SHE based Non Interactive Privacy Preserving Biometric Authentication Protocols

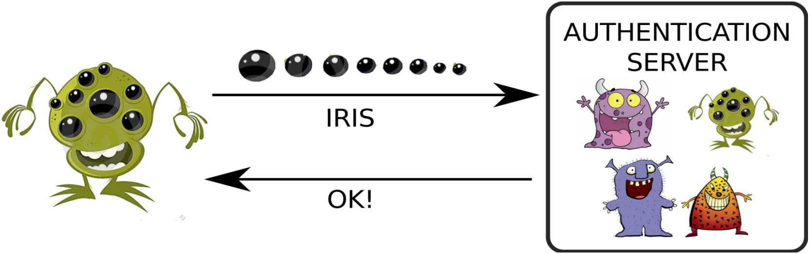
Giulia Droandi, Riccardo Lazzeretti

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

9th IEEE International Symposium on Intelligent Signal Processing - WISP 2015

Motivation

Biometric signals often used in access control systems.



Privacy Preserving Protocols

Need of data protection for everyone involved in the computation.

Examples: a database and a client, cloud computing.

Some solutions: Homomorphic Encryption, Garbled Circuit . . .
Interactive protocols.

Privacy Preserving Protocols

Need of data protection for everyone involved in the computation.

Examples: a database and a client, cloud computing.

Some solutions: Homomorphic Encryption, Garbled Circuit . . .
Interactive protocols.

Our Solution: **Somewhat Homomorphic Encryption**. Non interactive protocol.

What is Somewhat Homomorphic Encryption?

* A cryptographic scheme is called *homomorphic* if it allows to perform one or more operations on plain texts while they are encrypted.

What is Somewhat Homomorphic Encryption?

* A cryptographic scheme is called *homomorphic* if it allows to perform one or more operations on plain texts while they are encrypted.
* Classic homomorphic encryption (HE), are only additively or multiplicatively homomorphic.

What is Somewhat Homomorphic Encryption?

* A cryptographic scheme is called *homomorphic* if it allows to perform one or more operations on plain texts while they are encrypted.
* Classic homomorphic encryption (HE), are only additively or multiplicatively homomorphic.
* The schemes homomorphic for both addition and multiplication, are called:

Somewhat Homomorphic (SHE): it can perform a limited number of operations.

Fully Homomorphic (FHE): it can perform a virtually infinite number of operations.

The first FHE scheme was proposed by Gentry in 2009. From his intuition many others schemes followed.

Pisa et al. Cryptosystem

* Usually a SHE scheme works on the integer ring $\mathbb{Z}_2$.
* Pisa et al. propose a SHE scheme working in the ring $\mathbb{Z}_b$ ($b \in \mathbb{Z}$).

Pisa et al. Cryptosystem

* Usually a SHE scheme works on the integer ring $\mathbb{Z}_2$.
* Pisa et al. propose a SHE scheme working in the ring $\mathbb{Z}_b$ ($b \in \mathbb{Z}$).
* Given an integer base $b = 2^k$ ($k > 0$), a secret odd integer p, and two random integers r, q, a message m can be encrypted as:

$$c = [\![m]\!] = m + br + p \cdot q$$

* To decrypt:

$$\left[[c]_p \right]_b$$

Pisa et al. Cryptosystem

* Usually a SHE scheme works on the integer ring $\mathbb{Z}_2$.
* Pisa et al. propose a SHE scheme working in the ring $\mathbb{Z}_b$ $(b \in \mathbb{Z})$.
* Given an integer base $b = 2^k$ $(k > 0)$, a secret odd integer p, and two random integers r, q, a message m can be encrypted as:

$$c = [\![m]\!] = m + br + p \cdot q$$

* To decrypt:

$$\left[[c]_p \right]_b$$

* We extend Pisa et al. scheme to encrypt negative integers.

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.

Cryptosystem

* The scheme described so far can be modified to be asymmetric, it has a public key (a set of particular integers) for encryption and computation and a secret key (p) for decryption.

Cryptosystem

* The scheme described so far can be modified to be asymmetric, it has a public key (a set of particular integers) for encryption and computation and a secret key (p) for decryption.
* Addition and multiplication can be performed simply adding or multiplying two ciphertexts.

Cryptosystem

* The scheme described so far can be modified to be asymmetric, it has a public key (a set of particular integers) for encryption and computation and a secret key (p) for decryption.
* Addition and multiplication can be performed simply adding or multiplying two ciphertexts.
* The message is hidden in the noise: $m + br$.
* After every operation the noise increases. Until $|m + br| < \frac{p}{2}$ the message can be recovered easily.

