

On the Uniqueness of RSA Keys in EJBCA PKI Systems

DOI: <https://doi.org/10.54654/isj.v1i27.1242>

Dinh Quoc Tien*, Nguyen Le Minh

Abstract— RSA deployments in EJBCA-based public key infrastructures (PKIs) rely on X.509 Subject Key Identifiers (SKIs) to prevent reuse of identical public keys. However, SKI-level checks fail to detect shared-prime collisions, where distinct RSA moduli reuse a prime factor and become factorable via batch GCD analysis. This paper examines why existing SKI mechanisms cannot guarantee RSA key uniqueness and how such weaknesses propagate through enrollment and renewal processes in EJBCA.

We propose two complementary controls that extend uniqueness from the public key to RSA's secret parameters. The first is a secrecy-preserving duplication check, where one-way commitments to the primes (p, q) are embedded in the certificate request and verified against a commitment registry to block reuse without exposing secret values. The second is an identity-scoped RSA generation algorithm that deterministically maps subject identifiers to disjoint prime-search intervals, ensuring well-spaced and non-overlapping primes across users.

Tóm tắt— Hiện nay, hạ tầng khóa công khai (Public Key Infrastructure - PKI) sử dụng EJBCA với thuật toán RSA hiện đang dựa vào trường Subject Key Identifier (SKI) trong chứng thư X.509 để ngăn việc tái sử dụng cùng một khóa công khai. Tuy nhiên, kiểm tra ở mức SKI không phát hiện được các va chạm “shared-prime”, khi các mô-đun RSA khác nhau cùng dùng chung một thừa số nguyên tố, khiến chúng có thể bị phân tích nhân tử thông qua phép GCD hàng loạt. Bài báo này phân tích lý do vì sao cơ chế SKI hiện tại không thể bảo đảm tính duy nhất của khóa RSA, cũng như cách điểm yếu này lan truyền trong quá trình đăng ký và gia hạn chứng thư trên EJBCA.

Nhóm tác giả đề xuất hai cơ chế bổ trợ nhằm mở rộng phạm vi kiểm soát tính duy nhất từ khóa công khai sang cả tham số bí mật của RSA. Thứ nhất là cơ chế kiểm tra trùng lặp bảo toàn quyền riêng tư các “cam kết một chiều” đối với hai số nguyên tố (p, q) được nhúng vào yêu cầu chứng thư và kiểm tra với một “sổ đăng ký cam kết” để phát hiện việc tái sử dụng mà không làm lộ giá trị bí mật. Thứ hai là thuật toán sinh khóa RSA theo phạm vi định danh, ánh xạ xác định (deterministic) thông tin chủ thể sang các khoảng tìm kiếm nguyên tố không giao nhau, từ đó bảo đảm các số nguyên tố được phân bố cách đều và không trùng lặp giữa các người dùng.

Keywords— RSA; EJBCA; Public Key Infrastructure (PKI); X.509; Subject Key Identifier (SKI); shared-prime collisions; prime reuse; GCD attack; privacy-preserving duplication check; HSM/PKCS#11 integration.

Từ khóa— RSA, EJBCA, hạ tầng khóa công khai (PKI), X.509, SKI, tái sử dụng số nguyên tố, tấn công GCD, định danh, tích hợp HSM/PKCS#11.

I. INTRODUCTION

Nowadays, all areas of social life tend to be technological, and electronic transactions are mandatory. Security and confidentiality issues in electronic transactions are always a matter of concern, but the development of the Public Key Infrastructure (PKI) application has completely solved those problems. The PKI public key infrastructure includes a set of roles, policies, hardware, software and processes needed to create, manage, distribute, use, store and revoke digital certificates and manage public key encryption. To deploy PKI, people use the Digital Certificate Issuance and Management System (abbreviated as the CA System).

In the implementation of a PKI system, technologies such as ASN.1 and X.509, public key standards like RSA, ECC, and APIs are extremely complex for most users and developers. Most importantly, integrating all of

This manuscript was received on December 10, 2025. It was reviewed on January 27, 2026, revised on March 3, 2026 and accepted on May 4, 2026.

* Corresponding author

these components (to ensure system compatibility) is very challenging. However, PKI system implementers do not need to be overly concerned with these complex technologies due to the availability of several helpful tools:

- ASN.1 parsing is handled by the BouncyCastle library.
- Digital signatures using RSA and ECC are also implemented using BouncyCastle in software or are executed within a Hardware Security Module (HSM).
- Standardization has been carried out by organizations such as the IETF, etc.

