# Cryptographic Vulnerabilities in Real-Life Web Servers

Eman Salem Alashwali
College of Computing & IT
King Abdulaziz University
Jeddah, Saudi Arabia
ealashwali@kau.edu.sa

***Abstract*—This paper examines the security of real-life Internet servers using the most popular Secure Socket Layer (SSL) protocol to ensure secure connections. We concentrate on Rivest-Shamir-Adleman (RSA) public-key vulnerabilities which result from the initial settings of web servers. We look at the question of breaking individual RSA keys. The possibility of factoring RSA keys used by real web servers on the Internet has been a disturbing discovery which received a lot of press in the recent months. We have conducted an Internet scan with a particular focus on commercial websites (.com and .co domains). We have created a database containing over 3 million certificate chains together with detailed information about each website, its security settings, geographic location and other relevant data. This allowed us to see how different key sizes are adopted, how many servers are using weak keys and which countries are quicker to adopt secure keys. We attempted to factor all keys we were able to collect from our scan and from another public database. The method to achieve this seemed trivial at first, but it can only be done efficiently by using a special algorithm proposed by Bernstein. We ran the computation based on an open implementation of Bernstein's algorithm. We have been able to factor few thousands keys. The infected servers we inspected appear as Embedded Web Servers (EWS). Although we have not yet found any immediate threats to e-commerce websites, the risks that such vulnerable servers present should not be underestimated as they can be exploited to perform different types of attacks, including Denial of Service (DoS), corporate espionage and firmware modification.**

***Keywords— public key; cryptography; encryption; information security.***

## I. INTRODUCTION

### A. Our Motivation: How Secure Is the Internet?

The Internet has become an integral part of everyone's daily life. It has been reported that, as of September 2009, 25.6% of the world's population uses the Internet [1], and usage is rapidly increasing. Between 2000 and 2009 alone, Internet usage increased by 380% [1]. The number of registered domain names is also rapidly increasing. In the third quarter of 2012, VeriSign has reported that .com domain names have exceeded 100 million [2].

With the growing number of users and the value of the data being transferred over the Internet, the number of threats and attacks has also increased. Therefore, Internet security has become essential. When clients make an online transaction, they require a secure connection that ensures that their data are not intercepted; that they are connecting to the genuine website; and finally, that their data are not tampered with [3]. These requirements can respectively be restated as confidentiality, authenticity and integrity.

Today, Secure Socket Layer/Transport Layer Security (SSL/TLS) is one of the most important network security protocols used to secure the Internet. It is designed precisely to address the aforementioned concerns [3]. SSL provides confidentiality, authenticity and integrity by employing encryption, digital signatures and hash functions.

However, SSL is only a communication protocol, and it has several limitations [3]. One of the limitations is that it relies on different cryptographic algorithms to provide security services [3]. As a result, SSL is only as strong as the cryptographic algorithms it employs [3]. For example, SSL can not secure a transaction that uses the broken DES algorithm. Another limitation arises from the implementation of SSL [3]. For instance, if the key size is not sufficient, or if the system's Random Number Generator (RNG) is faulty, SSL provides no security.

Recent studies by Lenstra *et al.* [4], and Heninger *et al.* [5] have revealed alarming findings about factorable keys used in SSL servers. Heninger *et al.* [5] concluded that the root cause of such vulnerable keys is not a cryptographic issue, but rather an improper implementation [5].

It is thus very important for security researchers to continuously evaluate the real-world implementation of any security protocol. Such evaluations help by spotting weaknesses and providing solutions and recommendations. Toward this end, this paper aims to shed light on the importance of proper implementation for SSL protocol. We aim to raise awareness about the recently discovered prevalence of factorable RSA keys on the Internet by [4] and [5].

The rest of the paper is organized as follows: In section 2, we provide background on the SSL/TLS[1] protocol and RSA cryptosystem. In section 3, we summarize some major studies related to our topic. In section 4, we briefly describe our methodology. Section 5 presents and discusses our results. Section 6 outlines the limitations of our study, while section 7 provides a defensive solution. We conclude in section 8.

[1]For brevity, we will use the term SSL to refer to either SSL or TLS.

## II. BACKGROUND

### A. SSL, an Overview

SSL is a network security protocol that was originally developed by Netscape in 1994 [3]. In 1996, the responsibility of developing SSL was passed on to the Internet Engineering Task force (IETF) [3]. IETF released TLSv1 that was based on SSLv3 [3]. SSL is used to secure Hypertext Transfer Protocol (HTTP) in order to provide HTTP over SSL (HTTPS).

