Information & Security

N. Stoianov & A. Ivanov
vol. 47, no. 2 (2020): 249-260
https://doi.org/10.11610/isij.4717

Published since 1998

ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

Public Key Generation Principles Impact Cybersecurity

Nikolai Stoianov (✉), ***Andrey Ivanov***

Bulgarian Defence Institute, Sofia, Bulgaria, https://di.mod.bg/en

ABSTRACT:

Public key cryptography algorithms are based on number theory laws and principles. For every cryptography system one of the most important issues is the user's key which he/she uses to encrypt the messages. That is the reason the key generation process is always fundamental for data protection and, since cryptography takes up more space in our daily lives, the public key generation principles are so important. In this article the authors discuss the Miller–Rabin primality test in its relation to the key generation process.

ARTICLE INFO:

RECEIVED: 22 JUNE 2020
REVISED: 08 SEP 2020
ONLINE: 22 SEP 2020

KEYWORDS:

public key cryptography, Miller–Rabin primality test improvement, cybersecurity

Introduction

Nowadays the growing use of online communications over the Internet and the associated threats to the data we exchange, requires sufficient and reliable protection of the information exchanged. One of the most reliable and basic method to make information secure, when two communicating parties don't know each other, is public key cryptography. Hard-to-solve mathematical problems are used to realize the mathematical foundations of the existing algorithms using public key cryptography.[1, 2] In this mathematics, integer operations with large numbers are used and are based on modulo calculations of large prime numbers. Large prime numbers are also used to produce the user's cryptographic keys (public and private).

✉ Corresponding Author: n.stoianov@di.mod.bg

We could state that the security of the exchanged data protected by public key cryptography is due to two main facts: the difficulty of solving a mathematical algorithm and the reliability of the generated prime numbers used as keys in such a system. In this paper we will consider deterministic and probabilistic primality tests and will focus over the most widely used in practice an algorithm for testing prime numbers, that of Miller-Rabin [3, 4] and we will propose a new addition to it, which will increase the reliability of the estimation that this probabilistic algorithm gives.

Public Key Generation and The Importance of The Prime Numbers

Public key cryptography algorithms are based on two main things: difficult to solve mathematical problems and prime numbers with big values that serve as user private keys. If that prime numbers are generated not according to the prescribed rules or are not reliably confirmed as such, the security strength of protected data could be not enough. In an effort to ensure better protection of information, new algorithms and rules for generating private keys and the prime numbers involved in their compilation are created and proposed.[5, 6]

The more than 85% from certificate authorities (CA) based their root certificate security by using RSA encryption and signing scheme. Approximately of 10% of CA combinate both RSA and ECDSA cryptographic schemes to protect their public key infrastructure (PKI). This statement is based on our study in which we analysed the certificates stored into Windows, Android and Linux operating systems (OS) certificates stores. These operating systems are the most commonly used worldwide. We can say that a reliable estimate of divisibility of numbers is essential. Connected with this we will consider algorithms for primality testing. In practice, they are divided into two main types. Deterministic and probabilistic algorithms.

Deterministic Primality Testing

The most elementary approach to primality proving is trial division. If attempt to divide p by every integer $n \leq \lfloor \sqrt{p} \rfloor$ and no such n divides p, then p is prime. But this task will take $O\left(\sqrt{p}\, M(\log p)\right)$ time complexity, which is impractical for large values of p. That is why the most practical algorithms have to be used to deter the big numbers divisibility to factors.

An algorithm created in 2002, AKS (Agrawal, Kayal, and Saxena), falls into the group of tests that give an unambiguous assessment of divisibility of numbers. At the heart of AKS algorithm is Fermat's Little Theorem.

The Fermat's Little Theorem states that: if a number p is prime, $a \in Z, p \in N$ and $GCD(a, p) = 1$ then

$a^p \equiv a \; mod \; p$

The primality test by using this theorem fails for a specific class of numbers, known as pseudoprimes, which include the Carmichael numbers.

Primarily based on a polynomial generalization of the Fermat's Little Theorem the AKS algorithm state that: the number p is prime if and only

$$(x + a)^p \equiv (x^p + a) \; mod \; p$$

where $a \in Z, p \in N$.