In the implementation of the EJBCA PKI system, there are 13 challenging issues discussed in the document “Hard Core PKI presentation at PrimeKey Tech Days 2015” [1]. These issues have essentially been resolved, including the problem of ensuring the uniqueness of public keys. However, for RSA public key systems, avoiding public key duplication alone is not sufficient.

To develop and integrate the key duplication prevention feature, we have carried out the following main tasks:

- Studied several key duplication prevention solutions for RSA and ECC public key cryptosystems, as these are used within the EJBCA PKI system.
- Researched and developed program modules to implement key duplication prevention mechanisms in the digital certificate issuance process, including the issuance of RSA and ECC software certificates, as well as RSA certificates for PKI tokens.

II. KEY DUPLICATION PREVENTION SOLUTIONS FOR RSA AND ECC PUBLIC KEY CRYPTOSYSTEMS

A. General Public Key Duplication Prevention for RSA and ECC Systems

Public key duplication prevention for RSA and ECC systems is generally implemented through the X.509 digital certificate format (according to RFC5280) [2], which includes the optional SubjectKeyIdentifier extension field:

SubjectKeyIdentifier ::= KeyIdentifier

KeyIdentifier ::= OCTET STRING

This field provides a means to identify a certificate that corresponds to a specific public key. The value of the KeyIdentifier field is an OCTET STRING derived from the public key or generated via a method that ensures uniqueness. Currently, there are two common methods for generating key identifiers from public keys:

- Method 1: The key identifier is derived from the 160-bit SHA-1 hash of the OCTET STRING of the SubjectPublicKey (the public key).

- Method 2: The key identifier is composed of a 4-bit prefix 0100 followed by the 60 least significant bits of the SHA-1 hash of the OCTET STRING of the SubjectPublicKey.

B. Public Key Duplication Prevention for ECC Systems

Issuing EC certificates to users is based on generating a secret/public key pair following the relationship $P = dG$ over a defined domain parameter set, where G is the base point, P is the public key, and d is the randomly generated private key. The challenge is to ensure that duplicate secret/public key pairs are not issued to users within the system.

However, intentionally processing the private key before issuance may introduce potential security vulnerabilities. Fortunately, in the case of elliptic curve cryptography (EC), duplicate private keys d will inevitably lead to duplicate public keys P , which is not the case with RSA systems. Therefore, the key duplication prevention solution for EC certificate issuance involves comparing the hash value of the public key P , without needing to hash or intentionally process the random private key d . This helps maintain both uniqueness and security. Similar concerns regarding the security of RSA parameters and digital signature schemes have also been analyzed in [9].

Thus, the public key duplication prevention solution for EC involves checking for duplicate SubjectKeyIdentifier values of EC public keys, which effectively ensures the prevention of duplication of the private key d .

C. Public Key Duplication Prevention for RSA Systems

An RSA digital signature key pair consists of an RSA private key, which is used to compute a digital signature, and an RSA public key, which is used to verify a digital signature. An RSA public key consists of a modulus M , which is the product of two positive prime integers p and q (i.e., $M = pq$) and a public key exponent e . Thus, the RSA public key is the pair of values (M, e) and is used to verify digital signatures. The corresponding RSA private key consists of the same modulus M and a private key exponent d that depends on M and the public key exponent e . Thus, the RSA private key is the pair of values (M, d) and is used to generate digital signatures.

For RSA public key systems, avoiding duplication of public keys alone is not sufficient. Therefore, the critical secret parameter in the RSA cryptosystem is d , which is calculated using the following formula:

$$d = e^{-1} \bmod ((p-1) * (q-1))$$

Therefore, to ensure the confidentiality of the parameter d , it is essential to keep the two secret prime numbers p and q confidential. However, according to the study in [3], the private key pair p and q can be completely recovered by computing the Greatest Common Divisor (GCD) of different modulus pairs, as different RSA moduli may still share a common prime factor, either p or q .

Since GCD computations on large numbers are performed very quickly (for example, the GCD of 1024-bit integers can be computed in microseconds), the study in [3] successfully calculated the GCD of 11 million public RSA moduli in under two hours. As a result, they were able to retrieve private keys from 64,000 TLS servers (0.5%) and 108,000 SSH servers (1.06%).

From the above analysis, it is clear that to ensure the security of RSA private keys, it is critical that RSA prime numbers must be unique at the time they are generated.

Thus, the general solution mentioned earlier only guarantees private key duplication prevention for ECC cryptosystems, while for

RSA cryptosystems, it only ensures public key duplication prevention. To solve this problem for RSA, we have studied and proposed two additional solutions, which are presented below.