Cryptosystem

* The scheme described so far can be modified to be asymmetric, it has a public key (a set of particular integers) for encryption and computation and a secret key (p) for decryption.
* Addition and multiplication can be performed simply adding or multiplying two ciphertexts.
* The message is hidden in the noise: $m + br$.
* After every operation the noise increases. Until $|m + br| < \frac{p}{2}$ the message can be recovered easily.
* Multiplication: The noise increases as $\mathcal{O}((br)^2)$, after few multiplications decryption becomes impossible.

Cryptosystem - Implementation

* The number of possible operations depends on the base and on a security parameter λ.
* Public key size grows with both λ and base b.
* For $\lambda = 20$, public key is too big (about 11GB for $b = 2^{50}$) to be used in practice.
* Multiplication is an expensive time-consuming operation.

Application of the Cryptosystem: Authentication Protocol

We assume client has already provided the public key and an encrypted feature vector $(s_1 \ldots s_n)$ in a registration phase.

Step 1: The client sends a probe $(q_1 \ldots q_n)$ to the server.

Application of the Cryptosystem: Authentication Protocol

We assume client has already provided the public key and an encrypted feature vector $(s_1 \dots s_n)$ in a registration phase.

Step 1: The client sends a probe $(q_1 \dots q_n)$ to the server.

Step 2: The server

1. computes the distance between the probe $(q_1 \dots q_n)$ and the stored feature $(s_1 \dots s_n)$.
2. compares the distance with the threshold.
3. blinds the result.

Application of the Cryptosystem: Authentication Protocol

We assume client has already provided the public key and an encrypted feature vector $(s_1 \dots s_n)$ in a registration phase.

Step 1: The client sends a probe $(q_1 \dots q_n)$ to the server.

Step 2: The server

1. computes the distance between the probe $(q_1 \dots q_n)$ and the stored feature $(s_1 \dots s_n)$.
2. compares the distance with the threshold.
3. blinds the result.

Step 3: The client decrypts the received result by using the secret key and checks if it is positive or negative.

Server Computation: Distances

Hamming. Usual: $HD(\mathbf{q},\mathbf{s}) = \|(\mathbf{q} \otimes \mathbf{s})\|$.
With Integers: $HD(\mathbf{q},\mathbf{s}) = \sum_{i=0}^{n} s_i + q_i(1-2s_i)$.
Encrypted:
$[\![HD(q,s)]\!] = \sum_{i=0}^{n} \left([\![s_i]\!] + [\![q_i]\!] \cdot [\![1-2s_i]\!]\right)$.

Server Computation: Distances

Hamming. Usual: $HD(\mathbf{q}, \mathbf{s}) = \|(\mathbf{q} \otimes \mathbf{s})\|$.
With Integers: $HD(\mathbf{q}, \mathbf{s}) = \sum_{i=0}^{n} s_i + q_i(1 - 2s_i)$.
Encrypted:
$[\![HD(q, s)]\!] = \sum_{i=0}^{n} \left([\![s_i]\!] + [\![q_i]\!] \cdot [\![1 - 2s_i]\!]\right)$.

Euclidean. Usual: $ED(\mathbf{q}, \mathbf{s}) = \left\|(\mathbf{q} - \mathbf{s})^2\right\|$
Encrypted: $[\![ED(q, s)]\!] = \sum_{i=0}^{n} \left([\![q_i]\!] + [\![-s_i]\!]\right)^2$.

Server Computation: Thresholding

The distance $D(\mathbf{q}, \mathbf{s})$ is compared with the threshold ϵ, by computing the difference between the distance and the threshold under encryption, i.e.

$$D(\mathbf{q}, \mathbf{s}) < \epsilon$$

$$\Downarrow$$

$$D(\mathbf{q}, \mathbf{s}) - \epsilon < 0$$

Server Computation: Thresholding

The distance $D(\mathbf{q}, \mathbf{s})$ is compared with the threshold ϵ, by computing the difference between the distance and the threshold under encryption, i.e.

$$D(\mathbf{q}, \mathbf{s}) < \epsilon$$

$$\Downarrow$$

$$D(\mathbf{q}, \mathbf{s}) - \epsilon < 0$$

$$[\![d]\!] = [\![D(\mathbf{q}, \mathbf{s}) - \epsilon]\!] = [\![D(\mathbf{q}, \mathbf{s})]\!] + [\![-\epsilon]\!]$$

Server Computation: Blinding

* The number $[\![d]\!]$ has to be blinded, before being disclosed to the client.

Server Computation: Blinding

* The number $[\![d]\!]$ has to be blinded, before being disclosed to the client.
* Additive blinding cannot be used because it could change the sign of $[\![d]\!]$.
* Multiplicative blinding is much less secure.