The time complexity here would be $\Omega(n)$ which is not polynomial time. To reduce complexity, we can divide both sides by $(x^r - 1)$.[7] Therefore, for a chosen r the number of computations needed to be performed is less. Hence, the main objective now is to choose an appropriately small r and test if the equation:

$$(x + a)^p \equiv (x^p + a)\ mod\ GCD(x^r - 1, p)$$

is satisfied for sufficient number of a's.

The algorithm proposed by Agrawal, Kayal and Saxena [8,9] for primality testing has following steps:

(1) if $p = a^b$ for $a \in N, b > 1$ output ***COMPOSIT***
(2) find smallest r such that $O_r(p) > log^2 p$
(3) if $1 < GCD(a, p) < p$ for some $a \leq r$ then output ***COMPOSIT***
(4) if $n \leq r$ output ***PRIME***
(5) for each $a \in 1..\left\lfloor \sqrt{\varphi(p)} \log p \right\rfloor$
(6) if $(x + a)^p \neq (x^p + a)\ mod\ GCD(x^r - 1, p)$
(7) output COMPOSIT
(8) output PRIME

The complexity of execution of that algorithm is $\tilde{O}(log^{10.5} n)$ time. Hence the execution time will be proportional to $(\log n)^{10.5}$ if p grows larger. This is a polynomial time function, which although not as fast as the probabilistic tests used nowadays, has the advantage of being fully deterministic.

Probabilistic Testing of Prime Numbers. Miller-Rabin Primality Test

We know two mathematical ways to prove that a number p is composite:

a. number p factorization, where:

$$p = a.b \text{ and } a, b > 1 \qquad (2.2.1)$$

b. Exhibit a Fermat witness for p, i.e. find a number x satisfying:

$$x^{p-1} \not\equiv 1\ mod\ p \qquad (2.2.2)$$

The speed of these algorithms, which certainly determine whether a number is divisible, is unsatisfactory. This requires some of the probabilistic algorithms for primality test to be more widely used.

The Miller-Rabin [10,11] test is based on a third way to prove that a number is composite.

c. Exhibit a no square root of $1\ mod\ p$. That means to find a number x such that:

$$x^2 \equiv 1\ mod\ p \text{ and } x \not\equiv \pm 1\ mod\ p \qquad (2.2.3)$$

The Miller-Rabin test is the most widely used probabilistic primality test. This algorithm was proposed in 70's. Miller and Rabin gave two versions of the same algorithm to test whether a number p is prime or not. Rabin's algorithm works with a randomly chosen $x \in Z_p$, and is therefore a randomized one. Correctness of Miller's algorithm depends on correctness of Extended Riemann Hypothesis. In his test method it is need to tests deterministically for all x's, where $1 < x < 4.log^2 p$.

If x is a witness for an integer p, then p must be composite and we say that x witnesses the compositeness of p. Prime numbers clearly have no witnesses. If we picked up enough count of x's (100 or more depends on size of p) and no one is a witness, we can accept that number p is probably prime.
The algorithm realization steps are:

(1)	Denote $p-1 = s.2^m$, where $s \bmod 2 = 1$	
(2)	randomly picked up $x \in Z_p$	
(3)	if $x^{p-1} \not\equiv 1 \bmod p$ output ***COMPOSITE***	(p is definitely composite)
(4)	$b = x^s \bmod p$	
(5)	if $b \equiv \pm 1 \bmod p$, output ***PRIME***	(x is not a witness, p could be prime)
(6)	Loop i ∈0..m-1	
(7)	$b \leftarrow b^2 \bmod p$	
(8)	if $b \equiv -1 \bmod p$ then	
(9)	output ***PRIME***	(x is not a witness, p could be prime)
(10)	output ***COMPOSITE***	x is a witness p, is definitely not prime

The time complexity of that algorithm is $\tilde{O}(y.logn)$ where the y is the count of the iterations i.e. the different values of randomly chosen x.

Subgroup Extending by New Generating Number of a Ring is Gained.