Solution 1: Hash Comparison of Prime Numbers for RSA Cryptosystems

The hash comparison of prime numbers solution for RSA is based on a simple idea: compare the hash values of the prime numbers used in RSA key pairs to ensure that these parameters are not duplicated.

In the EJBCA system, the issuance of software-based RSA digital certificates is performed similarly to Solution 1. However, the difference lies in the fact that the hash values of the two generated prime numbers p and q are additionally included in the SubjectKeyIdentifier field after key generation. This approach helps detect and prevent the reuse of prime factors across RSA key pairs.

This solution is implemented as illustrated in Figure 1 below.

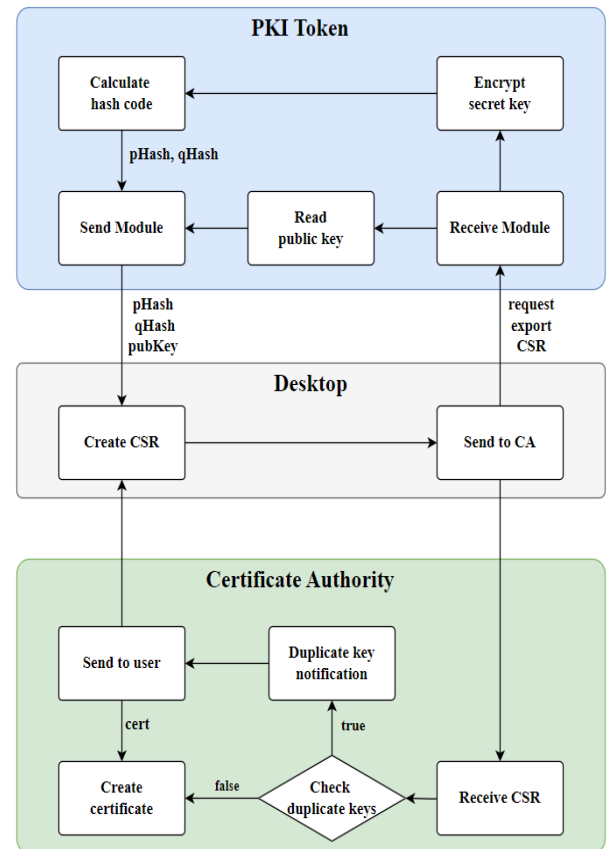


Figure 1. RSA Key Duplication Prevention Using Private Key Hash Comparison Method

A solution for generating 2048-bit RSA parameters directly on PKI Token devices has been presented in [10], ensuring the key pair is created inside the device and cannot be extracted.

This solution operates as follows:

Step 1: The computer sends a command to the PKI Token device requesting the generation of a CSR (Certificate Signing Request).

Step 2: Upon receiving the request, the PKI Token decrypts the private key, computes its hash, reads the public key from its internal memory, and sends both the private key hashes and the public key back to the computer.

Step 3: The computer receives the private key hash values (pHash, qHash) and the public key. It then generates the CSR, embedding the private key hashes into the State (ST) field of the CSR. The completed CSR is then submitted to the Certificate Authority (CA).

Step 4: The CA, upon receiving the CSR, performs a duplicate key check by comparing the private key hash values with those of previously issued keys stored in the database. If no duplication is found, the CA issues the digital certificate and sends it back to the user. If duplication is detected, the CA sends a duplicate key notification to the user.

Solution 2: A solution employing an identity-based unique RSA parameter generation algorithm.

One of the requirements of a digital certificate issuing system is to determine who has the authority to generate and issue secret parameters for users. In a PKI model—specifically PKIX (i.e., PKI based on X.509)—the certificate authority system includes several roles, such as CA (Certificate Authority), RA (Registration Authority), and RAO (Registration Authority Operator), among others.

However, according to RFC 2510 “PKI Certificate Management Protocols”, section 2.2.1.3 “Location of key generation”, the location of key generation can be the CA, RA, or EE (End Entity). It is noted that if the key pair is generated by the CA or RA, the certificate authority system only needs to maintain the

absolute secrecy of the user’s private/public key pair. This model is appropriate when there is control over key management.

If, on the other hand, the user generates the key pair themselves, several challenges arise: First, the user must have a key generation program; second, user-generated key pairs may collide (although the probability is very small). To avoid these complications, it is necessary to use algorithms that can generate unique parameters for each certificate.

