A New Generalisation of the Goldwasser-Micali Cryptosystem Based on the Gap 2^k-Residuosity Assumption

Diana Maimuţ[1] and George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
{diana.maimut,tgeorge}@dcti.ro
[2] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania

Abstract. We present a novel public key encryption scheme that enables users to exchange many bits messages by means of *at least* two large prime numbers in a Goldwasser-Micali manner. Our cryptosystem is in fact a generalization of the Joye-Libert scheme (being itself an abstraction of the first probabilistic encryption scheme). We prove the security of the proposed cryptosystem in the standard model (based on the gap 2^k-residuosity assumption) and report complexity related facts. We also describe an application of our scheme to biometric authentication and discuss the security of our suggested protocol. Last but not least, we indicate several promising research directions.

1 Introduction

The authors of [11] introduced a public key encryption (PKE) scheme[3] representing a rather natural extension of the Goldwasser-Micali (GM) [9,10] cryptosystem, the first probabilistic encryption scheme. The Goldwasser-Micali cryptosystem achieves ciphertext indistinguishability under the *Quadratic Residuosity* (QR) assumption. Despite being simple and stylish, this scheme is quite uneconomical in terms of bandwidth[4]. Various attempts of generalizing the Goldwasser-Micali scheme were proposed in the literature in order to address the previously mentioned issue. The Joye-Libert (JL) scheme can be considered a follow-up of the cryptosystems proposed in [13] and [7] and efficiently supports the encryption of larger messages.

Inspired by the Joye-Libert scheme, we propose a new public key cryptosystem, analyze its security and provide the reader with an implementation and performance discussion. We construct our scheme based on 2^k-th power residue symbols. Our generalization of the Joye-Libert cryptosystem makes use of two important parameters when it comes to the encryption and decryption functions:

[3] reconsidered in [5]

[4] $k \cdot \log_2 n$ bits are needed to encrypt a k-bit message, where n is an RSA modulus as described in [9, 10]

the number of bits of a message and the number of distinct primes of a public modulus n. Thus, our proposal not only supports the encryption of larger messages (as in the Joye-Libert variant), but also operates on *a variable number of large primes* (instead of two in the Joye-Libert case). Both these parameters can be chosen depending on the desired security application.

Our scheme can be viewed as a flexible solution characterized by the ability of making adequate trade-offs between encryption speed and ciphertext expansion in a given context.

In biometric authentication protocols, when a user identifies himself using his biometric characteristics (captured by a sensor), the collected data will vary. Thus, traditional cryptographic approaches (such as storing a hash value) are not suitable in this case, since they are not error tolerant. As a result, biometric-based protocols must be constructed in a special way and, moreover, the system must protect the sensitivity and privacy of a user's biometric characteristics. Such a protocol is proposed in [6]. Its core is the Goldwasser-Micali encryption scheme. Thus, a natural extension of the protocol in [6] can be obtained using our generalization of the Joye-Libert scheme. Thus, we describe such a biometric authentication protocol and discuss its security.

Structure of the paper. In Section 2 we introduce notations, definitions, security assumptions and schemes used throughout the paper. Inspired by the Joye-Libert PKE scheme and aiming at obtaining a relevant generalization, in Section 3 we propose a new scheme based on 2^k residues, prove it secure in the standard model and analyze its performance compared to other related cryptosystems. An application of our scheme to biometric authentication and its security analysis are presented in Section 4. We conclude in Section 5 and in Appendix A we present some optimized decryption algorithms for our proposed scheme.

2 Preliminaries

Notations. Throughout the paper, λ denotes a security parameter. We use the notation $x \xleftarrow{\$} X$ when selecting a random element x from a sample space X. We denote by $x \leftarrow y$ the assignment of the value y to the variable x. The probability that event E happens is denoted by $Pr[E]$. The Jacobi symbol of an integer a modulo an integer n is represented by $\left(\frac{a}{n}\right)$. J_n and $\bar{J}_n$ denote the sets of integers modulo n with Jacobi symbol 1, respectively -1. Throughout the paper, we let QR_n be the set of quadratic residues modulo n. We consider as $\mathcal{Z}_p = \{-(p-1)/2, \ldots, -1, 0, 1, \ldots, (p-1)/2\}$ the alternative representation modulo an integer p. The set of integers $\{0, \ldots, a-1\}$ is further denoted by $[0, a)$. Multidimensional vectors $v = (v_0, \ldots, v_{s-1})$ are represented as $v = \{v_i\}_{i \in [0,s)}$.

2.1 2^k-th power residue

In this paper, we consider the 2^k-th power residue symbol as presented in [15]. The classical Legendre symbol is obtained when $k = 1$.

Definition 1. *Let p be an odd prime such that $2^k | p-1$. Then the symbol*

$$\left(\frac{a}{p}\right)_{2^k} = a^{\frac{p-1}{2^k}} \bmod p$$

is called the 2^k-th power residue symbol modulo p, where $a^{\frac{p-1}{2^k}} \in \mathcal{Z}_p$.

