A BRIEF INTRODUCTION TO QUADRATIC RESIDUOSITY BASED CRYPTOGRAPHY

FERUCIO LAURENŢIU ŢIPLEA

Communicated by Dan Timotin

The quadratic residuosity problem is the problem to distinguish between the distributions of quadratic residues and quadratic non-residues modulo a composite integer. The intractability of the quadratic residuosity problem is the basis for the security of numerous constructions in cryptography, including public-key encryption schemes, pseudo-random generators, or cryptographic protocols. The aim of this talk is to provide a brief overview on the quadratic residuosity problem and its applications to cryptography.

AMS 2010 Subject Classification: 11T71, 94A60.

Key words: quadratic residue, Jacobi symbol, provable security, public-key encryption, pseudo-random generator.

1. INTRODUCTION AND PRELIMINARIES

Deciding whether or not an integer with the Jacobi symbol 1 is a quadratic residue when the modulus is composite proved to be a hard problem in computing. Failure to efficiently solve this problem has led to the quadratic residuosity assumption (QRA) which states that we cannot distinguish between a quadratic residue and a non-quadratic residue, except with negligible probability, if the modulus is a composite integer.

The QRA has found many applications in cryptography. Probably the first of them was signed by Rabin with the introduction of a public-key encryption scheme whereby messages are encrypted by squaring them (that is, by quadratic residues) modulo a Blum integer [20]. Another important application was proposed by Goldwasser and Micali with the introduction of the concept of probabilistic encryption and semantic security [11, 12]. Thus, they proposed a public-key encryption scheme that achieves semantic security under the QRA. The Blum-Blum-Shub (BBS) generator is another major application of the QRA [5]. The initial state of this generator is a random quadratic residue module a Blum integer. At each iteration, the current state is squared and the least significant bit is outputted.

Shamir's identity-based cryptography proposed in 1984 is an important beneficiary of the QRA. For instance, Cocks proposed in 2001 the first identity-based encryption scheme based on quadratic residues [8]. The scheme encrypts messages bit by bit and each encrypted bit is a pair of two integers. The decryption consists of computing the Jacobi symbol of one of the two integers in each pair. Although Cocks' IBE scheme is efficient only for small messages, it is very elegant and *per se* revolutionary. The scheme attracted the interest of many researchers [6, 3, 7, 16].

Contribution. The aim of this paper is to provide a brief introduction to quadratic residuosity based cryptography. We begin by provable security, hardness assumptions, and gap groups. We then advance to pseudo-random generators where we discuss the Blum-Blum-Shub generator and the Jacobi generator. Applications to public-key cryptography are presented in the next section. The focus is on the Goldwasser-Micali scheme, Cocks scheme for public-key and identity-based encryption, and then the Boneh-Gentry-Hamburg scheme.

Due to the space limitation our survey does not include recent results on the applications of high-order residues to cryptography.

Paper organization. Our paper is organized in five sections. The rest of this section is reserved to preliminary concepts and notations. Section 2 discusses provable security. The first application of quadratic residuosity to cryptography is the topic of the third section, where two pseudo-random generators are presented. In the fourth section we focus on applications of quadratic residuosity to public-key cryptography. We conclude in the last section.

Preliminaries. We recall a few concepts on number theory. For details, the reader is referred to standard textbooks such as [18].

The set of integers is denoted by $\mathbb{Z}$. Given $n \in \mathbb{Z}$, two integers a and b are called *congruent modulo* n, denoted $a \equiv b \bmod n$ or $a \equiv_n b$, if n divides $a - b$. $\mathbb{Z}_n$ stands for the set of remainders modulo n, and $\mathbb{Z}_n^*$ is the subset of integers in $\mathbb{Z}_n$ that are co-prime to n. Euler's totient function $\phi(n)$ gives the cardinality of $\mathbb{Z}_n^*$. A Blum prime is a prime integer that is co-prime to 3 modulo 4. A Blum integer is the product of two distinct Blum primes.

An integer a co-prime with n is a *quadratic residue modulo* n if $a \equiv_n x^2$, for some integer x; the integer x is called a *square root* of a modulo n. QR_n (QNR_n, $SQRT_n(a)$, resp.) stands for the set of quadratic residues (quadratic non-residues, square roots of a, resp.) modulo n. The *Legendre symbol* of an integer a modulo a prime p, denoted $\left(\frac{a}{p}\right)$, is 1 if a is a quadratic residue modulo p, 0 if p divides a, and -1 otherwise. The *Jacobi symbol* extends the Legendre

symbol to composite moduli. If $n = p_1^{e_1} \cdots p_m^{e_m}$ is the prime factorization of the positive integer n, then the Jacobi symbol of a modulo n is

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \cdots \left(\frac{a}{p_m}\right)^{e_m}$$

For the sake of simplicity we will use the terminology of Jacobi symbol in both cases (prime or composite moduli). For details regarding basic properties of the Jacobi symbol the reader is referred to [18, 24]. J_n stands for the set of integers in $\mathbb{Z}_n^*$ with the Jacobi symbol 1.

For concepts related to algorithms and complexity we follow [25]. Given a set A, $a \leftarrow A$ means that a is uniformly at random chosen from A. If $\mathcal{A}$ is a probabilistic algorithm, then $a \leftarrow \mathcal{A}$ means that a is an output of $\mathcal{A}$ for some given input. The probability of an event X is denoted $P(X)$, and $P(X \mid Y)$ stands for the probability of X conditioned by Y.

The asymptotic approach to security makes use of security parameters, denoted by λ in our paper. A positive function $f(\lambda)$ is called *negligible* if for any positive polynomial $poly(\lambda)$ there exists n_0 such that $f(\lambda) < 1/poly(\lambda)$, for any $\lambda \geq n_0$.