SSL has two main stages and is divided into two sub-protocols: the SSL Handshake Protocol and the SSL Record Protocol [6]. The SSL Handshake Protocol is responsible for negotiating the ciphersuite (cryptographic algorithms and key lengths) options between the client and the server and allows one or both parties to authenticate the other [6]. The main stages of a typical handshake protocol can be described as follows [3]: First, the client sends a *ClientHello* message to request the server to begin SSL negotiation. The message includes a list of ciphersuites proposed by the client. Second, the server responds with a *ServerHello* message. In this message, the server decides which ciphersuite will be used during the communication. Third, in order for the server to authenticate its identity, it sends a *Certificate* message, which contains its public-key information. Fourth, the server sends a *ServerHelloDone* message to confirm to the client that it has finished its part of the negotiation (the *ServerHello* phase). Then the server waits for the client response. Fifth, the client sends a *ClientKeyExchange* message. It contains the symmetric key information that both parties are going to use. Sixth, the client sends a *Finished* message to the server to inform it that the negotiation process has been completed successfully. Finally, similar to the previous step, the server sends a *Finished* message to the client for the same purpose.

### B. RSA Cryptosystem

RSA is one of the earliest known public-key cryptosystems, invented in the 1970s [7]. RSA allows two major types of applications: encryption and digital signature [7].

*1) RSA Setup:* In order to encrypt or sign with RSA, the message recipient or the signer of the outgoing message needs to set up his RSA private-key. This is done once for all messages and must be done prior to any actual communication using RSA. The keys are set up as follows. First, compute $N$ (the modulus), which is a product of two large prime numbers, $p$ and $q$, such that [7]:

$$N = p \times q \quad (1)$$

These two primes are randomly chosen [7]. Second, one needs to choose an encryption exponent $e$, which has to be relatively prime to both $(p-1)$ and $(q-1)$ [8]. This means that:

$$GCD\ (e, (p-1) \times (q-1)) = 1 \quad (2)$$

Finally, one can compute another exponent, called the decryption exponent $d$, from the previous values of $p$, $q$, and $e$, such that:

$$d = e^{-1} \bmod (p-1) \times (q-1) \quad (3)$$

These three numbers, $p$, $q$, and $d$, must remain confidential at all times. The RSA private key or decryption key is the pair $(d, N)$ [7] [8]. The user publishes the modulus $N$ and the public exponent $e$ [7] [8]. Hence, the public key or the encryption key is the pair $(e, N)$ [7] [8]. For more details about RSA cryptosystem and RSA key generation, refer to [7] [8].

*2) RSA in SSL:* In SSL, RSA is used in one of two cases [5]:

- If the key-exchange protocol is RSA: RSA is used to encrypt a session key chosen by the client.
- If the key exchange is Diffie-Hellman (DH): RSA can be used to provide a digital signature in order to authenticate messages exchanged during the DH key agreement process.

RSA has been the most widely used key exchange protocol for so many years. Also, it is the dominant signature scheme used to authenticate DH key-exchange messages. In 2006, Lee *et al.* examined over 19,000 SSL servers and showed that 99.86% of the servers support RSA key exchange protocol compared to 57.57% of the servers supporting DH protocol with RSA signature, while only 0.02% support DH with Digital Signature Standard (DSS) [9]. In 2011, Holz *et al.* measured the chosen ciphersuites from real SSL sessions in two different runs. In General, the results show that of the top 10 ciphersuites chosen by the servers, DH key-exchange with RSA signature appeared only in 3 ciphers, chosen by less than 30% of the servers [10]. On the other hand, RSA key-exchange is used in the rest of the top 10 chosen ciphersuites, representing the majority [10]. The principal role that RSA plays in SSL has motivated our choice in this paper to concentrate on RSA-key vulnerabilities in SSL servers.

## III. RELATED WORK

In 2012, Lenstra *et al.* conducted a study that questioned the validity of the assumption that distinct random numbers are generated for every key [4]. In their dataset that contained 6.38 million distinct RSA moduli, they could factor 12,934 of them (0.2% of the total) [4]. They concluded that keys generated using "single-secret" algorithms based on DH such as ECDSA and ElGamal are more secure than keys generated using "multiple-secret" algorithms such as in RSA [4].