Properties. The 2^k-th power residue symbol satisfies the following properties

1. If $a \equiv b \bmod p$, then $\left(\frac{a}{p}\right)_{2^k} = \left(\frac{b}{p}\right)_{2^k}$
2. $\left(\frac{a^{2^k}}{p}\right)_{2^k} = 1$
3. $\left(\frac{ab}{p}\right)_{2^k} = \left(\frac{a}{p}\right)_{2^k}\left(\frac{b}{p}\right)_{2^k} \bmod p$
4. $\left(\frac{1}{p}\right)_{2^k} = 1$ and $\left(\frac{-1}{p}\right)_{2^k} = (-1)^{(p-1)/2^k}$

2.2 Computational Complexity

In our performance analysis we use the complexities of the mathematical operations listed in Table 1. These complexities are in accordance with the algorithms presented in [8]. We do not use the explicit complexity of multiplication, but instead we refer to it as $M(\cdot)$ for clarity.

Table 1. Computational complexity for μ-bit numbers and k-bit exponents

Operation	Complexity
Multiplication	$M(\mu) = \mathcal{O}(\mu \log(\mu) \log(\log(\mu)))$
Exponentiation	$\mathcal{O}(kM(\mu))$
Jacobi symbol	$\mathcal{O}(\log(\mu)M(\mu))$

2.3 Security Assumptions

Definition 2 (Quadratic Residuosity - QR, Squared Jacobi Symbol - SJS and Gap 2^k-Residuosity - GR). *Choose two large prime numbers $p, q \geq 2^\lambda$ and compute $n = pq$. Let A be a probabilistic polynomial-time (PPT) algorithm that returns 1 on input (x, n) or (x^2, n) or (x, k, n) if $x \in QR_n$ or J_n or $J_n \setminus QR_n$. We define the advantages*

$$ADV_A^{\text{QR}}(\lambda) = \left| Pr[A(x,n) = 1 | x \xleftarrow{\$} QR_n] - Pr[A(x,n) = 1 | x \xleftarrow{\$} J_n \setminus QR_n] \right|,$$

$$ADV_A^{\text{SJS}}(\lambda) = \left| Pr[A(x^2,n) = 1 | x \xleftarrow{\$} J_n] - Pr[A(x^2,n) = 1 | x \xleftarrow{\$} \bar{J}_n] \right|,$$

$$ADV_{A,k}^{\text{GR}}(\lambda) = \left| Pr[A(x,k,n) = 1 | x \xleftarrow{\$} J_n \setminus QR_n] - Pr[A(x^{2^k},k,n) = 1 | x \xleftarrow{\$} \mathbb{Z}_n^*] \right|.$$

The Quadratic Residuosity assumption states that for any PPT algorithm A the advantage $ADV_A^{\mathrm{QR}}(\lambda)$ is negligible.

If $p, q \equiv 1 \bmod 4$, then the Squared Jacobi Symbol assumption states that for any PPT algorithm A the advantage $ADV_A^{\mathrm{SJS}}(\lambda)$ is negligible.

Let $p, q \equiv 1 \bmod 2^k$. The Gap 2^k-Residuosity assumption states that for any PPT algorithm A the advantage $ADV_A^{\mathrm{GR}}(\lambda)$ is negligible.

Remark 1. In [5], the authors investigate the relation between the assumptions presented in Definition 2. They prove that for any PPT adversary A against the GR assumption, we have two efficient PPT algorithms B_1 and B_2 such that

$$ADV_{A,k}^{\mathrm{GR}}(\lambda) \leq \frac{3}{2}\left((k-\frac{1}{3}) \cdot ADV_{B_1}^{\mathrm{QR}}(\lambda) + (k-1) \cdot ADV_{B_2}^{\mathrm{SJS}}(\lambda)\right).$$

2.4 Public Key Encryption

A *public key encryption* (PKE) scheme usually consists of three PPT algorithms: *Setup*, *Encrypt* and *Decrypt*. The *Setup* algorithm takes as input a security parameter and outputs the public key as well as the matching secret key. *Encrypt* takes as input the public key and a message and outputs the corresponding ciphertext. The *Decrypt* algorithm takes as input the secret key and a ciphertext and outputs either a valid message or an invalidity symbol (if the decryption failed).

Definition 3 (Indistinguishability under Chosen Plaintext Attacks - IND-CPA). *The security model against chosen plaintext attacks for a PKE scheme is captured in the following game:*

- *Setup(λ): The challenger C generates the public key, sends it to adversary A and keeps the matching secret key to himself.*
- *Query: Adversary A sends to C two equal length messages m_0, m_1. The challenger flips a coin $b \in \{0,1\}$ and encrypts m_b. The resulting ciphertext c is sent to the adversary.*
- *Guess: In this phase, the adversary outputs a guess $b' \in \{0,1\}$. He wins the game, if $b' = b$.*

The advantage of an adversary A attacking a PKE scheme is defined as

$$ADV_A^{\text{IND-CPA}}(\lambda) = |Pr[b = b'] - 1/2|$$

where the probability is computed over the random bits used by C and A. A PKE scheme is IND-CPA *secure, if for any PPT adversary A the advantage $ADV_A^{\text{IND-CPA}}(\lambda)$ is negligible.*

The Joye-Libert PKE scheme The Joye-Libert scheme was introduced in [11] and reconsidered in [5]. The scheme is proven secure in the standard model under the GR assumption. We shortly describe the algorithms of the Joye-Libert cryptosystem.

Setup(λ): Set an integer $k \geq 1$. Randomly generate two distinct large prime numbers p, q such that $p, q \geq 2^{\lambda}$ and $p, q \equiv 1 \bmod 2^k$. Output the public key $pk = (n, y, k)$, where $n = pq$ and $y \in J_n \setminus QR_n$. The corresponding secret key is $sk = (p, q)$.