2. PROVABLE SECURITY

2.1. PROVING SECURITY IN CRYPTOGRAPHY

For many years, the common approach to validate the security of a cryptographic scheme was to search for attacks against it. If an attack was found, the scheme was declared insecure; otherwise, with the passing of time, not finding an attack increased the confidence in the security of the scheme. With such an approach we can never be sure that there is no attack on the scheme.

The *provable security paradigm* proposes a general formal approach to the security of a cryptographic system. It assumes rigorous formalization of the cryptographic system, of the security property to be studied, of the type of adversary against whom the security property is to be studied, as well as the establishment of proof methods and techniques to analyze whether the security property is valid or not (within that cryptographic system and against that type of adversary).

The first major step in using this paradigm in cryptography was taken by Shannon in [23]. Shannon introduced rigorously the concept of encryption scheme, the property of *perfect secrecy*, and allowed adversaries (Turing algorithms) with unlimited power. According to him, an encryption scheme

provides perfect secrecy if the ciphertext reveals no information to the adversary about the message it encrypts (assuming that the adversary knows the ciphertext). An equivalent formulation says that the encryption scheme has perfect secrecy if it is impossible to distinguish an encryption of a message m_0 from one of a message m_1 based on a given ciphertext (this is called perfect indistinguishability). Although perfect secrecy is a big achievement in theory, it is too strong in practice because it leads to the fact that the encryption keys must be of the same length as the messages they encrypt [23].

The second major step in applying the provable security paradigm in cryptography was made in 1982 by Goldwasser and Micali [11]. They introduced the concept of *semantic security*, which is an adaptation of Shannon's perfect secrecy to the computational setting, considering only adversaries having bounded computational resources. The proof technique was by reduction from a hard problem. Roughly speaking, an encryption scheme is semantically secure if no probabilistic polynomial-time (PPT) algorithm (that plays the role of an adversary) that is given a ciphertext of a certain message (taken from any distribution on messages) can predict anything better about the message than when it is not given the ciphertext, except with negligible probability.

Provable security is of particular importance in cryptography. Applying this paradigm to a cryptographic scheme requires:

1. A formalization of the cryptographic scheme;
2. A *security model* $\mathcal{S}$ [17], which consists of:
   (a) a *security goal*, such as semantic security (SS), indistinguishability (IND), or non-maleability (NM);
   (b) an *attack model*, such as chosen plaintext attack (CPA) or chosen ciphertext attack (CCA1 or CCA2);
3. A *hardness assumption* $\mathcal{H}$ about some algorithmic problem;
4. A *reductionist proof* of $\mathcal{H}$ to $\mathcal{S}$ in case that the cryptographic scheme is to be proven to achieve the security goal under the attack model specified by $\mathcal{S}$.

2.2. HARDNESS ASSUMPTIONS

A *hardness assumption* is a hypothesis that a particular problem cannot be solved efficiently (in a given model of computation). An assumption A is *stronger* than an assumption B when A implies B (usually, the converse is asked to be false or not known). In such a case, B is *weaker* than A.

Hardness assumptions are crucial in provable security, as the methodology sketched in the previous sub-section shows. Namely, we usually relate the security problem to a hardness assumption about a problem better-understood. Among the most common hardness assumptions in cryptography are:

1. *Factoring assumption.* The integer factorization is without any doubt the most studied hard problem in algorithmic number theory. Given a PPT algorithm *Gen* that on input λ outputs a triple (n, p, q), where p and q are λ-bit primes and $n = pq$, we say that factoring is *hard relative to Gen* if
$$P((p, q) \leftarrow \mathcal{A}(n) \mid (n, p, q) \leftarrow Gen(\lambda))$$
is negligible as a function of λ, for any PPT algorithm $\mathcal{A}$.

   The *factoring assumption* is that there exists an algorithm *Gen* relative to which integer factorization is hard;

2. *RSA assumption.* Given a PPT algorithm *RSAGen* that on input λ outputs (n, e, d), where n is the product of two distinct λ-bit primes, $e \in \mathbb{Z}^*_{\phi(n)}$, and $e \cdot d \equiv_{\phi(n)} 1$, we say that the *RSA problem is hard relative to RSAGen* if
$$P(x \leftarrow \mathcal{A}(n, e, y) \,:\, y = x^e \bmod n \mid (n, e, d) \leftarrow RSAGen(\lambda),\, y \leftarrow \mathbb{Z}^*_n)$$
is negligible as a function of λ, for any PPT algorithm $\mathcal{A}$.

   The *RSA assumption* is that there exists an algorithm *RSAGen* relative to which the RSA problem is hard;

3. *Discrete logarithm assumption.* Given a PPT algorithm *GroupGen* that on input λ outputs (G, q, g), where G is a cyclic group of order q, q is a λ-bit prime, and g is a generator of G, we say that the *discrete logarithm problem is hard relative to GroupGen* if
$$P(x \leftarrow \mathcal{A}(G, q, g, y) \,:\, y = g^x \mid (G, q, g) \leftarrow GroupGen(\lambda),\, y \leftarrow G)$$
is negligible as a function of λ, for any PPT algorithm $\mathcal{A}$.

   The *discrete logarithm assumption* is that there exists an algorithm *GroupGen* relative to which the discrete logarithm problem is hard.

The status of a probem to be hard depends on the model of computation. The *standard model of computation*, which is the most commonly used model, regards algorithms as Turing machines. The *generic model of computation* considers generic algorithms modeled by oracles that perform only certain group/ring operations and test of equality. Why such a model? Because:

- Proving useful lower bounds in the standard model is sometimes a real challenge;

- Many important "hard" problems can be formulated in various groups or rings for which very few properties are known (e.g., elliptic curve groups);
- Algorithms in generic models are independent of the operands' representation and so they can be translated to any concrete instantiation.