Concurrent with the previous study, Heninger *et al.* conducted the largest ever Internet scan on the public IPv4 address space for hosts listening on ports 443 and 22 (SSL and SSH hosts) [5]. They factored 16,717 distinct moduli affecting 64,081 SSL hosts (0.50% of the SSL hosts in their scan) [5]. They did a comprehensive analysis to identify the problem and concluded that the main reason for widespread factorable keys is low entropy due to faulty implementation, not a cryptographic issue as Lenstra *et al.* concluded [5].

## IV. METHODOLOGY

Our methodology can be described in four main steps: first, we built a list of domain names. Since our target is commercial servers, we looked for domain names that end with .com.* and .co.* Top Level Domains (TLDs) where the * refers to the country code (cc) if applicable (for brevity we will use the terms .com and .co for the rest of the paper, but our dataset includes ccTLDs as well as generic TLDs). We obtained a giant domain names list from a public source. At the moment, we are reluctant to publish the source to avoid liabilities in helping attacks. From this list, we extracted the distinct .com and .co domain names without taking any sub-domain names (e.g. example.com). Second, we conducted an Internet scan to extract SSL certificates. We performed the scan between the end of November and the beginning of December 2012. For

every server in our list, we did the following: we created an SSL socket on port 443; initiated a handshake; extracted the whole certificate chain; stored each certificate; and finally, parsed it to extract its attributes such as the certificate issuer, subject, public key etc. Third, we retrieved the geolocation information for the servers that successfully completed an SSL handshake in the previous step. We achieved this through commercial services that maintain IP geolocation databases. The principal source was an on-line service provided by [11] while a downloadable database from [12] used for locating 63 IPs that the first provider failed to locate. The accuracy of such databases is questionable [13]. However, they can be considered accurate at the country level [13] which is the level we used in our study. We achieved step 2 and 3 programmatically using Java programming and MySQL relational database (part of step 3 code provided by [11]). We employed java multi-threaded programming in order to run several tasks simultaneously and collect the data in a reasonable time. Fourth, we computed the Greatest Common Divisor *(GCD)* for all distinct moduli pairs we could collect. In order to factor RSA keys efficiently, we ran an open source implementation for a quasilinear *GCD* computing algorithm based on the Bernstein algorithm [14] and provided by [15]. We used an Amazon EC2 instance running Ubuntu12 x86_64 with 34.2 GB of RAM. We also took research ethics into consideration. All the data we used are publicly available. We did not publish any affected website addresses, device vendors or keys we have factored.

## V. RESULTS AND DISCUSSION

In this section, we will present and analyze our results. Our study is concerned with two issues in RSA keys: key lengths and randomness. In our statistics, we used the distinct keys only. We use the term SSL-server or simply, server to refer to an SSL-enabled distinct domain name.

### A. Our Scan Results in a Nutshell

From the original domain names set, we extracted 6,248,784 distinct domain names registered to .com and .co TLDs. From those, we found 1,713,388 (27.42% of the original .com and .co set) SSL-enabled servers. Of those, 1,713,291 returned one or more certificate. Only 523,225 IPs were behind all of the SSL servers that returned one or more certificate. In terms of SSL certificates, our scan collected a total of 3,543,291 X.509 SSL certificate chains. Of those, there are 1,713,291 leaf certificates. We found a total of 385,140 distinct RSA public keys used in all the certificates we collected (99.45% of these keys are in the leaf certificates).

### B. Weak Keys

An RSA public key consists of the pair *(N, e)*: the modulus *N* and the public exponent *e*. The key length/size refers to the size of the modulus *N* in bits [16]. It is an important aspect in the security of any public-key cryptosystem. The longer the key, the more security it provides [16]. On the other hand, longer keys result in slower computation [16]. Keys with lengths of 512-bit and 768-bit have already been factored by academic researchers in 1999 and 2010 respectively [16] [17]. Therefore, keys with length ≤ 768-bit should not be used. According to the National Institute of Standards and Technology (NIST) recommendations, the security strength of 1024-bit keys (comparable to 80-bit security strength) in encryption is "deprecated" (i.e. can be used with possible risk) from the year 2011 to 2013 and "disallowed" by 2014 [18]. Lenstra's key-length recommendations which are based on mathematical equations suggest that 1024-bit keys can provide sufficient security until 2006 [19] and that the minimal key length for sufficient security until 2012 should be 1229-bit [20]. In August 2012, Microsoft made positive steps toward pushing servers' administrators to select sufficient key lengths. Microsoft has released a Windows update that will block applications using RSA keys < 1024-bit [21]. For example, after the update, Internet Explorer will not open any website that offers an RSA certificate with key < 1024-bit [21].