Encrypt(pk, m): To encrypt a message $m \in [0, 2^k)$, we choose $x \xleftarrow{\$} \mathbb{Z}_n^*$ and compute $c \equiv y^m x^{2^k} \bmod n$. Output the ciphertext c.

Decrypt(sk, c): Compute $z \equiv \left(\frac{c}{p}\right)_{2^k}$ and find m such that the relation $\left[\left(\frac{y}{p}\right)_{2^k}\right]^m \equiv z \bmod p$ holds. Efficient methods to recover m can be found in [12].

2.5 A Security Model for Biometric Authentication

We further consider the security model for biometric authentication described in [3] in accordance with the terminology established in [6]. We stress that the authors of [6] preferred a rather informal way of presenting their security model while the approach of [3] is formal.

Participants and Roles. The data flow between the different roles assumed in the authentication protocol of [3] is depicted in Figure 1.

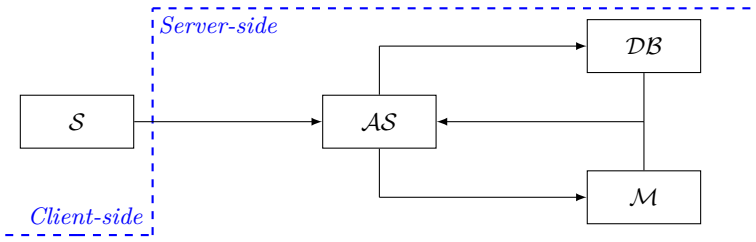

Fig. 1. Data flow and roles

The *server-side* functionality consists of three components to ensure that no single entity can associate a user's identity with the biometric data being collected during authentication. The roles assumed in the authentication protocol are:

– The *Sensor* ($\mathcal{S}$) represents the *client-side* component. As in [6], we assume that the sensor is capable of capturing the user's biometric data, extracting

it into a binary string[5], and performing cryptographic operations such as PKE. We also assume a *liveness link* between the sensor and the server-side components, to provide confidence that the biometric data received on the *server-side* is from a present living person.
- The *Authentication Server* ($\mathcal{AS}$) is responsible for communicating with the user who wants to authenticate and organizing the entire *server-side* procedure. In a successful authentication the AS obviously learns the user's identity, meaning that it should learn nothing about the biometric data being submitted.
- The *Database* ($\mathcal{DB}$) securely stores the users' profile and its job is to execute the pre-decision part of classification. Since the $\mathcal{DB}$ is aware of privileged biometric data, it should learn nothing about the user's identity, or even be able to correlate or trace authentication runs from a given (unknown) user.
- The *Matcher* ($\mathcal{M}$) completes the authentication process by taking the output produced by the $\mathcal{DB}$ server and computing the final decision step. This implies that the $\mathcal{M}$ possesses privileged information that allows it to make a final decision, and again that it should not be able to learn anything about the user's real identity, or even be able to correlate or trace authentication runs from a given (unknown) user.

Definition 4. *Let* $v = \{v_i\}_{i\in[0,s)}$ *and* $w = \{w_i\}_{i\in[0,s)}$ *be two* s*-dimensional vectors. Then the taxicab distance is defined as* $\mathcal{T}(v, w) = \sum_{i=0}^{s-1} |v_i - w_i|$*. The taxicab norm is defined as* $\mathcal{T}(v, 0)$*.*

The first step in having a useful authentication protocol is for it to be sound. This requirement is formalized in Requirement 1. Requirements 2. and 3. are concerned with the sensitive[6] relation between a user's identity and its biometric characteristics. We want to guarantee that the only entity in the infrastructure that knows information about this relation is the sensor.

Requirement 1. The matcher $\mathcal{M}$ can compute the taxicab distance $\mathcal{T}(b_i, b'_i)$, where b_i is the reference biometric template and b'_i is the fresh biometric template sent in the authentication request. Therefore, $\mathcal{M}$ can compare the distance to a given threshold value d and the server $\mathcal{AS}$ can make the right decision.

Requirement 2. For any identity ID_{i_0}, two biometric templates b'_{i_0}, b'_{i_1}, where $i_0, i_1 \geq 1$ and b'_{i_0} is the biometric template related to ID_{i_0}, it is infeasible for any of $\mathcal{M}$, $\mathcal{DB}$ and $\mathcal{AS}$ to distinguish between (ID_{i_0}, b'_{i_0}) and (ID_{i_0}, b'_{i_1}).

Requirement 3. For any two users U_{i_0} and U_{i_1}, where $i_0, i_1 \geq 1$, if U_{i_β}, where $\beta \xleftarrow{\$} \{0, 1\}$ makes an authentication attempt, then the database $\mathcal{DB}$ can only guess β with a negligible advantage. Suppose the database $\mathcal{DB}$ makes a guess β', the advantage is $|Pr[b = b'] - 1/2|$.

[5] We further consider the binary string as a vector of fixed length blocks.
[6] in terms of the system's security

3 A New Public Key Encryption Scheme

Inspired by the Joye-Libert scheme and wishing to obtain a meaningful generalization, we propose a new public key cryptosystem in Section 3.1 and analyze its security in Section 3.2. An implementation and performance analysis is provided in Section 3.3.