All assumptions in this talk are with respect to the standard model of computation. However, some comments will also be made with respect to a generic model of computation.

For a quite long time researchers have investigated the relationship between the integer factorization problem and the RSA problem. Clearly, hardness of the RSA problem implies hardness of the factorization problem. However, no one knows whether the converse holds true in the standard model. If the generic model is employed, then we have the following result:

THEOREM 1 ([1]). *Breaking the RSA problem is equivalent to factoring in the generic ring model of computation.*

Let us consider now a closely related problem to the RSA problem. Given a PPT algorithm *BlumGen* that on input λ outputs (n,p,q), where p and q are distinct λ-bit Blum primes and $n=pq$, we say that the *Rabin problem is hard relative to BlumGen* if

$$P(x \leftarrow \mathcal{A}(n,y) \,:\, y = x^2 \bmod n \mid (n,p,q) \leftarrow BlumGen(\lambda),\, y \leftarrow \mathbb{Z}_n^*)$$

is negligible as a function of λ, for any PPT algorithm $\mathcal{A}$.

Note that the Rabin problem is not the particular case of the RSA problem for $e=2$ because 2 is not co-prime with $\phi(n)$.

THEOREM 2 ([20]). *Breaking the Rabin problem is equivalent to factoring in the standard model.*

2.3. GAP GROUPS

Most problems have two facets: *computational* and *decisional*. We exemplify this on the *Diffie-Hellman (DH) problem*. Its computational version, denoted CDH, asks to compute g^{ab} given (G,q,g), g^a, and g^b (please see $GroupGen(\lambda)$). The decisional version, denoted DDH, asks to decide whether or not g^{ab} equals g^z, given (G,q,g), g^a, g^b, g^{ab}, and g^z. The hardness assumptions of these problems can be formulated as for the problems discussed in the previous sub-section.

It is an immediate consequence the fact that DDH is reducible to CDH (if CDH is easy than DDH is easy). However, the converse is an open problem. A group G where DDH is easy but CDH is hard is called a *gap group*.

In 2009, Hofheinz and Kiltz gave an example of a group that behaves like a gap group. Given a Blum integer $n = pq$, they viewed $\mathbb{Z}_n$ with the representatives $\mathbb{Z}_n = \{-\frac{n-1}{2}, \ldots, -1, 0, 1, \ldots, \frac{n-1}{2}\}$. Then, for a subgroup G of $\mathbb{Z}_n^*$ define $sG = \{|x| \mid x \in G\}$ and the binary operation $|x| \circ |y| = |x \cdot y|$, where $|\cdot|$ denotes the absolute value function. It is easy to see that $(sG, \circ)$ is a group, called the *signed group associated to* G.

THEOREM 3 ([14]). *Let n be a Blum integer. Then,*

1. *$(sQR_n, \circ)$ is a group;*
2. *$sQR_n = sJ_n$ has order $\phi(n)/4$;*
3. *If QR_n is cyclic, then sQR_n is cyclic.*

sQR_n behaves like a gap group: deciding membership to sQR_n is easy (Theorem 3(2)), but computing square roots is as hard as factoring n.

2.4. **INDISTINGUISHABILITY**

Given a PPT algorithm $\mathcal{A}$, define the *advantage of $\mathcal{A}$ in distinguishing two random variables A and B* as

$$Adv_{\mathcal{A},A,B} = |P(1 \leftarrow \mathcal{A}(A)) - P(1 \leftarrow \mathcal{A}(B))|$$

Extend this to families of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ by means of the function

$$Adv_{\mathcal{A},X,Y}(n) = Adv_{\mathcal{A},X_n,Y_n}, \quad \text{for all } n.$$

Now, we say that X and Y are *computationally indistinguishable* if $Adv_{\mathcal{A},X,Y}$ is negligible (as a function of n), for all PPT algorithms $\mathcal{A}$.

Given a generator Gen define the following two families of random variables $P_{QR} = (P_{QR}(\lambda))_{\lambda\in\mathbb{N}}$ and $P_{QNR} = (P_{QNR}(\lambda))_{\lambda\in\mathbb{N}}$:

1. $P_{QR}(\lambda):\ n \leftarrow Gen(\lambda),\ x \leftarrow QR_n,$ output (n, x);
2. $P_{QNR}(\lambda):\ n \leftarrow Gen(\lambda),\ x \leftarrow J_n \setminus QR_n,$ output (n, x).

Now, the *quadratic residuosity problem* (QRP) is the problem to distinguish between these two families of random variables. For the time being, no one knows how to solve efficiently this problem without factoring.

The *quadratic residuosity assumption* (QRA) is that there exists an algorithm Gen relative to which the QRP problem is hard.

The following facts summarize the knowledge about this problem:

1. In the standard model

(a) QRA is harder than the factoring assumption (from definitions);

(b) Open problem: Is factoring reducible to QRP?

2. In the generic ring model

(a) Computing Jacobi symbols is as hard as factoring, while it is easy in the standard model [15];

(b) QRP is equivalent to factoring [15].

3. PSEUDO-RANDOM GENERATORS

3.1. INTRODUCTION

A *pseudo-random generator* (PRG) is an algorithm which takes as input a security parameter λ and a seed and produces as output a sequence of bits whose distribution is indistinguishable from the random uniform distribution.

Given a polynomial ℓ with positive values, $X_{\ell,n}$ denotes an arbitrary but fixed probability distribution over $\{0,1\}^{\ell(n)}$, while $U_{\ell,n}$ denotes the uniform distribution over $\{0,1\}^{\ell(n)}$. When $\ell(n)=n$ for all n, the subscript ℓ will be omitted from the notation above.

A family of distributions $X_\ell=(X_{\ell,n})_{n\in\mathbb{N}}$ is called *pseudo-random* if it is computationally indistinguishable from $U_\ell=(U_{\ell,n})_{n\in\mathbb{N}}$. That is, the function $Adv_{\mathcal{A},X_\ell,U_\ell}(n)$ is negligible, for all PPT algorithms $\mathcal{A}$.