Our results show that 0.3% of the keys have lengths < 1024-bit, providing inadequate security. Table I summarizes our findings of key sizes which may be considered weak in the year 2013.

Weak keys are more dangerous when used in intermediate or root certificates since they sign other entities' keys. In non-leaf certificates, we found 14 keys ≤ 512-bit affecting 18 servers, 16 keys < 1024-bit affecting 413 servers and 768 keys ≤ 1024-bit affecting 135,981 servers.

### C. Key Length Practices and IP Geolocations

In this section, we want to answer two questions: 1) are strong keys adopted in certain countries faster than others? and 2) are weak keys more frequent in certain countries than others? In this section, our definition for weak keys is a lower bound. Today, a 512-bit key can be factored by an average adversary using typical hardware [5]. Therefore, we define weak keys as those with lengths ≤ 512-bit. We also defined strong keys as those with lengths ≥ 2048-bit (per NIST's recommendations [20]). To have more reliable statistics, we limited our comparisons to the 10 countries which showed the highest number of distinct moduli.

Our results show that the United Kingdom (UK) is the top country in terms of adopting strong keys. In terms of weak keys, Italy showed the highest percentage, followed by France, while the Netherlands showed the lowest percentage. It is worth noting that we recompiled the statistics for keys < 1024-bit and the countries' order remained identical to that of keys ≤ 512-bit except in Germany and Japan, they showed equal percentages in keys < 1024-bit (0.23%). Table II summarizes our findings. These results could be studied in light of government actions to improve cyber security, in particular, what various governments' security services have recommended for the industry. For example, the French Network and Information Security Agency (FNISA) has set 2048-bit as a minimal key length since 2010 [20]. However, there is no evidence that the industry is following these recommendations.

TABLE I. CUMULATIVE REPRESENTATION FOR WEAK KEYS

| **Length** | **# and % of Weak Keys and The Affected Leaves** | | |
|---|---|---|---|
| | ***# keys*** | ***% keys*** | ***# affected servers (leaves)*** |
| ≤ 512-bit | 1,082 | 0.28% | 5,003 |
| ≤ 768-bit | 1,126 | 0.29% | 5,532 |
| < 1024-bit | 1,140 | 0.30% | 5,552 |
| ≤ 1024-bit | 89,978 | 23.36% | 406,698 |

TABLE II. WEAK VS. STRONG KEYS IN THE TOP 10 COUNTRIES

| Country | Distinct Keys | | | | |
|---|---|---|---|---|---|
| | *Total* | *# Strong* | *% Strong* | *# Weak* | *% Weak* |
| **US** | 247,307 | 196,250 | **79.35%** | 582 | 0.24% |
| **UK** | 34,537 | 27,486 | **79.58%** | 89 | 0.26% |
| Germany | 16,360 | 11,590 | 70.84% | 36 | 0.22% |
| Canada | 12,100 | 8,855 | 73.18% | 37 | 0.31% |
| **Australia** | 11,587 | 9,204 | **79.43%** | 35 | 0.30% |
| Japan | 10,164 | 7,802 | 76.76% | 23 | 0.23% |
| **France** | 7,032 | 4,874 | 69.31% | 50 | **0.71%** |
| Netherlands | 4,924 | 3,002 | 60.97% | 6 | 0.12% |
| Spain | 3,856 | 2,763 | 71.65% | 14 | 0.36% |
| **Italy** | 3,048 | 1,839 | 60.33% | 22 | **0.72%** |

### *D. Can RSA Keys Be Broken?*

RSA security is based on the difficulty of factoring [22] [9]. To date, nobody has yet been able to factor a 1024-bit modulus [5]. The largest modulus that has been shown to be actually factored is 768-bit [5] [17]. However, there is a known vulnerability that leads to factoring a 1024-bit RSA modulus [5]. It can be exploited if an adversary can find a pair of moduli that share a prime factor [5]. The recent discovery of the widespread factorable RSA moduli on SSL hosts on the Internet by [4] and [5] was achieved by exploiting this vulnerability.