In this part of the paper we will considering primality test based on two criteria and an idea of a method of transitioning (without intersection) or extending different multiplicative subgroups formed by their generator integer. To describe this method of transitioning/extending multiplicative subgroups formed by number p, we will use linear Diophantine equation:

$$d_x \,.\, d_y - p.k = d_z \qquad (2.3.1)$$

where $d_i = g^i \bmod p$. Every $d_i \in Z_p$ and it is part of a ring generated by number g and has ring order $\#O_{(g,p)}$ or smaller. In the particular case when $d_x = {d_y}^{-1} \bmod p$, value of $d_z = 1$.
In our practice dealing with number rings we saw that if $d_z = 1$ quadratic reciprocity $\left(\frac{d_x}{p}\right) = \left(\frac{d_y}{p}\right)$, and when size of $\#O_{(g,p)} < p-1$ then number k could highly has different quadratic reciprocity, i.e. $\left(\frac{d_x}{p}\right) \neq \left(\frac{k}{p}\right)$. In cases when that is not true, we can just do that with new value of $d_x \leftarrow k$ and in few steps of repeating that we can reach a value of k which has different quadratic reciprocity

to p than the initial d_x. We saw more important thing, that the rings formed with generators d_x and k, have very often different elements on its sets. If we use $q = d_x.k \; mod \; p$ as a ring generator its $\#O_{(q,p)}$ is different then $\#O_{(d_x,p)}$ and $\#O_{(k,p)}$. To show that we will use two examples. Into the first we will use a prime number with value 1117 and in the second one composite number $21421 = 11 \, . 1931$.

Example 1

$p = 1117$, $g = 430$, $\#O_{(p,p)} = 372$, $\left(\frac{430}{1117}\right) = -1$

1	430	**59**	688	**117**	654	**175**	823	**233**	870	**291**	275	**349**	440
2	595	**60**	952	**118**	853	**176**	918	**234**	1022	**292**	965	**350**	427
3	57	**61**	538	**119**	414	**177**	439	**235**	479	**293**	543	**351**	422
4	1053	**62**	121	**120**	417	**178**	1114	**236**	442	**294**	37	**352**	506
5	405	**63**	648	**121**	590	**179**	944	**237**	170	**295**	272	**353**	882
6	1015	**64**	507	**122**	141	**180**	449	**238**	495	**296**	792	**354**	597
7	820	**65**	195	**123**	312	**181**	946	**239**	620	**297**	992	**355**	917
8	745	**66**	75	**124**	120	**182**	192	**240**	754	**298**	983	**356**	9
9	888	**67**	974	**125**	218	**183**	1019	**241**	290	**299**	464	**357**	519
10	943	**68**	1062	**126**	1029	**184**	306	**242**	713	**300**	694	**358**	887
11	19	**69**	924	**127**	138	**185**	891	**243**	532	**301**	181	**359**	513
12	351	**70**	785	**128**	139	**186**	1116	**244**	892	**302**	757	**360**	541
13	135	**71**	216	**129**	569	**187**	687	**245**	429	**303**	463	**361**	294
14	1083	**72**	169	**130**	47	**188**	522	**246**	165	**304**	264	**362**	199
15	1018	**73**	65	**131**	104	**189**	1060	**247**	579	**305**	703	**363**	678
16	993	**74**	25	**132**	40	**190**	64	**248**	996	**306**	700	**364**	3
17	296	**75**	697	**133**	445	**191**	712	**249**	469	**307**	527	**365**	173
18	1059	**76**	354	**134**	343	**192**	102	**250**	610	**308**	976	**366**	668
19	751	**77**	308	**135**	46	**193**	297	**251**	922	**309**	805	**367**	171
20	117	**78**	634	**136**	791	**194**	372	**252**	1042	**310**	997	**368**	925
21	45	**79**	72	**137**	562	**195**	229	**253**	143	**311**	899	**369**	98
22	361	**80**	801	**138**	388	**196**	174	**254**	55	**312**	88	**370**	811
23	1084	**81**	394	**139**	407	**197**	1098	**255**	193	**313**	979	**371**	226
24	331	**82**	753	**140**	758	**198**	766	**256**	332	**314**	978	**372**	1
25	471	**83**	977	**141**	893	**199**	982	**257**	901	**315**	548	373	430
26	353	**84**	118	**142**	859	**200**	34	**258**	948	**316**	1070	374	595
27	995	**85**	475	**143**	760	**201**	99	**259**	1052	**317**	1013	375	57
28	39	**86**	956	**144**	636	**202**	124	**260**	1092	**318**	1077	376	1053
29	15	**87**	24	**145**	932	**203**	821	**261**	420	**319**	672	377	405