3.1 Description

Setup(λ): Set an integer $k \geq 1$. Randomly generate $\gamma + 1$ distinct large prime numbers $p_i, i \in [0, \gamma + 1)$ such that $p_i \geq 2^\lambda$ and $p_i \equiv 1 \bmod 2^k$. Let $n = p_0 \cdot \ldots \cdot p_\gamma$. Select $y_i \xleftarrow{\$} \mathbb{Z}_n^*, i \in [0, \gamma)$, such that the following conditions hold

1. $\left(\frac{y_i}{p_i}\right) = -1$
2. $\left(\frac{y_i}{p_\gamma}\right) = -1$
3. $\left(\frac{y_i}{p_j}\right)_{2^k} = 1$, where $j \neq i$

We denote by $y = \{y_i\}_{i \in [0,\gamma)}$ and $p = \{p_i\}_{i \in [0,\gamma)}$. Output the public key $pk = (n, y, k)$. The secret key is $sk = p$.

Encrypt(pk, m): To encrypt message $m \in [0, 2^k\gamma)$, first we divide it into γ blocks $m = m_0 \| \ldots \| m_{\gamma-1}$. Then, we choose $x \xleftarrow{\$} \mathbb{Z}_n^*$ and compute $c \equiv x^{2^k} \cdot \prod_{i=0}^{\gamma-1} y_i^{m_i} \bmod n$. The output is ciphertext c.

Decrypt(sk, c): For each $i \in [0, \gamma)$, compute $m_i = Dec_{p_i}(p_i, y_i, c)$.

Algorithm 1: $Dec_{p_i}(p_i, y_i, c)$

Input: The secret prime p_i, the value y_i and the ciphertext c
Output: The message block m_i

$m_i \leftarrow 0$, $B \leftarrow 1$
foreach $s \in [1, k + 1)$ **do**
  $z \leftarrow \left(\frac{c}{p_i}\right)_{2^s}$
  $t \leftarrow \left(\frac{y_i}{p_i}\right)_{2^s}$
  $t \leftarrow t^{m_i} \bmod p_i$
  if $t \neq z$ **then**
    $m_i \leftarrow m_i + B$
  end
  $B \leftarrow 2B$
end
return m_i

Correctness. Let $m_i = \sum_{w=0}^{k-1} b_w 2^w$ be the binary expansion of block m_i. Note that

$$\left(\frac{c}{p_i}\right)_{2^s} = \left(\frac{x^{2^k} \cdot \prod_{v=0}^{\gamma-1} y_v^{m_v}}{p_i}\right)_{2^s} = \left(\frac{y_i^{m_i}}{p_i}\right)_{2^s} = \left(\frac{y_i}{p_i}\right)_{2^s}^{\sum_{w=0}^{s-1} b_w 2^w}$$

since

1. $\left(\frac{x^{2^k}}{p_i}\right)_{2^s} = 1$, where $1 \leq s \leq k$
2. $\left(\frac{y_j}{p_i}\right)_{2^k} = 1$, where $j \neq i$
3. $\sum_{w=0}^{k-1} b_w 2^w = \left(\sum_{w=0}^{s-1} b_w 2^w\right) + 2^s \cdot \left(\sum_{w=s}^{k-1} b_w 2^{w-s}\right)$

As a result, the message block m_i can be recovered bit by bit using p_i.

Remark 2. The case $\gamma = k = 1$ corresponds to the Goldwasser-Micali cryptosystem [9] and the case $\gamma = 1$ corresponds to the Joye-Libert PKE scheme [11].

Remark 3. In the *Setup* phase, we have to compute a special type of y_i. An efficient way to perform this step is to randomly select $y_{i,i} \xleftarrow{\$} \mathbb{Z}_{p_i}^*$, $y_{i,\gamma} \xleftarrow{\$} \mathbb{Z}_{p_\gamma}^*$ and $w_j \xleftarrow{\$} \mathbb{Z}_{p_j^*}$, compute $y_j \leftarrow w_j^{2^k} \bmod p_j$ and finally use the Chinese remainder theorem to compute an element $y_i \in \mathbb{Z}_n^*$ such that $y_i \equiv y_{i,\ell} \bmod p_\ell$.

3.2 Security Analysis

Theorem 1. *Assume that the* QR *and* SJS *assumptions hold. Then, the proposed scheme is* IND-CPA *secure in the standard model. Formally, let A be an efficient PPT adversary, then there exist two efficient PPT algorithms B_1 and B_2 such that*

$$ADV_A^{\text{IND-CPA}}(\lambda) \leq \frac{3}{2}\gamma \left((k - \frac{1}{3}) \cdot ADV_{B_1}^{\text{QR}}(\lambda) + (k-1) \cdot ADV_{B_2}^{\text{SJS}}(\lambda) \right).$$

Proof. To prove the statement, we simply replace the distribution of the public key y for the encryption query. Let $n_i = p_i p_\gamma$, $i \in [0, \gamma)$. Instead of choosing $y_i \in J_{n_i} \setminus QR_{n_i}$ we choose y_i from the multiplicative subgroup of 2^k residues modulo n_i. Under the GR assumption, the adversary does not detect the difference between the original scheme and the one with the modified y_is. In this case, the value c is not carrying any information about the message. Thus, the IND-CPA security of our proposed cryptosystem follows.