A PPT algorithm $\mathcal{A}$ as above that tries to distinguish between X_ℓ and U_ℓ is called a *polynomial time statistical test*. If $Adv_{\mathcal{A},X_\ell,U_\ell}$ is negligible, we say that X_ℓ *passes the test* $\mathcal{A}$; otherwise, X_ℓ *fails the test* $\mathcal{A}$.

A family of distributions $X_\ell=(X_{\ell,n})_{n\in\mathbb{N}}$ passes the *next bit test* if for any PPT algorithm $\mathcal{A}$, $n\geq 0$, and $1\leq i<\ell(n)$,

$$\left|P(t_{i+1}\leftarrow\mathcal{A}(1^n,t[1,i]):t\leftarrow X_{\ell,n})-\frac{1}{2}\right|$$

is negligible (as a function of n), where t_i stands for the ith bit of t and $t[1,i]$ stands for the prefix $t_1\cdots t_i$.

The following very important result reduces the study of pseudo-randomness to the next bit test.

THEOREM 4 ([30]). *A family of distributions $X_\ell=(X_{\ell,n})_{n\in\mathbb{N}}$ is pseudo-random if and only if it passes the next bit test.*

A *pseudo-random generator* (PRG) is a deterministic polynomial-time algorithm G satisfying the following conditions:

1. There exists a function $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(n) > n$ for all n;
2. $|G(s)| = \ell(|s|)$ for all $s \in \{0,1\}^*$;
3. The distribution $(G_{\ell,n})_{n\in\mathbb{N}}$ of G's outputs on inputs from $U = (U_n)_{n\in\mathbb{N}}$ is pseudo-random.

3.2. **THE BBS PRG**

The Blum-Blum-Shub (BBS) generator was proposed in 1982 [4, 5]. The starting point is Rabin's public-key encryption scheme (please see the Rabin problem in Section 2.2).

Blum-Blum-Shub PRG [4, 5]

1. Generate $(n, p, q) \leftarrow BlumGen(\lambda)$ and $a \leftarrow QR_n$;
2. Output $LSB(a) \parallel LSB(a^2 \bmod n) \parallel LSB((a^2)^2 \bmod n) \parallel \cdots$, where LSB stands for the least significant bit.

The function $x \mapsto x^2 \bmod n$, where n is a Blum integer, is a permutation on QR_n. This can be easily shown by taking into account the fact that each quadratic residue has exactly one square root that is a quadratic residue too when the modulus is a Blum integer. From this remark one can easily obtain:

THEOREM 5 ([4, 5]). *The BBS PRG is unpredictable under the factoring assumption.*

The BBS PRG is quite slow: each iteration needs a modular multiplication while just one bit is extracted. An improvement was proposed by Vazirani et al. and Alexi et al.

THEOREM 6 ([29, 2]). *If* $\log\log n$ *bits are outputted at each iteration in the BBS PRG, it still remains unpredictable under the factoring assumption.*

Extracting the maximum number of bits in the BBS PRG while maintaining its cryptographic security remains an open problem.

3.3. **THE JACOBI PRG**

Damgård has proposed another method to obtain pseudo-random binary sequences, namely by using Legendre and Jacobi sequences [10]. These are simply obtained by computing the Legendre or Jacobi symbol of a sequence of integers, usually taken in increasing order. Clearly, the security of such sequences depend on the distribution of Legendre and Jacobi symbols, a problem

that still needs deep research. It was shown in [10] that weakly unpredictable Legendre sequences may be used to define strongly unpredictable Jacobi sequences. However, the question of whether Legendre sequences are weakly unpredictable has left open.

Given a *security parameter* λ, a λ-bit positive integer n, an integer $a \in \mathbb{Z}_n^*$, and a positive integer ℓ of polynomial size in λ, define an (n, a, ℓ)-*Jacobi sequence* [10] as being the sequence of Jacobi symbols

$$\left(\frac{a}{n}\right), \left(\frac{a+1}{n}\right), \ldots, \left(\frac{a+\ell}{n}\right)$$

The integer a is the *root* or *starting point*, n is the modulus, and ℓ is the length of the sequence. An *Jacobi generator* is a deterministic polynomial-time algorithm G that, when seeded by λ, n, a, and ℓ as above, generates a Jacobi sequence of length ℓ with the root a.

One of the main questions on such generators is about their pseudo-randomness. This clearly can be reduced to the prediction of the next element of a sequence of polynomial length. In other words, given a sequence as above, where ℓ is polynomial in the security parameter λ, the question is to predict the next symbol $\left(\frac{a+\ell+1}{n}\right)$.

Damgård introduced two notions of unpredictability:

1. A generator G is *strongly unpredictable* if for any polynomial *poly*, any probabilistic circuit C, there exists λ_0 such that
$$P(C(G(a)_1, \ldots, G(a)_{i-1}) = G(a)_i) < \frac{1}{2} + \frac{1}{poly(\lambda)},$$
for all $\lambda \geq \lambda_0$ and all $i > 1$;
2. A generator G is *weakly unpredictable* if for any polynomial *poly*, any probabilistic circuit C, there exists λ_0 such that
$$P(C(G(a)_1, \ldots, G(a)_{i-1}) = G(a)_i) < 1 - \frac{1}{poly(\lambda)},$$
for all $\lambda \geq \lambda_0$ and all $i > 1$.

Given a polynomial $Q(\lambda)$ and $a = (a_1, \ldots, a_{Q(\lambda)})$, define G^Q as being

$$G^Q(a) = G(a_1) \oplus \cdots \oplus G(a_{Q(\lambda)})$$

(the XOR is component-wise computed).