*1) Computing the Shared Prime Leads to the Private Key:* The shared prime between two distinct moduli $N_1$ and $N_2$ can be found by computing the *GCD* of the two moduli, i.e. $GCD\ (N_1, N_2)$ [5]. If the *GCD* is not equal to 1, this means that both moduli share a prime factor $p$ [5]. Therefore, the second prime $q$ for both moduli can be computed [5]. This in turn leads to computing the private key $d$ for both moduli using equation 3 in section II [5]. Figure 1 illustrates the vulnerability.

*2) Computing the GCD Using the Naïve Method:* In theory, anyone can collect public keys from the Internet and factor some of these keys by computing pair-wise *GCDs*. In practice, this is not feasible for large sets. Even with a very fast *GCD*, it is not conceivable for an ordinary adversary to compute the *GCD* for all of the public keys he may be able to collect. We analyze the complexity of the computation using a similar approach to [5] using our own measurements and datasets. In our case, the execution time for a single *GCD* computation in a modern PC with an Intel Xeon 3.4 GHz processor and 16-GB RAM, using the GNU Multiple Precision (GMP) arithmetic library for two distinct 2048-bit moduli (the majority representation of key lengths in our set) took 0.029 milliseconds (ms). The *GCD* for a pair of numbers can be computed with complexity $O(n^2)$ for n-bit numbers [5] [23]. This would cost $O((mn)^2)$ to compute all pair-wise *GCDs* [23]. We computed the total time complexity as:

$$Complexity = T \times \binom{n}{2} \qquad (6)$$

where $n$ is the number of distinct keys and $T$ is the execution time for computing the *GCD* of a pair of moduli. Therefore, computing pairwise *GCDs* for our set of 385,140 distinct keys would take approximately 26 days (assuming the average key length is 2048-bit). We could run the computation for our (smaller) set. But since we were looking for more factorable keys, we combined our set with the Electronic Frontier Foundation (EFF) set [24] and this resulted in a giant set that would take around 4.4 years (assuming the average key length is 1024-bit). However computing *GCDs* can be further improved considerably.

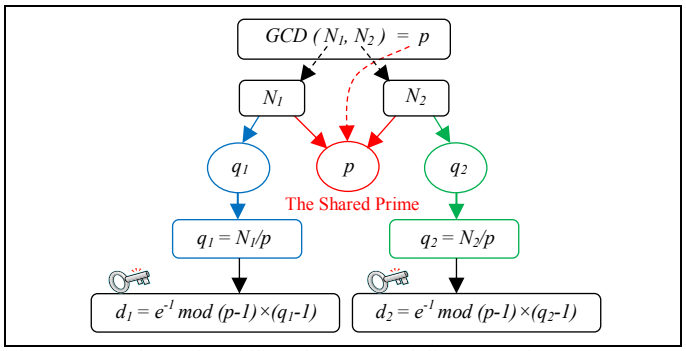

Figure 1. RSA-key vulnerability when two moduli share a prime

*3) Computing the GCD Using Efficient GCD:* For a more efficient method to compute the pair-wise *GCD* for all pairs of distinct moduli, a quasilinear-time algorithm based on Bernstein's algorithm can be employed [5]. The algorithm's description can be found in [5]. The whole Bernstein method would cost: $O(nm \times (\log m)^2 \times \log\log m)$ [5] [23].

### *E. Factoring RSA keys*

*1) Factorable Keys in Our Dataset:* In this section, we want to test the hypothesis that the problems detected of factorable keys on the Internet could also concern e-commerce websites. Therefore, we ran an open source implementation of the quasilinear *GCD* computation [15] on the set of distinct moduli which resulted from our scan. We could factor 7 distinct moduli affecting 7 different SSL servers registered to .com and .co domain names. All the 7 factored moduli share the same factor. By manual inspection, we found that 6 of the factored keys are still in use by the servers as of this writing. The affected servers' web pages have the same design, serving as login pages only. There is no information that can confirm the devices type, but the HTML code for the login pages confirms that the web pages are a user interface administration tool. However, after we completed the next experiment (in the combined moduli set), we found many affected servers that showed the exact web interface and have the word "adsl" appears in a sub-domain for Internet Service Provider (ISP) domain names. This indicates that the affected web servers we found are installed in routers. Such web servers, which are built into the devices during the manufacturing and serve as a web management interfaces for the devices are known as EWSs [25]. We tried to open the web pages for the same domain names using plain HTTP protocol and the websites we found are as follows: 4 websites belong to companies; 1 general website; 1 non-English website registered to a broadcasting corporate (according to whois online tool [26]). The companies' websites open only with plain HTTP, while the HTTPS opens the EWSs login page. In terms of certificate details, the moduli were found in expired, self-signed (system-generated) certificates and issued by the same issuer. The validity dates for all the affected certificates show that they share the same starting year, day, hour and, with one exception,

