplus x_i)\|$$

In order to transform 2's in zeros, we after perform $(q \cdot x_i)$ where $\cdot$ stands for multiplication.

dataBase (D)	1	0	1	1	1	0	0
probe (P)	0	1	1	0	1	0	0
$D + P = A$	1	1	2	1	2	0	0
$D \cdot P = B$	0	0	1	0	1	0	0
$2 \cdot B$	0	0	2	0	2	0	0

Server Computing

$$D(q, x_i) = \|(q \oplus x_i)\|$$

Finally we compute

$$D(q, x_i) = \|(q + x_i) - 2 \cdot (q \cdot x_i)\| = \|q \cdot (1 - 2x_i) + x_i\| .$$

dataBase (D)	1	0	1	1	1	0	0
probe (P)	0	1	1	0	1	0	0
$D + P = A$	1	1	2	1	2	0	0
$D \cdot P = B$	0	0	1	0	1	0	0
$2 \cdot B$	0	0	2	0	2	0	0
$A - 2 \cdot B$	1	1	0	1	0	0	0

Server Computing

So in the end we compute

$$D(q, x_i) = \|(q + x_i) - 2 \cdot (q \cdot x_i)\| = \|q \cdot (1 - 2x_i) + x_i\| .$$

$$\mathcal{E}_{\mathsf{pk}}[D(q, x_i)] = \|\mathcal{E}_{\mathsf{pk}}[q] \cdot \mathcal{E}_{\mathsf{pk}}[1 - 2x_i] + \mathcal{E}_{\mathsf{pk}}[x_i]\|$$

Server Computing Continue

$$\mathcal{E}_{pk}[D(q, x_i) - \epsilon] = \mathcal{E}_{pk}[D(q, x_i)] + \mathcal{E}_{pk}[-\epsilon] = \mathcal{E}_{pk}[d_i].$$

Server Computing Continue

$$\mathcal{E}_{pk}[D(q, x_i) - \epsilon] = \mathcal{E}_{pk}[D(q, x_i)] + \mathcal{E}_{pk}[-\epsilon] = \mathcal{E}_{pk}[d_i].$$

The number $\mathcal{E}_{pk}[d_i]$ is multiplied by a random number in $(0, 2^{k-1}]$

Server Computing Continue

$$\mathcal{E}_{pk}[D(q, x_i) - \epsilon] = \mathcal{E}_{pk}[D(q, x_i)] + \mathcal{E}_{pk}[-\epsilon] = \mathcal{E}_{pk}[d_i].$$

The number $\mathcal{E}_{pk}[d_i]$ is multiplied by a random number in $(0, 2^{k-1}]$
The resulting vector is randomly permutated

Communication Complexity - Cipher size.

λ	**Secret Key (KB)**	**Public Key (MB)**	**Ciphertext (KB)**
42.00	0.13	2.83	18.31
52.00	0.19	56.60	101.32
62.00	0.26	1056.43	512.70

Table : Size of the cipthertext and of the public and secret keys.

- PK varies from 3MB to 1GB. We assume Client already provided it.
- Each Ciphertext varies from 18KB to 513KB
- Each query sent to server needs 6.59 MB, 202.44MB and 1GB
- The sever sends back a ciphertext of the same size for each element of the data base.

Validation - Crypto System

Base	λ	Cif	Dec	PubKey	PrivKey	Add	Mult
2	42	0.37	0.75	133.73	154.44	0.00	88.75
	52	6.43	4.05	665.94	3662.76	0.25	2653.98
	62	97.77	2.40	1883.80	65784.24	0.50	68518.53
2048	42	0.12	0.75	134.82	139.87	0.12	85.18
	52	5.28	3.70	564.36	33192.42	0.18	2588.43
	62	94.37	28.56	1856.56	83778.98	0.88	66570.42
4096	42	0.50	0.37	141.94	104.17	0.06	86.48
	52	5.88	4.29	580.28	4204.64	0.06	2575.74
	62	90.46	27.54	1828.22	62273.04	0.62	67112.76

Table : Average execution time (ms) of the proposed scheme as a function of the *base* b and of the system parameters.

Validation - Protocol

lambda	Client(s) part 1	Server (s) part 2	Client (s) part 3
42.00	0.062	22.288	0.00042
52.00	0.377	68.422	0.00260
62.00	2.317	17034.959	0.02263

Table : Average time (s) of execution for each part of the proposed protocol. Times in parts 2 and 3 are relative to a single iris template in the database.

Future Works

- Security demonstration
- Parallelize server computing
- Change of bases