THEOREM 7 ([10]). *The generator G^{λ^2} is strongly unpredictable, provided that the generator G is weakly unpredictable.*

COROLLARY 8 ([10]). *If the Legendre generator is weakly unpredictable then the Jacobi generator is strongly unpredictable.*

Whether or not the Legendre generator is weakly unpredictable is an interesting open problem.

4. PUBLIC-KEY CRYPTOGRAPHY

Quadratic residuosity plays an important role in building public-key encryption (PKE) schemes. The first construction of this type was proposed by Goldwasser and Micali [11, 12]. Later, Cocks [8] and Boneh, Gentry, and Hamburg [6], have proposed identity-based encryption schemes that also use quadratic residuosity. We will present in the following the main ideas underlying the construction of these schemes.

4.1. BITS AS QUADRATIC (NON-)RESIDUES

In their seminal paper in 1982 [11, 12], Goldwasser and Micali proposed the first probabilistic public-key encryption scheme that achieves semantic security under the QRA. Its main idea is the following:

- Each bit $b \in \{0,1\}$ is viewed as one of the integers -1 or 1 by a suitable encoding such as $(-1)^b$;
- Encrypt $m = 1$ as an integer in QR_n and $m = -1$ as an integer in $J_n \setminus QR_n$. This can simply be done in the form $c = m \cdot r^2 \bmod n$, where n is a Blum integer and $r \leftarrow \mathbb{Z}_n^*$;
- The decryption of c requires to decide whether c is a quadratic residue modulo n or not, and this can efficiently be done by the factorization of n. Remark that the Jacobi symbol $\left(\frac{c}{n}\right)$, which can be efficiently computed without the factorization of n, does not help to decrypt c because this symbol is always 1 (n is a Blum integer).

Goldwasser-Micali PKE scheme [12]

Setup(λ)**:** Generate $(n,p,q) \leftarrow BlumGen(\lambda)$, make n as the public key, and keep (p,q) as the private key;

Encrypt(m,n)**:** To encrypt a bit $m \in \{-1,1\}$ by the public key n, choose at random $r \leftarrow \mathbb{Z}_n^*$ and output the ciphertext $c = m \cdot r^2 \bmod n$;

Decrypt($c,(p,q)$)**:** Return $m = 1$ if $c \in QR_n$, and -1, otherwise. This can efficiently be done by testing whether $\left(\frac{c}{p}\right) = 1$ and $\left(\frac{c}{q}\right) = 1$.

THEOREM 9 ([12]). *The Goldwasser-Micali PKE scheme is IND-CPA secure under the QRA for BlumGen.*

Even though the Goldwasser-Micali scheme is not efficient in practice because it encrypts bit by bit, its realization has been an important step in developing the concepts of probabilistic encryption and provable security. Besides, the way of looking at bits as quadratic residues or non-residues has been the starting point for many cryptographic schemes.

4.2. HIDING BITS IN QUADRATIC CONGRUENCES

In 2001, Cocks proposed a PKE scheme [8] where each bit $m \in \{-1, 1\}$ is firstly encoded by a random integer t with the Jacobi symbol m, and then t is hidden in a quadratic congruence $c = t + at^{-1} \bmod n$, where n is a Blum integer, $a \in QR_n$, and c is the ciphertext. The choice of a as a quadratic residue is motivated by the fact that, in such a case, the congruence

$$t^2 - ct + a \equiv 0 \bmod n$$

has four solution in t, of which two have the Jacobi symbol 1 and the other two have the Jacobi symbol -1. Therefore, even if this congruence can be efficiently solved, it encrypts -1 and 1 with equal probability. However, to get the correct decryption one may use a square root r of a and the congruence

$$c + 2r \equiv_n t(1 + rt^{-1})^2,$$

which shows that $\left(\frac{c+2r}{n}\right) = \left(\frac{t}{n}\right) = m$.

Cocks PKE scheme [8]

Setup(λ)**:** Generate $(n, p, q) \leftarrow BlumGen(\lambda)$ and $r \leftarrow \mathbb{Z}_n^*$ and output the public key (n, a), where $a = r^2 \bmod n$. The private key is r;

Encrypt$(m, (n, a))$**:** To encrypt a bit $m \in \{-1, 1\}$ by the public key (n, a), choose at random $t \in \mathbb{Z}_n^*$ such that $\left(\frac{t}{n}\right) = m$ and output the ciphertext $c = t + at^{-1} \bmod n$;

Decrypt(c, r)**:** Output $\left(\frac{c+2r}{n}\right)$.

THEOREM 10 ([8]). *The Cocks PKE scheme is IND-CPA secure under the QRA for BlumGen.*

In 1984 Adi Shamir introduced *identity-based encryption* (IBE) [22] as a special case of public-key encryption. This model avoids the public-key infrastructure and the trust chain for public keys. It uses instead a string which

uniquely identifies the receiver and computes his public key based on it, using a publicly known hash function.

An IBE scheme consists of four PPT algorithms, *Setup*, *Extract*, *Encrypt*, and *Decrypt*. $Setup(\lambda)$ outputs the public parameters PP together with the master secret msk, having as input the security parameter λ. The algorithm $Extract(PP, msk, ID)$ outputs the secret key for the identity ID. The third algorithm, $Encrypt(PP, ID, m)$, computes the ciphertext of the message m for a given identity, while the last algorithm, $Decrypt(c, r_{ID})$, decrypts the cryptotext c using the secret key r_{ID} of the identity ID.

Regarding the security, an IBE scheme is said to be IND-ID-CPA secure if the advantage of any efficient PPT adversary against the scheme is negligible, where the adversary is allowed to query the challenger for the secret keys corresponding to the identities it chooses (please see [27] for more details).