minute (the exception differs only by approximately 1 minute). We also queried all the distinct certificates (i.e. with distinct moduli) that have been issued by the same issuer. We found 27 certificates with 27 distinct non-vulnerable moduli. Most of them share the same validity dates with one or more other certificates, with slight differences in minutes. Reference to [5], the root cause for the occurrence of such factorable keys is insufficient entropy at boot time. From our experiment, we conclude that there is no factored moduli belong to e-commerce websites, and that the affected servers are EWSs serving as remote management web interfaces for network devices. Our results confirm the findings of [5] which provided thorough analysis of the low-entropy problem. However, we believe that such vulnerability in embedded devices should not be underestimated. Factored keys would allow an adversary from decrypting an intercepted traffic coming to the vulnerable device. This in turn can result in compromising the administrator's credentials. Once the administrative credentials become in the adversary's hands, he has control of the device and this would allow him several types of attacks including DoS, corporate espionage, and firmware modification [25].

*2) Factorable Keys in the Combined Set:* Because we were looking for more factorable keys, we combined our moduli set with EFF set. Our set added 291,693 new distinct moduli to the EFF set. We repeated the experiment, resulting in a total of 6,513 distinct factorable keys. Out of these, 5 new distinct factorable keys came from our dataset. Table III shows the number of factorable keys found in each set along with the original set size and the execution time. The big difference in the number of factorable keys found in our set compared to the combined set can be due to several factors: First, our Internet scan was based on domain names while EFF scan was based on IP addresses. There is a considerable number of affected devices that did not return a domain name when we performed rDNS lookup. Second, most of the factored keys appear in certificates for sub-domains and seldom in main domain names. Our scan included main domain names only. Third, a single IP can be behind several domain names. Therefore, domain-name-based scans such as ours would require much larger input than it in IP-based scans. Finally, EFF scan is universal while our scan is for a fraction of domain names. However, we chose domain-name-based scanning strategy as it would allow us to target specific sector by the TLDs (commercial: .com and .co) and directly and accurately identify the companies or stores to which the factored key belongs.

*3) Factorable Keys in Non-expired Certificates:* In this section, we want to determine whether certain keys factored in the past (EFF keys have been included in [4] [5] work) are still in use today. We limited our inspection to vulnerable moduli resulted from the combined set that appear in non-expired certificates. As of January 4, 2013, we had found 798 distinct vulnerable moduli in non- expired certificates, affecting 6,167 servers with distinct IPs (the vulnerable moduli were repeated among many servers). Figure 2 illustrates the area we inspected.

We questioned how many .com, .co and also .edu servers are affected by the non-expired vulnerable moduli. Since the EFF scan was based on IPs, we had to make a reverse DNS lookup (rDNS) to retrieve the domain names for the IPs that showed

TABLE III. FACTORABLE KEYS

| # | The Set | | |
|---|---|---|---|
| | ***Our Set*** | ***Our Set + EFF Set*** | ***EFF Set*** |
| Distinct Keys | 385,140 | 4,225,058 | 3,933,365 |
| Factored Keys | 7 | 6,513 | 6,508 |
| Execution time | 286.075 sec. | 3,381.675 sec. | 2,890.326 sec. |