30	865	**88**	267	**146**	874	**204**	58	**262**	763	**320**	774	378	1015
31	1106	**89**	876	**147**	508	**205**	366	**263**	809	**321**	1071	379	820
32	855	**90**	251	**148**	625	**206**	1000	**264**	483	**322**	326	380	745
33	157	**91**	698	**149**	670	**207**	1072	**265**	1045	**323**	555	381	888
34	490	**92**	784	**150**	1031	**208**	756	**266**	316	**324**	729	382	943
35	704	**93**	903	**151**	998	**209**	33	**267**	723	**325**	710	383	19
36	13	**94**	691	**152**	212	**210**	786	**268**	364	**326**	359	384	351
37	5	**95**	8	**153**	683	**211**	646	**269**	140	**327**	224	385	135
38	1033	**96**	89	**154**	1036	**212**	764	**270**	999	**328**	258	386	1083
39	741	**97**	292	**155**	914	**213**	122	**271**	642	**329**	357	387	1018
40	285	**98**	456	**156**	953	**214**	1078	**272**	161	**330**	481	388	993
41	797	**99**	605	**157**	968	**215**	1102	**273**	1093	**331**	185	389	296
42	908	**100**	1006	**158**	716	**216**	252	**274**	850	**332**	243	390	1059
43	607	**101**	301	**159**	705	**217**	11	**275**	241	**333**	609	391	751
44	749	**102**	975	**160**	443	**218**	262	**276**	866	**334**	492	392	117
45	374	**103**	375	**161**	600	**219**	960	**277**	419	**335**	447	393	45
46	1089	**104**	402	**162**	1090	**220**	627	**278**	333	**336**	86	394	361
47	247	**105**	842	**163**	677	**221**	413	**279**	214	**337**	119	395	1084
48	95	**106**	152	**164**	690	**222**	1104	**280**	426	**338**	905	396	331
49	638	**107**	574	**165**	695	**223**	1112	**281**	1109	**339**	434	397	471
50	675	**108**	1080	**166**	611	**224**	84	**282**	1028	**340**	81	398	353
51	947	**109**	845	**167**	235	**225**	376	**283**	825	**341**	203	399	995
52	622	**110**	325	**168**	520	**226**	832	**284**	661	**342**	164	400	39
53	497	**111**	125	**169**	200	**227**	320	**285**	512	**343**	149	401	15
54	363	**112**	134	**170**	1108	**228**	209	**286**	111	**344**	401	402	865
55	827	**113**	653	**171**	598	**229**	510	**287**	816	**345**	412	403	1106
56	404	**114**	423	**172**	230	**230**	368	**288**	142	**346**	674	404	855
57	585	**115**	936	**173**	604	**231**	743	**289**	742	**347**	517	405	157
58	225	**116**	360	**174**	576	**232**	28	**290**	715	**348**	27	406	490

If we use $x = 2$, $d_x = 595$, $\left(\frac{595}{1117}\right) = 1$, $d_y = 595^{-1}\ mod\ 1117 = 811$ and calculate: $k = \frac{d_x .\ d_y - 1}{p} = \frac{595 .\ 811 - 1}{1117} = 432$. The generated ring will be:

1	432	**59**	281	**117**	1080	**175**	144	**233**	466	291	360	349	48
2	85	**60**	756	**118**	771	**176**	773	**234**	252	292	257	350	630
3	976	**61**	428	**119**	206	**177**	1070	**235**	515	293	441	351	729
4	523	**62**	591	**120**	749	**178**	919	**236**	197	294	622	352	1051
5	302	**63**	636	**121**	755	**179**	473	**237**	212	295	624	353	530