The Cocks PKE scheme can easily be transformed into an IBE scheme. What we have to do is to consider a truly random function $h : \{0,1\}^* \to J_n$ to map identities ID into integers with the Jacoby symbol 1 modulo n. To this, we remark that if $a = h(ID)$ is not a quadratic residue, then $-a$ is (n is a Blum integer). Therefore, to be able to decrypt, each bit $m \in \{-1, 1\}$ has to be encrypted by both a and $-a$. The private key of the decryptor will be a square root of a, if $a \in QR_n$, or of $-a$, if $-a \in QR_n$.

One may also remark that $-a$ can be replace by any product $e \cdot a \bmod n$, where $e \in J_n \setminus QR_n$. Moreover, in this case n is not required to be a Blum integer. Thus, we arrive at the following general version of the Cocks IBE scheme.

Cocks IBE scheme [8]

Setup(λ)**:** Generate $(n, p, q) \leftarrow Gen(\lambda)$, $e \leftarrow J_n \setminus QR_n$, and output the public parameters $PP = (n, e, h)$, where h is a hash function that maps identities to J_n. The master key is the factorization of n, namely (p, q);

Extract$(PP, (p, q), ID)$**:** Let $a = h(ID)$. If $a \in QR(n)$, set the private key as a random square root r of a; otherwise set the private key as a random square root r of ea;

Encrypt(PP, ID, m)**:** Let $a = h(ID)$. To encrypt a bit $m \in \{-1, 1\}$, randomly choose $t_1, t_2 \in \mathbb{Z}_n^*$ such that $\left(\frac{t_1}{n}\right) = \left(\frac{t_2}{n}\right) = m$. Compute then $c_1 = t_1 + at_1^{-1} \bmod n$ and $c_2 = t_2 + eat_2^{-1} \bmod n$ and output the pair (c_1, c_2) as being the ciphertext associated to m;

Decrypt$((c_1, c_2), r)$**:** Set $c = c_1$ if $r^2 \equiv a \bmod n$, and $c = c_2$, otherwise. Then, $m = \left(\frac{c+2r}{n}\right)$.

THEOREM 11 ([8, 13]). *The Cocks IBE scheme is IND-ID-CPA secure in the random oracle model under the QRA for Gen.*

The Cocks IBE scheme encrypts a message bit by bit, and each bit is encrypted by $2\log n$ bits, where n is the modulus used by the scheme. Therefore, the Cocks IBE scheme can be considered very bandwidth consuming. As Cocks remarked in his paper [8], the scheme can be used in practice to encrypt short session keys in which case it becomes very attractive.

The Cocks IBE scheme is not anonymous in the sense that the Cocks cryptotexts contain information about the receiver so one can check if the ciphertext was encrypted for a specific identity. For details regarding this interesting topic the reader is referred to [19, 28].

4.3. POLYNOMIALS ASSOCIATED IN RESIDUOSITY

The encryption by a stream cipher starts from the idea that the encryptor and decryptor share a generator and an initial seed to generate the same stream key for both of them. The encryption is usually by XOR. Think now that the message to be encrypted is a sequence of Jacobi symbols and the XOR is replaced by multiplication. A key generator should now generate sequences of Jacobi symbols or integers to compute the Jacobi symbols. If it generates integers, then it may be the case that the decryptor's generator generates a different sequence of integers than the one generated by the encryptor's generator. However, for the correctness of decryption, the two sequences must lead to the same sequence of Jacobi symbols.

Trying to extrapolate this idea to IBE, we see that the decryptor's sequence of integers must depend on the private key (extracted from its identity), while the encryptor's sequence of integers must depend on some secret initially chosen by encryptor.

Definition 12. Let n be a positive integer, $a, S \in \mathbb{Z}_n^*$, and $f, g \in \mathbb{Z}_n[x]$. We say that (f, g) is a pair of (a, S)-*associated polynomials* if the following properties hold:

1. if $a, S \in QR_n$, then $f(r)g(s) \in QR_n$, for all $r \in SQRT_n(a)$ and $s \in SQRT_n(S)$;
2. if $a \in QR_n$, then $f(r)f(-r)S \in QR_n$, for all $r \in SQRT_n(a)$.

Roughly speaking, the integer a will play the role of public key, while each $r \in SQRT_n(a)$ will be a private key. The square roots of S are used to randomize the encryption. Thus, the first condition in Definition 12, which

implies $\left(\frac{g(s)}{n}\right) = \left(\frac{f(r)}{n}\right)$, guarantees the correctness of the decryption process: a bit m is encrypted by multiplying it by $\left(\frac{g(s)}{n}\right)$, and the result is decrypted by multiplying the ciphertext by $\left(\frac{f(r)}{n}\right)$. The second condition in Definition 12 is less intuitive: it is necessary to prove security.

The following IBE scheme, called $BasicIBE$, was proposed in [6].

$BasicIBE$ scheme [6]
% In this scheme, $\mathcal{D}$ is an unspecified deterministic algorithm that on
% input (n, a, S) outputs a pair (f, g) of (a, S)-associated polynomials,
% where n is a positive integer and $a, S \in \mathbb{Z}_n^*$.