3.3 Implementation and Performance Analysis

Complexity Analysis For simplicity, when computing the ciphertext expansion, the encryption and the decryption complexities, we consider the length of the prime numbers as being λ. Based on the complexities presented in Table 1, we obtain the results listed in Table 2.

Table 2. Performance analysis for an η-bit message

Scheme	Ciphertext size	Encryption Complexity
GM [9]	$2\lambda \cdot \eta$	$\mathcal{O}(2M(2\lambda)\eta)$
JL [11]	$2\lambda \cdot \lceil \frac{\eta}{k} \rceil$	$\mathcal{O}(2(k+1)M(2\lambda)\lceil \frac{\eta}{k} \rceil)$
This work	$(\gamma+1) \cdot \lambda \cdot \lceil \frac{\eta}{\gamma k} \rceil$	$\mathcal{O}((\gamma+1)(k+1)M((\gamma+1)\lambda)\lceil \frac{\eta}{\gamma k} \rceil)$

Scheme	Decryption Complexity
GM [9]	$\mathcal{O}(\log(\lambda)M(\lambda)\eta)$
JL [11]	$\mathcal{O}((2k\lambda + \frac{k^2}{2})M(\lambda)\lceil \frac{\eta}{k} \rceil)$
This work	$\mathcal{O}(\gamma(2k\lambda + \frac{k^2}{2})M(\lambda)\lceil \frac{\eta}{\gamma k} \rceil)$

Implementation Details We further provide the reader with benchmarks for our proposed PKE scheme.

We ran each of the three sub-algorithms on a CPU Intel i7-4790 4.00 GHz and used GCC to compile it (with the O3 flag activated for optimization). Note that for all computations we used the GMP library [2]. To calculate the running times we used the *omp_get_wtime()* function [1]. To obtain the average running time we chose to encrypt 100 128-bit messages.

For generating the primes needed in the *Setup* phase we used the naive implementation[7]. A more efficient method of generating primes is presented in [5, 11].

We further list our results in Table 3 (running times in seconds). When analyzing Table 3, note that in the case $\gamma = 1$ we obtain the Goldwasser-Micali scheme ($k = 1$) and the Joye-Libert scheme ($k = 2, 4, 8$). We stress that we considered $\lambda = 1536$[8].

For completeness, in Table 4 we also present the ciphertext size (in kilobytes $= 10^3$ bytes) for the previously mentioned parameters.

4 An Application to Biometric Authentication

In [6], the authors propose a biometric authentication protocol based on the Goldwasser-Micali scheme. A security flaw[9] of the protocol was indicated and fixed in [3]. A natural extension of Bringer *et al.*'s protocol can be obtained using the scheme proposed in Section 3.1. Thus, we describe our protocol in Section 4.1

[7] *i.e.* we randomly generated $r \xleftarrow{\$} [2^{\lambda-k}, 2^{\lambda-k+1})$ until the $2^k r + 1$ was prime.
[8] According to NIST this choice of λ offers a security strength of 128 bits.
[9] The running time is exponential in the number of users

Table 3. Average running times for a 128-bit message

Algorithm	$\gamma = 1$				
	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
Setup	0.680128	0.632187	0.647911	0.648035	0.606200
Encrypt	0.001062	0.000661	0.000457	0.000333	0.000232
Decrypt	0.091672	0.091081	0.093016	0.090269	0.081925

Algorithm	$\gamma = 2$			
	$k = 1$	$k = 2$	$k = 4$	$k = 8$
Setup	1.115970	1.186430	1.592270	15.27510
Encrypt	0.001050	0.000778	0.000570	0.000477
Decrypt	0.098191	0.096581	0.093230	0.094690

Algorithm	$\gamma = 4$			$\gamma = 8$	
	$k = 1$	$k = 2$	$k = 4$	$k = 1$	$k = 2$
Setup	3.215690	14.67540	762.1870	109.4590	12429.10
Encrypt	0.001287	0.001190	0.001052	0.001829	0.002174
Decrypt	0.098939	0.097237	0.099977	0.096664	0.092930

Table 4. Ciphertext size for a 128-bit message

	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
$\gamma = 1$	49.152	24.576	12.288	6.1440	3.0720
$\gamma = 2$	36.864	18.432	9.2160	4.6080	–
$\gamma = 4$	30.720	15.360	7.6800	–	–
$\gamma = 8$	27.648	13.824	–	–	–

and analyze its security in Section 4.2. A performance analysis is provided in Section 4.3.

4.1 Description

Enrollment Phase In the protocol we consider U_i's biometric template b_i as being a γM-dimensional vector $b_i = \{b_{i,j}\}_{j\in[0,M)}$, where $b_{i,j} = \{b_{i,j,\ell}\}_{\ell\in[0,\gamma)}$ and $b_{i,j,\ell} \in [0, 2^k)$.

In the enrollment phase, U_i registers (b_i, i) at the database $\mathcal{DB}$ and (ID_i, i) at the authentication server $\mathcal{AS}$, where ID_i is U_i's pseudonym and i is the index of record b_i in $\mathcal{DB}$. Let N denote the number of records in $\mathcal{DB}$. Note that the matcher $\mathcal{M}$ possesses a key pair (sk, pk) for the scheme presented in Section 3.1.