6	892	**64**	1087	**122**	1113	**180**	1042	**238**	1107	296	371	**354**	1092
7	1096	**65**	444	**123**	506	**181**	1110	**239**	148	297	541	**355**	370
8	981	**66**	801	**124**	777	**182**	327	**240**	267	298	259	**356**	109
9	449	**67**	879	**125**	564	**183**	522	**241**	293	299	188	**357**	174
10	727	**68**	1065	**126**	142	**184**	987	**242**	355	300	792	**358**	329
11	187	**69**	993	**127**	1026	**185**	807	**243**	331	301	342	**359**	269
12	360	**70**	48	**128**	900	**186**	120	**244**	16	302	300	**360**	40
13	257	**71**	630	**129**	84	**187**	458	**245**	210	303	28	**361**	525
14	441	**72**	729	**130**	544	**188**	147	**246**	243	304	926	**362**	49
15	622	**73**	1051	**131**	438	**189**	952	**247**	1095	305	146	**363**	1062
16	624	**74**	530	**132**	443	**190**	208	**248**	549	306	520	**364**	814
17	371	**75**	1092	**133**	369	**191**	496	**249**	364	307	123	**365**	910
18	541	**76**	370	**134**	794	**192**	925	**250**	868	308	637	**366**	1053
19	259	**77**	109	**135**	89	**193**	831	**251**	781	309	402	**367**	277
20	188	**78**	174	**136**	470	**194**	435	**252**	58	310	529	**368**	145
21	792	**79**	329	**137**	863	**195**	264	**253**	482	311	660	**369**	88
22	342	**80**	269	**138**	855	**196**	114	**254**	462	312	285	**370**	38
23	300	**81**	40	**139**	750	**197**	100	**255**	758	313	250	**371**	778
24	28	**82**	525	**140**	70	**198**	754	**256**	175	314	768	**372**	996
25	926	**83**	49	**141**	81	**199**	681	**257**	761	315	27	**373**	227
26	146	**84**	1062	**142**	365	**200**	421	**258**	354	316	494	**374**	885
27	520	**85**	814	**143**	183	**201**	918	**259**	1016	317	61	**375**	306
28	123	**86**	910	**144**	866	**202**	41	**260**	1048	318	661	**376**	386
29	637	**87**	1053	**145**	1034	**203**	957	**261**	351	319	717	**377**	319
30	402	**88**	277	**146**	1005	**204**	134	**262**	837	320	335	**378**	417
31	529	**89**	145	**147**	764	**205**	921	**263**	793	321	627	**379**	307
32	660	**90**	88	**148**	533	**206**	220	**264**	774	322	550	**380**	818
33	285	**91**	38	**149**	154	**207**	95	**265**	385	323	796	**381**	404
34	250	**92**	778	**150**	625	**208**	828	**266**	1004	324	953	**382**	276
35	768	**93**	996	**151**	803	**209**	256	**267**	332	325	640	**383**	830
36	27	**94**	227	**152**	626	**210**	9	**268**	448	326	581	**384**	3
37	494	**95**	885	**153**	118	**211**	537	**269**	295	327	784	**385**	179
38	61	**96**	306	**154**	711	**212**	765	**270**	102	328	237	**386**	255
39	661	**97**	386	**155**	1094	**213**	965	**271**	501	329	737	**387**	694
40	717	**98**	319	**156**	117	**214**	239	**272**	851	330	39	**388**	452
41	335	**99**	417	**157**	279	**215**	484	**273**	139	331	93	**389**	906
42	627	**100**	307	**158**	1009	**216**	209	**274**	847	332	1081	**390**	442
43	550	**101**	818	**159**	258	**217**	928	**275**	645	333	86	**391**	1054

44	796	**102**	404	**160**	873	**218**	1010	**276**	507	334	291	**392**	709
45	953	**103**	276	**161**	707	**219**	690	**277**	92	335	608	**393**	230
46	640	**104**	830	**162**	483	**220**	958	**278**	649	336	161	**394**	1064
47	581	**105**	3	**163**	894	**221**	566	**279**	1	337	298	**395**	561
48	784	**106**	179	**164**	843	**222**	1006	280	432	338	281	**396**	1080
49	237	**107**	255	**165**	34	**223**	79	281	85	339	756	**397**	771
50	737	**108**	694	**166**	167	**224**	618	282	976	340	428	**398**	206
51	39	**109**	452	**167**	656	**225**	13	283	523	341	591	**399**	749
52	93	**110**	906	**168**	791	**226**	31	284	302	342	636	**400**	755
53	1081	**111**	442	**169**	1027	**227**	1105	285	892	343	1087	**401**	1113
54	86	**112**	1054	**170**	215	**228**	401	286	1096	344	444	**402**	506
55	291	**113**	709	**171**	169	**229**	97	287	981	345	801	**403**	777
56	608	**114**	230	**172**	403	**230**	575	288	449	346	879	**404**	564
57	161	**115**	1064	**173**	961	**231**	426	289	727	347	1065	**405**	142
58	298	**116**	561	**174**	745	**232**	844	290	187	348	993	**406**	1026

It is easy to see that two rings have no common elements among their groups. But if we calculate $q = d_x.k \; mod \; p = 430 \; .432 \; mod \; 1117 = 338$ and construct a ring with generator q, that ring will has order $\#O_{(q,p)} = 1116 = p - 1$.