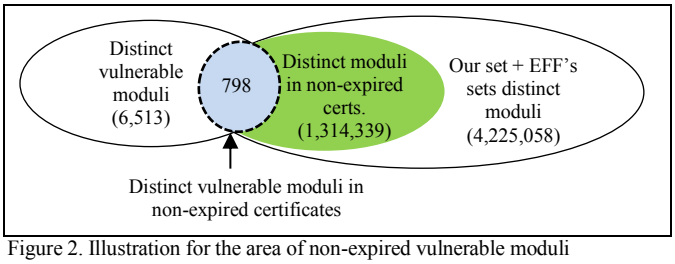


Figure 2. Illustration for the area of non-expired vulnerable moduli

factorable keys. 1,109 (17.98%) of the IPs did not return domain names. The rDNS lookup returned 268 .com and .co, of which 112 servers (41.79%) are available and still using the factored keys, 34 servers (12.69%) have changed their keys and 122 (45.52%) are not available. We also found a total of 171 .edu servers, of which 63 (36.84%) still use factored keys, 49 servers (28.65%) have changed their keys and 59 servers (34.50%) are not available. Note that the nature of these devices may cause them to be unavailable at all times. Therefore, we expect to have found more vulnerable keys if we had tested the unavailable devices at other times. Again, all the affected web servers that we inspected appear as EWSs deployed in a variety of hardware including network printers, photocopiers, routers, and IP-based remote power management.

## VI. STUDY LIMITATIONS

In this section, we list some limitations to our study. The registration for .com and .co domains is not restricted. Therefore, not every .com and .co domain name in our list is for a commercial website. In addition, the domain names list we used was created in 2009 and is not very recent. From the technical side, since our scanning program was built in Java, our scan was bounded to Java Security. For example, we could not complete SSL handshakes with servers that offered SSLv2 or servers with key lengths < 512-bit (length = 512 is allowed). However, we found means of circumventing retrieving untrusted certificates or certificates with invalid hostnames that Java does not allow by default [27].

## VII. DEFENSES

Several solutions have been proposed to warn end users about vulnerable keys. "Check Your Key" is an online solution developed by [5]. It checks keys against a database of known vulnerable keys such as the factored RSA keys in SSL certificates which resulted from [5] study. We tested some SSL servers that showed factorable keys in our study, the service successfully detected them but it shows an error whenever we check a server with expired certificate. Although many factored keys appear in expired certificates and are still in use by web servers, it seems that the service can not report them. We have sent an enquiry about this issue to one of the authors to confirm this point, but we have not yet received a response.

## VIII. CONCLUSION

In response to recent claims by several authors that many RSA keys can be factored, with the hypothesis that the problems detected for some devices could concern major e-commerce websites, we decided to proceed with our own Internet scan with a particular focus on commercial websites. Our study also aimed to determine whether certain keys factored in the past are still in use today, and if it is possible for an amateur with a single PC using publicly available information to break these keys in a reasonable amount of time.

Our work involved looking at large datasets gathered from our scan and one public database. We compiled various statistics on key sizes used in various countries. We report that servers located in Italy and France showed the highest percentage of insecure key sizes, while servers located in the UK, US and Australia were leading in adopting secure key lengths with 2048-bit or more out of all of the countries in the comparison. This indicates that it may be riskier to make secure connections with servers located in Italy or France than with servers located in the UK and US. However, as this depends on many factors, we were unable to verify this.

We devoted much effort to study individual RSA keys. We analyzed the feasibility of factoring some of these keys by the naïve pair-wise *GCD* and also by the efficient algorithm based on Bernstein algorithm. We ran the whole attack in order to see if there are vulnerable keys. We identified and factored a total of 6,513 such keys.

From our scan, in the keys that we factored and for which we traced the web servers, none affected e-commerce websites. However, we report factored keys used by corporations but the certificates were deployed for EWSs and not for an ordinary web server. Further research could focus on exhibiting vulnerable live commercial websites by replicating the experiment with more recent and larger lists of commercial domain names.

From the combined set of moduli, we inspected the non-expired vulnerable moduli that appeared in .com, .co and .edu servers. On average, we found around 40% of the keys which known to be factored in the past are still in use today and some affected certificates will not expire until 2038.

Our results suggest that, even though we have not identified a specific live Internet store for which we can crack the keys, there is a possibility for anyone who might replicate the same experiment to mount attacks on the Internet. This could include MITM, DoS, data modification and firmware modification attacks. It also seems that it is now possible to impersonate a number of embedded devices such as routers or gateway servers on the Internet. The exact implications of this for national and international security require further investigation.

## ACKNOWLEDGMENT

A major part of this paper has spanned from the author's MSc. thesis submitted to University College London. Special thanks go to Dr. Nicolas Courtois, the thesis supervisor.

## REFERENCES

[1] R. Atkinson, S. Ezell, A. Scott, D. Castro, and B. Richard. (2010, Mar.). *The Internet Economy 25 Years After .com* [Online]. Available: http://www.itif.org/files/2010-25-years.pdf.