We further denote by $\mathcal{E}(pk, \cdot)$ and $\mathcal{E}_{JL}(pk, y_\ell, \cdot)$ the encryption algorithms for the scheme presented in Section 3.1 with $pk = (n, y, k)$ and the Joye-Libert scheme[10] with $pk = (n, y_\ell, k)$, where $\ell \in [0, \gamma)$.

[10] Note that in this case we consider n to be a product of $\gamma + 1$ primes.

Verification Phase If a user U_i wishes to authenticate himself to $\mathcal{AS}$, the next procedure is followed:

1. $\mathcal{S}$ captures the user's biometric data b'_i and sends to $\mathcal{AS}$ the user's identity ID_i together with $\mathcal{E}(pk, b'_i) = \{\mathcal{E}(pk, b'_{i,j})\}_{j\in[0,M)}$. Note that a *liveness link* is available between $\mathcal{S}$ and $\mathcal{AS}$ to ensure that data is coming from the sensor are indeed fresh and not artificial.
2. $\mathcal{AS}$ retrieves the index i using ID_i and then sends $\mathcal{E}_{JL}(pk, y_\ell, t_j)$ to the database, for $\ell \in [0, \gamma)$ and $j \in [0, N)$, where $t_j = 1$ if $j = i$, $t_j = 0$ otherwise.
3. For every $s \in [0, M)$, $\mathcal{DB}$ computes

$$\mathcal{E}(pk, b_{i,s}) = \prod_{j=0}^{N-1}\prod_{\ell=0}^{\gamma-1} \mathcal{E}_{JL}(pk, y_\ell, t_j)^{b_{j,s,\ell}} \bmod n.$$

   To prevent $\mathcal{AS}$ from performing an exhaustive search of the profile space, $\mathcal{DB}$ re-randomizes the encryptions by calculating $\mathcal{E}(pk, b_{i,s}) = x_s^{2^k}\mathcal{E}(pk, b_{i,s})$, where $x_s \xleftarrow{\$} \mathbb{Z}_n^*$. Then, $\mathcal{DB}$ sends $\mathcal{E}(pk, b_{i,s})$, for $s \in [0, M)$ to the authentication server.
4. $\mathcal{AS}$ computes v_s, $s \in [0, M)$, where

$$v_s = \mathcal{E}(pk, b'_{i,s})/\mathcal{E}(pk, b_{i,s}) \bmod n = \mathcal{E}(pk, b'_{i,s} - b_{i,s}), \tag{1}$$

   and $b'_{i,s} - b_{i,s} = \{b'_{i,s,\ell} - b_{i,s,\ell}\}_{\ell\in[0,\gamma)}$. Then, $\mathcal{AS}$ makes a random permutation among v_s, for $s \in [0, M)$, and sends the permuted vector w_s, for $s \in [0, M)$, to $\mathcal{M}$. Note that Item 4 will return a valid result with high probability, thus we do not explicitly require $\mathcal{E}(pk, b_{i,s})$ to be invertible.
5. $\mathcal{M}$ decrypts w_s to check that the taxicab norm of the corresponding plaintext vector

$$\sum_{s=0}^{M-1}\sum_{\ell=0}^{\gamma-1} |w_{s,\ell}|$$

   is equal to or less than d and sends the result $\mathcal{AS}$.
6. $\mathcal{AS}$ accepts or rejects the authentication request accordingly.

Correctness (Requirement 1). We need to show that $v_s = \mathcal{E}(pk, b'_{i,s} - b_{i,s})$, for $s \in [0, M)$. First observe that

$$\begin{aligned}
\mathcal{E}(pk, b_{i,s}) &= \prod_{j=0}^{N-1}\prod_{\ell=0}^{\gamma-1} \mathcal{E}_{JL}(pk, y_\ell, t_j)^{b_{j,s,\ell}} \\
&\equiv \prod_{j=0}^{N-1}\prod_{\ell=0}^{\gamma-1} (r_{j,\gamma}^{2^k} y_\ell^{t_j})^{b_{j,s,\ell}} \\
&\equiv r_i^{2^k} \prod_{\ell=0}^{\gamma-1} y_\ell^{b_{i,s,\ell}} \bmod n.
\end{aligned}$$

Thus,

$$\mathcal{E}(pk, b'_{i,s})/\mathcal{E}(pk, b_{i,s}) \equiv \mathcal{E}(pk, b'_{i,s} - b_{i,s}) \bmod n.$$

It is obvious that the taxicab distance between b_i and b'_i

$$\sum_{s=0}^{M-1} \sum_{\ell=0}^{\gamma-1} |b'_{i,s,\ell} - b_{i,s,\ell}|$$

is equal to the taxicab norm of the plaintext vector corresponding to $\{v_s\}_{s\in[0,M)}$ and $\{w_s\}_{s\in[0,M)}$.

4.2 Security Analysis

The proofs of Theorems 2 and 3 are similar to the security proofs from [6] and, thus, are omitted. The only changes we have to make in the proofs of Theorems 2 and 3 is replacing Goldwasser-Micali with our scheme and, respectively, the Joye-Libert scheme.