As is well known, in a certificate issuing system, the certificate’s serial number (i.e., its identifier) is unique. Therefore, we propose using the certificate’s serial number as an identifier to generate unique parameters p and q for each certificate. However, there are two main security concerns regarding this approach.

First, since the serial number is publicly available along with the certificate, is it still secure? In 2004, Dan Kaminsky’s paper [8] demonstrated a collision attack to forge certificates, assuming the system used the MD5 hash function and assigned serial numbers sequentially. Today, such attacks are largely ineffective as certificate issuing systems use collision-resistant hash functions like SHA-256/384/512.

Second, if the key identifier information is exposed, can the parameters p and q be determined? Below, we address this issue to generate prime numbers from an identity within a range that maintains a sufficiently high prime density relative to the security level (128, 160, and 192 bits), while also conforming to RSA parameter generation standards in the field.

III. GENERATE UNIQUE RSA PRIME NUMBERS BY IDENTIFIER

A. For the criterion of prime number density in the range large enough compared to the security s

The main purpose of the interval partitioning method is to generate different prime numbers for different users. To generate an n -bit prime number for different users, we divide the interval $[2^{n-1}, 2^n]$ into multiple distinct sub-intervals. A prime number is then generated for each user

within a specific sub-interval. The number of users we consider here is 2^{64} , which corresponds to the number of certificate serial numbers in the certificate issuance system.

We divide the interval $[2^{n-1}, 2^n]$ into 2^b distinct sub-segments (corresponding to the number of users). In here we choose $b = 64$ to correspond to the serial number of the certificate on the certificate issuance system. Let $A = 2^{n-1}$, $C = 2^n$ then the prime number will have to be generated in its range of $\left[A + \frac{B(C-A)}{2^b}, A + \frac{(B+1)(C-A)}{2^b}\right]$. $B = 0, 1, \dots, 2^b - 1$ for each user we give a different value B , called an identifier. That means the value B can be used to identify the user. The remaining problem here is that the prime number must be in a sufficiently large range and meet the RSA parameter generation standards.

For criteria $|p - q| \geq 2^a$

Each user will generate a pair of prime numbers p, q for the RSA cryptosystem in a given range. In addition, according to the secure RSA parameters, the pair of prime numbers p, q must satisfy the condition $|p - q| \geq 2^a$. For different values of $nlen$ (i.e., different key sizes), random primes or primes with conditions are methods allowed for the generation of p and q . In addition, all IFC keys shall meet the following criteria in order to conform to this standard, FIPS 186-5 [6] step 2 in section A.1.1 (page 35): $a = (nlen/2) - 100$, where $nlen$ is the appropriate length for the desired $s = \text{security_strength}$.

We divide the interval $\left[A + \frac{B(C-A)}{2^b}, A + \frac{(B+1)(C-A)}{2^b}\right]$ into 2 intervals so that the distance between these 2 segments is equal to 2^a . We take $N_1 = A + \frac{B(C-A)}{2^b}$ and $N_2 = A + \frac{(B+1)(C-A)}{2^b}$. Let $N_{TB} = \frac{N_1 + N_2}{2}$, take $N_{11} = N_{TB} - \frac{2^a}{2}$ and $N_{22} = N_{TB} + \frac{2^a}{2}$. Then we have:

$$[N_1, N_2] = [N_1, N_{11}] \cup [N_{11}, N_{22}] \cup [N_{22}, N_2] \text{ and } N_{22} - N_{11} = 2^a.$$

That is, we generate prime number q in the interval $[N_1, N_{11}]$ and generate prime number p in the interval $[N_{22}, N_2]$, then the condition is

guaranteed $|p - q| \geq 2^a$. Thus, each user is only allowed to generate prime number pairs p, q in the interval according to the unique identifier B .

In the following section, we will present the problem of determining bounds for auxiliary primes F and Q that satisfy the parameter criteria through evaluating the cardinality of primes in the interval $[N_{22}, N_2]$. We have:

$$\begin{aligned} N_{22} &= N_{TB} + \frac{2^a}{2} = N_{TB} + 2^{a-1} \\ &= A + \frac{(2B+1)(C-A)}{2^{b+1}} + 2^{a-1} \\ &\approx 2^{n-1} + \frac{(2B+1)2^{n-1}}{2^{b+1}} + 2^{a-1} \\ &= 2^{n-1} + B2^{n-b-1} + 2^{n-2} + 2^{a-1} \end{aligned} \quad (1)$$

$$\begin{aligned} N_2 &= A + \frac{(B+1)(C-A)}{2^b} \\ &\approx 2^{n-1} + \frac{(B+1)2^{n-1}}{2^b} \\ &= 2^{n-1} + (B+1)2^{n-b-1} \end{aligned} \quad (2)$$