[2] VeriSign. Inc. (2012, Jul.). *The Domain Name Industry Brief* [Online]. Available: http://www.verisigninc.com/assets/domain-name-brief-july2012.pdf

[3] S. Thomas, *SSL and TLS Essentials*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[4] A. Lenstra, J. Hughes, M. Augier, J. Bos, T. Kleinjung and C. Wachter, "Ron was wrong, Whit is right," IACR Cryptology ePrint Archive, Report 2012/064, 2012.

[5] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices," in *Proc. of the 21st USENIX conf. on Security symposium (Security'12)*, Berkeley, CA, USA, 2012, pp. 35-35.

[6] T. Dierks and E. Rescorla. (2008, Aug.). *The TLS protocol version 1.2* [Online]. Available: https://tools.ietf.org/html/rfc5246

[7] A. Shamir, R. Ronald, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM 21*, vol. 26, no. 1, pp. 120–126, 1978.

[8] D. Boneh and H. Shacham, "Fast variants of RSA," *CryptoBytes*, vol. 5, no. 1, pp. 1–9, 2002.

[9] H. Lee, T. Malkin, and E. Nahum, "Cryptographic Strength of SSL / TLS Servers: Current and Recent Practices," in *Proc. of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07)*, New York, NY, USA, 2007, pp. 83–91.

[10] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL Landscape – A Thorough Analysis of the X . 509 PKI Using Active and Passive Measurements," in *Proc. of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC '11)*, New York, NY, USA, 2011, pp. 427–444.

[11] IP Address Geolocation Online Service [Online]. Available: http://www.ipaddresslabs.com.

[12] MaxMind-GeoIP IP Address Locating Database [Online]. Available: http://www.maxmind.com/en/geolocation_landing.

[13] I. Poese, S. Uhlig, M. Kaafar, B. Donnet, and B. Gueye, "IP Geolocation Databases: Unreliable?," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 53–56, 2011.

[14] D. BERNSTEIN (2004, May), *HOW TO FIND SMOOTH PARTS OF INTEGERS* [Online]. Available: http://cr.yp.to/papers.html.

[15] N. Heninger and J. Halderman (2012, Mar.). *Fastgcd Source Code* [Online]. Available: https://factorable.net/resources.html.

[16] EMC Corporation (2012). *RSA Laboratories - 3.1.5 How large a key should be used in the RSA cryptosystem?* [Online]. Available: http://www.rsa.com/rsalabs/node.asp?id=2218#footnote.

[17] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. Osvik, H. Riele, A. Timofeev, and P. Zimmermann, "Factorization of a 768-bit RSA modulus," in *Proc. of the 30th annual conf. on Advances in cryptology (CRYPTO'10)*, Santa Barbara, CA, USA, 2010, pp. 333–350.

[18] E. Barker, W. Barker, W. Burr, P. William, and M. Smid. (2012, Jul.). *Recommendation for Key Management – Part 1: General (Revision3)* [Online]. Available: http://csrc.nist.gov/publications/nistpubs

[19] A. Lenstra (2004, June), *Key Lengths* [Online]. Available FTP: ftp://cm.bell-labs.com/who/akl/key_lengths.pdf

[20] D. Giry. (2013, Feb.). *Cryptographic Key Length Recommendations* [Online]. Available: http://www.keylength.com.

[21] Microsoft (2012, Aug.). *Microsoft Security Advisory: Update for minimum certificate key length* [Online]. Available: http://support.microsoft.com/kb/2661254.

[22] J. Katz and Y. Lindell, *Introduction To Modern Cryptography*. Boca Raton, FL: Chapman & Hall/CRC, 2008.

[23] I. Mironov (2012, May). *Factoring RSA Moduli. Part I* [Online]. Available: http://windowsontheory.org/2012/05/15/979/.

[24] The Electronic Frontier Foundation (2010, Aug.). *The EFF SSL Observatory* [Online]. Available: https://www.eff.org/observatory.

[25] M. Sutton (2012). *Corporate Espionage for Dummies: The Hidden Threat of Embedded Web Servers* [Online]. Available: http://365.rsaconference.com/servlet/JiveServlet/previewBody/3453-102-1-4552/HT2-202.pdf.

[26] *Domain tools* [Online]. Available: http://whois.domaintools.com.

[27] S. Nakov (2009, Jul.). *Disable Certificate Validation in Java SSL Connections* [Online]. Available: http://www.nakov.com/blog/2009/07/16