Setup(λ)**:** Generate $(n, p, q) \leftarrow Gen(\lambda)$, $e \leftarrow J_n \setminus QR_n$, and choose a hash function $h : \{0,1\}^* \times \{1, \ldots, \ell\} \to J_n$ for some integer $\ell \geq 1$. Output the public parameters $PP = (n, e, h)$; the master key $msk = (p, q, K)$ is the factorization of n together with a random key K of some pseudo-random function $F_K : \{0,1\}^* \times \{1, \ldots, \ell\} \to \{0, 1, 2, 3\}$ (F_K chooses one of the four square roots of $h(ID, i)$ or $eh(ID, i)$, depending on which of them is a quadratic residue);

Extract(PP, msk, ID)**:** For each $1 \leq j \leq \ell$, let $a_j = h(ID, j)$ and $i_j = F_K(ID, j)$. If r_0, r_1, r_2, r_3 is a fixed total ordering of the square roots of a_j or ea_j (depending on which of them is a quadratic residue), then the private key is $r = (r_{i_1}, \ldots, r_{i_\ell})$;

Encrypt(PP, ID, m)**:** Assume $m = m_1 \cdots m_\ell \in \{-1, 1\}^\ell$ is the ℓ-bit sequence to be encrypted. The encryption process is as follows:

- Generate at random $s \in \mathbb{Z}_n^*$ and set $S = s^2 \bmod n$;
- For $j := 1$ to ℓ do
  - Compute $a_j = h(ID, j)$;
  - Compute $(f_j, g_j) = \mathcal{D}(n, a_j, S)$ and $(\bar{f}_j, \bar{g}_j) = \mathcal{D}(n, ea_j, S)$;
  - Compute $c_j = m_j \cdot \left(\frac{g_j(s)}{n}\right)$ and $\bar{c}_j = m_j \cdot \left(\frac{\bar{g}_j(s)}{n}\right)$;
- Return $(c, \bar{c}, S)$, where $c = c_1 \cdots c_\ell$ and $\bar{c} = \bar{c}_1 \cdots \bar{c}_\ell$;

Decrypt$((c, \bar{c}, S), r)$**:** For $j := 1$ to ℓ do

- Compute $a_j = h(ID, j)$;
- If $a_j \in QR_n$ then $b_j = a_j$ else $b_j = ea_j$;
- Compute $(u_j, v_j) = \mathcal{D}(n, b_j, S)$;

- Compute $m_j = c_j \cdot \left(\frac{u_j(r_{i_j})}{n}\right)$;

Return $m = m_1 \cdots m_\ell$.

As with respect to the security of $BasicIBE$ we have the following result (please see [26] for an alternative approach).

THEOREM 13 ([6]). *The $BasicIBE$ scheme is IND-ID-CPA secure in the random oracle model under the QRA for Gen and provided that F is a secure pseudo-random function.*

We emphasize that $BasicIBE$ is an abstract IBE scheme because no concrete algorithm $\mathcal{D}$ to compute (a, S)-associated polynomials is presented. In [6], the method proposed to construct such polynomials is based on the congruence $QC_n(a, S)$ given by

$$ax^2 + Sy^2 \equiv 1 \bmod n,$$

where $n = pq$ is an RSA modulus and $a, S \in \mathbb{Z}_n^*$. Any solution (x_0, y_0) to $QC_n(a, S)$ gives rise to two polynomials f and g

$$\begin{aligned} f(r) &= x_0 r + 1 \bmod n \\ g(s) &= 2(y_0 s + 1) \bmod n \end{aligned}$$

that are (a, S)-associated (see [6] for details).

The $BasicIBE$ scheme is more space efficient than the Cocks IBE scheme: ℓ bits are encrypted by $2\ell + \log n$ bits. The time complexity of the $BasicIBE$ scheme depends on the time complexity of the algorithm $\mathcal{D}$. If this implements the method described above, then the encryptor must solve 2ℓ equations of the form $QC_n(a_i, S)$ and $QC_n(ea_i, S)$, for all $1 \leq i \leq \ell$. The decryptor needs to solve only ℓ of these equations. An improvement at the decryptor side can be obtained starting from the remark that if (x_1, y_1) is a solution to $QC_n(a, S)$ and (x_2, y_2) is a solution to $QC_n(e, S)$, then (x_3, y_3) is a solution to $QC_n(ea, S)$, where $x_3 = \frac{x_1 x_2}{S y_1 y_2 + 1} \bmod n$ and $y_3 = \frac{y_1 + y_2}{S y_1 y_2 + 1} \bmod n$. Therefore, the encryptor only needs to solve the equations $QC_n(e, S)$ and $QC_n(a_i, S)$ for all $1 \leq i \leq \ell$. This means $\ell + 1$ equations instead of 2ℓ equations.

The algorithm proposed in [6] to find solutions to $QC_n(a, S)$ is quartic in the security parameter, making thus the $BasicIBE$ scheme more expensive than all standard IBE and PKE schemes. Designing a more efficient algorithm to find solutions to $QC_n(a, S)$ is an important open problem (see also [9]).

There were several attempts to improve the $BasicIBE$ scheme with respect to the number of congruential equations to be solved. However, none of them did it in a secure way. For details the reader is referred to [27, 21].

5. CONCLUSION

Looking back to the development of cryptography and its evolution to the modern stage, we find that the quadratic residuosity theory has had a major impact on the development of some key concepts. Thus, the probabilistic encryption and the semantic security have been exemplified by means of an encryption scheme that calls for quadratic residues. Also, the BBS generator is based on a permutation on the set of quadratic residues. The Cocks IBE scheme, based on quadratic residues, was among the first schemes to illustrate the possibility of identity-based encryption.

We believe that the quadratic residuosity theory has proven its potential for applicability in the field of cryptography. But, it seems that this potential has not been sufficiently explored. When we say this we refer mainly to high-order residues where cryptographic applications have only just begun to appear.

Acknowledgments. The author wishes to thank Dr. Răzvan Diaconescu for inviting him to give a talk on quadratic residuosity based cryptography at the *9th Congress of Romanian Mathematicians*, June 28 – July 3, 2019, Galaţi, Romania.

REFERENCES

[1] D. Aggarwal and U. Maurer, *Breaking RSA Generically Is Equivalent to Factoring.* In: A. Joux (Ed.)*, Advances in Cryptology – EUROCRYPT 2009*, pp. 36–53, Berlin, Heidelberg. Springer Berlin Heidelberg, 2009.

[2] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, *RSA and Rabin Functions: Certain Parts Are as Hard as the Whole.* SIAM J. Comput. **17** (Apr. 1988), *2*, 194–209.