Theorem 2 (Requirement 2). *For any identity ID_{i_0} and two biometric templates b'_{i_0}, b'_{i_1}, where $i_0, i_1 \geq 1$ and b'_{i_0} is the biometric template related to ID_{i_0}, any $\mathcal{M}$, $\mathcal{DB}$ and $\mathcal{AS}$ can distinguish between (ID_{i_0}, b'_{i_0}) and (ID_{i_0}, b'_{i_1}) with negligible advantage.*

Theorem 3 (Requirement 3). *For any two users U_{i_0} and U_{i_1}, where $i_0, i_1 \geq 1$, if U_{i_β}, where $\beta \xleftarrow{\$} \{0, 1\}$ makes an authentication attempt, then the database $\mathcal{DB}$ can only guess β with a negligible advantage.*

4.3 Performance Analysis

It is easy to see that the sensor $\mathcal{S}$ and the matcher $\mathcal{M}$ perform only M encryptions and, respectively, decryptions. Comparing our proposed protocol's complexity with Bringer *et al.*'s, reduces to comparing the scheme from Section 3.1 with the Goldwasser-Micali cryptosystem.[11] On the authentication server's side, we perform γN Joye-Libert encryptions (which can be precomputed) and M divisions. Bringer *et al.*'s protocol, performs step 2 using the Goldwasser-Micali scheme and, thus, in step 4 they can use multiplications instead of divisions [12]. Since we took into consideration the fix from [3] when proposing our protocol, we have to perform M extra multiplications compared to the scheme in [6]. Since we have to assemble our scheme's ciphertexts from Joye-Libert's ciphertexts we have a blowout of γ multiplications on the database's side. Thus, we perform $\gamma MN/2$ multiplications on average .

[11] See Section 3.3

[12] In $\mathbb{Z}_2$ addition and subtraction are equivalent.

5 Conclusions and further development

Based on the Joye-Libert scheme we proposed a new PKE scheme, proved its security in the standard model and analyzed its performance in a meaningful context. We also described an application of our cryptosystem to biometric authentication and presented its security analysis.

Future Work. An attractive research direction for the future is the construction of *lossy trapdoor functions* (based on the inherited homomorphic properties of our proposed cryptosystem). Another appealing future work idea is to propose a threshold variant of our scheme and to discuss security and efficiency matters.

References

1. OpenMP. https://www.openmp.org/
2. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/
3. Barbosa, M., Brouard, T., Cauchie, S., De Sousa, S.M.: Secure Biometric Authentication with Improved Accuracy. In: ACISP 2008. Lecture Notes in Computer Science, vol. 5107, pp. 21–36. Springer (2008)
4. Barker, E.: NIST SP800-57 Recommendation for Key Management, Part 1: General. Retrieved January (2016), 147 (2016)
5. Benhamouda, F., Herranz, J., Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. Journal of Cryptology **30**(2), 519–549 (2017)
6. Bringer, J., Chabanne, H., Izabachéne, M., Pointcheval, D., Tang, Q., Zimmer, S.: An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication. In: ACISP 2007. pp. 96–106. Springer (2007)
7. Cohen, J., Fischer, M.: A Robust and Verifiable Cryptographically Secure Ellection Scheme (extended abstract). In: FOCS 1985. pp. 372–382. IEEE Computer Society Press (1985)
8. Crandall, R., Pomerance, C.: Prime Numbers: A Computational Perspective. Number Theory and Discrete Mathematics, Springer (2005)
9. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: STOC 1982. pp. 365–377. ACM (1982)
10. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
11. Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. In: EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 76–92. Springer (2013)
12. Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. IACR Cryptology ePrint Archive **2013/435** (2014)
13. Naccache, D., Stern, J.: A New Public Key Cryptosytem Based on Higher Residues. In: CCS 1998. pp. 59–66. ACM (1998)
14. Simoens, K., Bringer, J., Chabanne, H., Seys, S.: A Framework for Analyzing Template Security and Privacy in Biometric Authentication Systems. IEEE Transactions on Information Forensics and Security **7**(2), 833–841 (2012)
15. Yan, S.Y.: Number Theory for Computing. Theoretical Computer Science, Springer (2002)

A Optimized Decryption Algorithms

In [12], the authors provide the reader with different versions of the decryption algorithm corresponding to the Joye-Libert cryptosystem. We present slightly modified versions of [12, Algorithm 3 and 4] in Algorithms 2 and 3. The authors also propose two other optimizations [12, Algorithm 5 and 6], but their complexity is similar with Algorithm 3 and 4's complexity. Note that these optimizations contain a typo: in line 5, Algorithm 5 and line 6, Algorithm 6 we should have $A^{k-j} \neq C[k-j] \bmod p$ instead of $A \neq C[k-j] \bmod p$.

For these algorithms to work we need to enhance the *Setup* algorithm of our proposed cryptosystem. More precisely, we generate the $\gamma + 1$ prime numbers p_i with the supplementary restriction $p_i \not\equiv 1 \bmod 2^{k+1}$. For $0 \leq i < \gamma$, let $p'_i = (p_i - 1)/2^k$. We precompute $D_i = y_i^{-p'_i}$ for Algorithm 2 and $D_i[j] = D_i^{2^{j-1}} \bmod p_i$, $1 \leq j \leq k-1$, for Algorithm 3 and augment the private key with these values. Remark that Algorithm 3 requires more memory than Algorithm 2.