To demonstrate that this works when the number p is composite we make an example with $p = 21421$.

Example 2

$$p = 21421 \; , \; g = 430 \; , \; \#O_{(p,p)} = 690 \; , \; \left(\frac{430}{21421}\right) = 1$$

Use $x = 1$, $d_x = 430$, $d_y = 430^{-1} \; mod \; 21421 = 2441$

calculate: $k = \frac{d_x \; . \; d_y - 1}{p} = \frac{430 \; . \; 2441 - 1}{21421} = 49.$

If you try to make tables as this above and use this two generator numbers (430 and 49) you will see that they have no common elements too. More over if you calculate $q = d_x.k \; mod \; p = 430 \; .49 \; mod \; 21421 = 21070$ you will be convinced that ring with generator q will not have intersection elements neither with ring R_g nor with ring R_k.

That is our idea to use a start generate number g and next use several conscience values of k in two primality test criteria and not to be used randomly generated values. This can lead to make the test more deterministic due to more subgroups could be tested and pass the criteria.

The criteria of the test which we will use are:

(1) If p is a prime, then Jacobi's symbol is equal to Legendre's symbol.[12]

(2) The Miller-Rabin test base: To find a number x such that:

$$x^2 \equiv 1 \; mod \; p \text{ and } x \not\equiv \pm 1 \; mod \; p$$

We will describe the algorithm steps which we propose. To do that we will first point out the steps of 3 basic functions which we use into it:

function modK (in generator, in modulonumber)

```
a = (generator ^ -1) mod modulonumber
return ( (a * generator) – 1 ) div modulonumber
```

function getNotPRU (in generator, in modulonumber)

```
g = generator
kJacobi = JacobiSimbol(g,m)
if (kJacobi = -1)
  g = (g * g) mod modulonumber
i = 0
m = g
DO
  i++
  m = modulonumber – m
  m = modK(m, modulonumber)
  if (m<2)
     m = modulonumber div 3
     i = 0
  kJacobi = JacobiSimbol(m,modulonumber)
  if (kJacobi=0)
     return m
WHILE (kJacobi=1) or (i>33)
return m
```

function TryToGetPRU (in generator, in modulonumber)

```
g = generator
kJacobi = JacobiSimbol(g,m)
if (kJacobi = -1)
   g = (g * g) mod modulonumber
e = modulonumber mod 5
if (e in [2,3])
   return = (5*g) mod modulonumber
e = modulonumber mod 6
if (e = 5)
```

```
        return = (-3*g) mod modulonumber
    e = modulonumber mod 8
    if (e in [3,5])
        return = (2*g) mod modulonumber
    if (e = 7)
        return = (-2*g) mod modulonumber
    i = 0
    m = g
    DO
        i++
        m = modulonumber – m
        m = modK(m, modulonumber)
        if (m<2)
            m = modulonumber div 3
            i = 0
        kJacobi = JacobiSimbol(m,modulonumber)
        if (kJacobi=0)
            return m
    WHILE (kJacobi=-1) or (i>33)
    if ((m*g) mod modulonumber = 1)
        return m
    return (m*g) mod modulonumber
```

The steps of the algorithm which we propose to be estimated primality of a number are:

(1) Pick up small prime number q such that $a = p \bmod q > 1$

(2) $b = a^{p-1} \bmod p$

(3) if $b > 1$

(4) output ***COMPOSITE***

(5) $JacobiSymbol = 1$

(6) $f = (p - 1)$

(7) $loop\ i \in 0..4$

(7.1) $b = f\ /\ 2$

(7.2) $if\ JacobiSymbol = 1$

(7.3) $g = TryToGetPRU\ (a, p)$

(7.4) $if\ JacobiSymbol = -1$

(7.5) $g = getNotPRU\ (a, p)$

(7.6) $JacobiSymbol \leftarrow \left(\frac{g}{p}\right)$

(7.7) $if\ JacobiSymbol = 0$

(7.8) output ***COMPOSITE***

(7.9) $LegendreSymbol \leftarrow g^b\ mod\ p$

(7.10) $if\ \ JacobiSymbol <> LegendreSymbol$

(7.11) output ***COMPOSITE***

(7.12) $while\ (LegendreSymbol = 1)\ and\ (b\ mod\ 2 = 0)$

(7.12.1) $b \leftarrow b\ /\ 2$

(7.12.2) $LegendreSymbol \leftarrow g^b\ mod\ p$

(7.12.3) $if\ (LegendreSymbol > 1)\ and\ (LegendreSymbol < f)$

(7.12.4) output ***COMPOSITE***

(7.13) $a \leftarrow [g * modfK(g,p)]\ \ mod\ p$

(7.14) $if\ a = 1\ then$

(7.15) $a \leftarrow g$

(8) output ***PRIME***

The proposed algorithm was used to test first 50th millions of prime numbers and test passed successfully. We made some test with several hundreds of big numbers with size of 200, 300 and 400 decimal digits and primality test estimation was correct.