[3] G. Ateniese and P. Gasti, *Universally Anonymous IBE Based on the Quadratic Residuosity Assumption.* In: *CT-RSA 2009.* Lecture Notes in Computer Science, Vol. 5473, pp. 32–47. Springer, 2009.

[4] L. Blum, M. Blum, and M. Shub, *Comparison of Two Pseudo-Random Number Generators.* In: D. Chaum, R. L. Rivest, and A. T. Sherman (Eds.), *Advances in Cryptology*, pp. 61–78. Boston, M.A. Springer US, 1983.

[5] L. Blum, M. Blum, and M. Shub, *A Simple Unpredictable Pseudo-random Number Generator.* SIAM J. Comput. **15** (1986), *2*, 364–383.

[6] D. Boneh, C. Gentry, and M. Hamburg, *Space-efficient Identity Based Encryption Without Pairings.* In: *FOCS 2007*, pp. 647–657. IEEE Computer Society, 2007.

[7] M. Clear, H. Tewari, and C. McGoldrick, *Anonymous IBE from Quadratic Residuosity with Improved Performance.* In: *AFRICACRYPT 2014.* Lecture Notes in Computer Science, Vol. 8469, pp. 377–397. Springer, 2014.

[8] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*. In: *IMACC 2001*. Lecture Notes in Computer Science, Vol. 2260, pp. 360–363. Springer, 2001.

[9] J. E. Cremona and D. Rusin, *Efficient Solution of Rational Conics*. Math. Comput. **72** (2003), *243*, 1417–1441.

[10] I. B. Damgård, *On The Randomness of Legendre and Jacobi Sequences*. In: S. Goldwasser (Ed.), *Advances in Cryptology — CRYPTO' 88*, pp. 163–172. Springer New York, 1990.

[11] S. Goldwasser and S. Micali, *Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information*. In: *STOC 1982*, pp. 365–377. ACM, 1982.

[12] S. Goldwasser and S. Micali, *Probabilistic Encryption*. Journal of Computer and System Sciences **28** (1984), 270–299.

[13] S. Goldwasser, *Cocks' IBE Scheme. Bilinear Maps*. MIT Lecture Notes: "6876: Advanced Cryptography", 2004.

[14] D. Hofheinz and E. Kiltz, *The Group of Signed Quadratic Residues and Applications*. In: S. Halevi (Ed.), *Advances in Cryptology – CRYPTO 2009*, pp. 637–653. Springer Berlin Heidelberg, 2009.

[15] T. Jager and J. Schwenk, *On the Analysis of Cryptographic Assumptions in the Generic Ring Model*. In: M. Matsui (Ed.), *Advances in Cryptology – ASIACRYPT 2009*, pp. 399–416. Springer Berlin Heidelberg, 2009.

[16] M. Joye, *Identity-Based Cryptosystems and Quadratic Residuosity*. In: *PKC 2016*. Lecture Notes in Computer Science, Vol. 9614, pp. 225–254. Springer, 2016.

[17] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. 2^{nd} edition. Chapman and Hall/CRC, 2014.

[18] M. B. Nathanson, *Elementary Methods in Number Theory*. Graduate Texts in Mathematics. Springer, 2000.

[19] A.-M. Nica and F. L. Ţiplea, *On Anonymization of Cocks Identity-based Encryption Scheme*. Computer Science Journal of Moldova **27** (2019), *3*, 283–298.

[20] M. O. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. Tech. rep. (1979), MIT.

[21] A. G. Schipor, *On the Security of Jhanwar-Barua Identity-Based Encryption Scheme*. In: J.-L. Lanet and C. Toma (Eds.), *Innovative Security Solutions for Information Technology and Communications*, pp. 368–375. Cham, Springer International Publishing, 2019.

[22] A. Shamir, *Identity-Based Cryptosystems and Signature Schemes*. In: *CRYPTO 1984*. Lecture Notes in Computer Science, Vol. 196, pp. 47–53. Springer, 1985.

[23] C. E. Shannon, *Communication theory of secrecy systems*. The Bell System Technical Journal **28** (1949), *4*, 656–715.

[24] V. Shoup, *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2008.

[25] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[26] G. Teşeleanu, F. L. Ţiplea, S. Iftene, and A.-M. Nica, *Boneh-Gentry-Hamburg's Identity-based Encryption Schemes Revisited.* In: 5^{th} *International Conference on Mathematical Foundations of Informatics (MFOI 2019)*, pp. 45–57. Iaşi (Romania), 2019.

[27] F. L. Ţiplea, S. Iftene, G. Teşeleanu, and A.-M. Nica, *Security of Identity-Based Encryption Schemes from Quadratic Residues.* In: *SECITC 2016.* Lecture Notes in Computer Science, Vol. 10006, pp. 63–77. Springer, 2016.

[28] F. L. Tiplea, S. Iftene, G. Teseleanu, and A.-M. Nica, *On the Distribution of Quadratic Residues and Non-residues Modulo Composite Integers and Applications to Cryptography.* Applied Mathematics and Computation **372** (2020).

[29] U. V. Vazirani and V. V. Vazirani, *Efficient And Secure Pseudo-Random Number Generation.* In: *25th Annual Symposium on Foundations of Computer Science,* 1984, pp. 458–463. IEEE Computer Society, 1984.

[30] A. C. Yao, *Theory and Application of Trapdoor Functions.* In: *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science* (SFCS 82), pp. 80–91. IEEE Computer Society, 1982.

Alexandru Ioan Cuza University of Iaşi
Department of Computer Science
Iaşi, Romania

and

Simion Stoilow Institute of Mathematics
of the Romanian Academy
Bucharest 010702, Romania
ferucio.tiplea@uaic.ro, fltiplea@gmail.com