Algorithm 2: Fast decryption algorithm Version 1

Input: The secret values (p_i, p'_i, D_i), the value y_i and the ciphertext c
Output: The message block m_i

1 $m_i \leftarrow 0$, $B \leftarrow 1$
2 $C \leftarrow c^{p'_i} \bmod p_i$
3 **foreach** $j \in [1, k-1]$ **do**
4 $\quad z \leftarrow C^{2^{k-j}} \bmod p_i$
5 $\quad$ **if** $z \neq 1$ **then**
6 $\quad\quad m_i \leftarrow m_i + B$
7 $\quad\quad C \leftarrow C \cdot D_i \bmod p_i$
8 $\quad$ **end**
9 $\quad B \leftarrow 2B$, $D \leftarrow D^2 \bmod p_i$
10 **end**
11 **if** $C \neq 1$ **then**
12 $\quad m_i \leftarrow m_i + B$
13 **end**
14 **return** m_i

Correctness. Let $m_i = \sum_{w=0}^{k-1} b_w 2^w$ be the binary expansion of block m_i. We define $\alpha_i[s] = 2^{k-s} p_i'$. Note that

$$\begin{aligned} c^{\alpha_i[s]} &\equiv (x^{2^k} \cdot \prod_{v=1}^{\gamma} y_v^{m_v})^{\alpha_i[s]} \\ &\equiv y_i^{\alpha_i[s] \sum_{w=0}^{s-1} b_w 2^w} \\ &\equiv y_i^{b_{s-1} 2^{k-1} p_i'} y^{\alpha_i[s] \sum_{w=0}^{s-2} b_w 2^w} \\ &\equiv (-1)^{b_{s-1}} y^{\alpha_i[s] \sum_{w=0}^{s-2} b_w 2^w} \mod p_i \end{aligned}$$

since

1. $(x^{2^k})^{\alpha_i[s]} = x^{2^{k-s}(p_i-1)} = 1$
2. $\left(\frac{y_j}{p_i}\right)_{2^k} = 1$, where $j \neq i$
3. $\sum_{w=0}^{k-1} b_w 2^w = \left(\sum_{w=0}^{s-1} b_w 2^w\right) + 2^s \cdot \left(\sum_{w=s}^{k-1} b_w 2^{w-s}\right)$
4. $\left(\frac{y_i}{p_i}\right) = -1$

As a result, the message block m_i can be recovered bit by bit using the values p_i, p_i' and the vector D_i.

Algorithm 3: Fast decryption algorithm Version 2

Input: The secret values $(p_i, p_i', D_i[1], \ldots D_i[k-1])$, the value y_i and the ciphertext c

Output: The message block m_i

```
m_i ← 0, B ← 1
C ← c^{p_i'} mod p_i
foreach j ∈ [1, k − 1] do
    z ← C^{2^{k−j}} mod p_i
    if z ≠ 1 then
        m_i ← m_i + B
        C ← C · D_i[j] mod p_i
    end
    B ← 2B
end
if C ≠ 1 then
    m_i ← m_i + B
end
return m_i
```

Implementation Details The complexities of Algorithms 2 and 3 are $\mathcal{O}(\gamma(\lambda + \frac{k^2}{2} + \frac{3k}{2})M(\lambda)\lceil\frac{\eta}{\gamma k}\rceil)$ and $\mathcal{O}(\gamma(\lambda + \frac{k^2}{2} + \frac{k}{2})M(\lambda)\lceil\frac{\eta}{\gamma k}\rceil)$.

We further provide the reader with benchmarks for the optimized versions of our PKE scheme.

Table 5. Average running times for Algorithm 2.

Algorithm	$\gamma = 1$				
	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
Setup	0.736027	0.691385	0.704239	0.673276	0.7184
Encrypt	0.001218	0.000787	0.000516	0.000383	0.000296
Decrypt	0.052399	0.026501	0.013326	0.006679	0.003577

Algorithm	$\gamma = 2$			
	$k = 1$	$k = 2$	$k = 4$	$k = 8$
Setup	1.210450	1.334020	1.926020	15.35740
Encrypt	0.001137	0.000843	0.000638	0.000555
Decrypt	0.052409	0.026340	0.013168	0.007064

Algorithm	$\gamma = 4$			$\gamma = 8$	
	$k = 1$	$k = 2$	$k = 4$	$k = 1$	$k = 2$
Setup	3.662620	15.26860	828.9620	107.6630	14429.00
Encrypt	0.001423	0.001318	0.001058	0.002007	0.00244632
Decrypt	0.054294	0.026909	0.012723	0.052906	0.026168

Table 6. Average running times for Algorithm 3.

Algorithm	$\gamma = 1$				
	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
Setup	0.702962	0.709076	0.684529	0.713416	0.711517
Encrypt	0.001117	0.000796	0.000499	0.000378	0.000287
Decrypt	0.048072	0.024782	0.012958	0.006617	0.003436

Algorithm	$\gamma = 2$			
	$k = 1$	$k = 2$	$k = 4$	$k = 8$
Setup	1.086650	1.181620	1.877680	13.54860
Encrypt	0.001177	0.000798	0.000600	0.000518
Decrypt	0.049691	0.025127	0.012281	0.006574

Algorithm	$\gamma = 4$			$\gamma = 8$	
	$k = 1$	$k = 2$	$k = 4$	$k = 1$	$k = 2$
Setup	3.354720	14.33620	847.9770	104.0870	12741.90
Encrypt	0.001323	0.001296	0.001087	0.001936	0.002280
Decrypt	0.050909	0.026515	0.012982	0.051005	0.024521