Server Computation: Blinding

* The number $[\![d]\!]$ has to be blinded, before being disclosed to the client.
* Additive blinding cannot be used because it could change the sign of $[\![d]\!]$.
* Multiplicative blinding is much less secure.
* We adopt a hybrid multiplicative/additive blinding:

$$[\![k_1(D(\mathbf{q},\mathbf{s})-\epsilon)+k_2]\!] = [\![k_1]\!][\![D(\mathbf{q},\mathbf{s})-\epsilon]\!]+[\![k_2]\!]$$

where k_1, k_2 are randomly chosen in a way to preserve the sign.

First Example: Iriscode Protocol

* Find pupil center and iris boundaries.
* Each iris local region is demodulated to extract its phase information using 2-D Gabor filters.
* The resulting complex coefficients are used to set two bits for the iriscode vector.
* The process is repeated along all the iris with many filters configurations: sizes, orientations and frequencies.

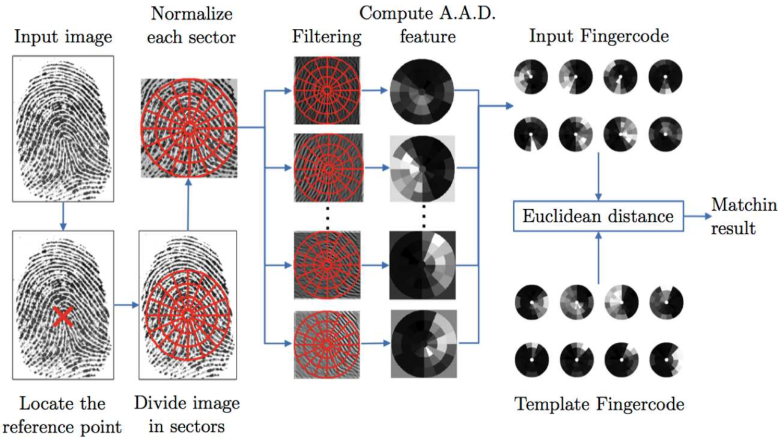
First Example: Iriscode Protocol

* An iris image can be encoded as a 2048 bit vector called iriscode.
* Maximum Hamming distance can be represented by 11 bits.

λ	Base	Step 1 Client Enc.	Step 2 Server Comp.	Step 3 Client Dec.
10	2^{50}	1.2 s	2.0 s	0.2 ms
	2^{100}	1.2 s	5.3 s	4.3 ms
	2^{150}	1.8 s	9.7 s	5.6 ms
15	2^{50}	29s	14 min 33 s	0.2 s

Table: Iriscode protocol's average execution time

Second Example: Fingercode Protocol



Second Example: Fingercode Protocol

* A fingerprint can be represented by a vector of 96 features of 2 bits each, called fingercode.
* The maximum distance value can be represented with 10 bits.

λ	Base	Step 1 Client Enc.	Step 2 Server Comp.	Step 3 Client Dec.
10	2^{50}	0.03 s	0.10 s	1 ms
	2^{100}	0.06 s	0.30 s	3 ms
	2^{150}	0.09 s	0.53 s	5 ms
15	2^{50}	1.37 s	37 s	0.18 s
	2^{100}	2.80 s	1min 48 s	0.45 s
	2^{150}	4.25 s	3 min 17 s	0.79 s

Table: Fingerprint protocol's average execution time

Communication Complexity

	λ	Base		
		2^{50}	2^{100}	2^{150}
Iris	10	13 MB	25 MB	38 MB
	15	750 MB	1.4GB	2 GB
Finger	10	606 KB	1.2 MB	1.7 MB
	15	34 MB	68 MB	102 MB

Table: Iriscode and fingercode communication complexity.

* Iriscode protocol has higher computational complexity than fingercode protocol.
* During computation, three vectors should be memorize at the same time. That needs about 6 GB in base 2^{150}.

Conclusions

* We presented a new solution for privacy preserving biometric authentication protocols: SHE.

Conclusions

* We presented a new solution for privacy preserving biometric authentication protocols: SHE.
* Novelties: SHE for negatives numbers; blinding.
* The runtimes needed by the SHE implementation are by far larger than the execution time of protocols based on Paillier HE or GC (about 1s - interactive protocols).

Conclusions

* We presented a new solution for privacy preserving biometric authentication protocols: SHE.
* Novelties: SHE for negatives numbers; blinding.
* The runtimes needed by the SHE implementation are by far larger than the execution time of protocols based on Paillier HE or GC (about 1s - interactive protocols).
* In our protocol all the computation is completely moved onto the server side and no interaction is needed.
* Runtimes can be lowered by using powerful servers allowing for parallelization across more threads.

Thank you for your attention.