For comparison between new proposal method which can be used as extension of Miller-Rabin primality test and AKS, we can say that AKS is deterministic primality test algorithm and new addition of Millar-Rabin algorithm stays probabilistic one but highly increases possibility of correct estimation to check primality of a number with only 5 main iterations, whatever the size of tested number is. The power of that new approach is in that different sets of number rings without elements intersections between them are used in primality check process. From other comparison point of view, we can say that AKS algorithm in general operates with polynomial mathematical operations that is way Millar-Rabin algorithm is faster and remains most commonly used in worldwide public key cryptographic systems. That is why we directed our efforts to gain a way to increase the probability of the correct result of Miller-Rabin primality test algorithm.

Conclusions

Into suggested method of switching between different subgroups there is no need to use randomly chosen values of integers which have to be used to pass the test criteria. Smaller number of test integers are need to execute the test.

If the suggested method in this paper could be improved it is highly possible that the Miller-Rabin primality test could be gained to deterministic one with low complexity.

References

1 William Stallings, *Cryptography and Network Security: Principles and Practice*, 5th. ed. (Prentice Hall Press, 2010).

2 Ann Murphy and David Murphy, "The Role of Cryptography in Security for Electronic Commerce," *The ITB Journal* 2, no. 1, (2001): 21-50.

3 Moses Liskov, "Miller–Rabin Probabilistic Primality Test," In: van Tilborg H.C.A. (eds.) *Encyclopedia of Cryptography and Security* (Boston, MA: Springer, 2005).

4 Keith Conrad, "Miller-Rabin Test," *Encyclopedia of Cryptography and Security* (Boston, MA: Springer, 2011).

5 Fanyu Kong, Jia Yu, and Lei Wu, "Security Analysis of an RSA Key Generation Algorithm with a Large Private Key," In *Proceedings of the 14th international conference on Information security (ISC'11)* (Berlin, Heidelberg, Springer-Verlag, 2011), 95–101.

6 Luis Hernandez Encinas, Jaime Masqué, and Araceli Queiruga-Dios, "An Algorithm to Obtain an RSA Modulus with a Large Private Key," IACR Cryptology ePrint Archive, 2003, p. 45.

7 Vijay Menon, "Deterministic Primality Testing – Understanding the AKS Algorithm," 2013.

8 Robert G. Salembier and Paul Southerington, "An Implementation of the AKS Primality Test," IEEE, May 12, 2005.

9 Martin Dietzfelbinger, *Primality Testing in Polynomial Time: From Randomized Algorithms to "Primes Is in P,"* Lecture Notes in Computer Science, (SpringerVerlag, 2004).

10 Bulat Mubarakov and Ramilya Rubtsova, "On the Number of Witnesses in the Miller–Rabin Primality Test," *Symmetry* 12, no. 6 (2020): 890, https://doi.org/10.3390/sym12060890.

11 Keith Conrad, "The Miller–Rabin Test," 2011, http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/millerrabin.pdf.

12 W. Sierpiński, "Chapter IX Legendre's Symbol and Jacobi's Symbol," *North-Holland Mathematical Library* 31 (1988): 340-359.

About the Authors

Associate Professor Nikolay **Stoyanov**, PhD, is Deputy Director of the Bulgarian Defence Institute "Professor Tsvetan Lazarov." In 2003 he acquired a doctoral degree in the Advanced Defence Research Institute, "G. S. Rakovski" National Defence College. In 2020 he graduated the "G. S. Rakovski" National Defence College with specialty "Strategic Management of Defence and the Armed Forces." https://orcid.org/0000-0002-4953-4172

Andrey **Ivanov** is an engineer. In 2000, he graduated the Higher Military Artillery School "P. Volov" – Shumen with a degree in automated management systems. https://orcid.org/0000-0003-4466-9569