B. For the criterion of prime number density in the range large enough compared to security s

To consider the criterion of prime density in the interval large enough compared to the safety level s , we evaluate the density of prime numbers in the interval $[N_{22}, N_2]$ based on Ivan Soprounov's theorem [7] stated as follows:

Theorem 2.1 [7]. Let $\pi(x, q)$ denote the number of all prime numbers p no greater than x , congruent to $a \pmod q$, with $a, q \in \mathbb{N}$ such that $\gcd(a, q) = 1$. Then:

$$\pi(x, q) \sim \frac{x}{\varphi(q) \log x} \quad (3)$$

where $\varphi(q)$ is the Euler function.

Our prime numbers are of the form $N = 2F[Qz + u_0] + 1 = 2FQz + (2Fu_0 + 1)$. We let $t = \gcd(2FQ, 2Fu_0 + 1)$, since F and Q are prime numbers and since t must be $2Fu_0 + 1 = 2v_0Q - 1$ odd, t can only be F or Q or FQ or 1. Therefore $\gcd(2Fu_0 + 1, F) = 1$. t is different from F and FQ . $v_0Q - u_0F = 1$. Thus, we have

$\gcd(U, V) = 1, \gcd(2Fu_0 + 1, Q) = 1$ so we can apply Dirichlet's theorem on the number of prime numbers $\pi_{A,B}(N_{22})$ and $\pi_{A,B}(N_2)$ (or abbreviated as $\pi(N_{22})$ and $\pi(N_2)$) belong to the arithmetic progression $Uz + V$ not exceeding N_{22} and N_2 respectively as follows:

Our prime numbers are of the form $N = 2F[Qz + u_0] + 1 = 2FQz + (2Fu_0 + 1) = Uz + V$, with $U = 2FQ$, $V = 2Fu_0 + 1$. Consider the arithmetic progression $N = 2F[Qz + u_0] + 1 = 2FQz + (2Fu_0 + 1)$ with F, Q is prime, and satisfies. Furthermore, we have $\gcd(2FQ, 2Fu_0 + 1) = 1$.

$$\begin{aligned}\pi(N_{22}) &= \frac{1}{\varphi(2FQ)} \frac{N_{22}}{\ln N_{22}} \\ &\approx \frac{1}{\varphi(2FQ)} \frac{2^{n-1} + B2^{n-b-1} + 2^{n-2} + 2^{a-1}}{\ln(2^{n-1} + B2^{n-b-1} + 2^{n-2} + 2^{a-1})} \\ \pi(N_2) &= \frac{1}{\varphi(2FQ)} \frac{N_2}{\ln N_2} \\ &\approx \frac{1}{\varphi(2FQ)} \frac{2^{n-1} + (B+1)2^{n-b-1}}{\ln(2^{n-1} + (B+1)2^{n-b-1})}\end{aligned}$$

So, the number of prime numbers in the interval $[N_{22}, N_2]$ with be $\pi(N_2) - \pi(N_{22})$, that is, we have:

$$\begin{aligned}\pi(N_2) - \pi(N_{22}) &\approx \frac{1}{\varphi(2FQ)} \left[\frac{2^{n-1} + (B+1)2^{n-b-1}}{\ln(2^{n-1} + (B+1)2^{n-b-1})} - \frac{2^{n-1} + B2^{n-b-1} + 2^{n-2} + 2^{a-1}}{\ln(2^{n-1} + B2^{n-b-1} + 2^{n-2} + 2^{a-1})} \right] \\ &\approx \frac{1}{\varphi(2FQ)} \frac{1}{n \ln 2} [2^{n-1} + (B+1)2^{n-b-1} - (2^{n-1} + B2^{n-b-1} + 2^{n-b-2} + 2^{a-1})] \\ &= \frac{1}{\varphi(2FQ)} \frac{1}{n \ln 2} [2^{n-b-2} - 2^{a-1}] \\ &= \frac{2^{a-1}[2^{n-a-b-1} - 1]}{\varphi(2FQ)n \ln 2}\end{aligned}$$

With $F \approx 2^h$, $Q \approx 2^k$, $n \approx 2^m$, and due to $\varphi(2FQ) \approx FQ \approx 2^{h+k}$, $\ln 2 < 1 \Rightarrow n \ln 2 < n \Rightarrow n \ln 2 < 2^m$, so $\varphi(2FQ)n \ln 2 < 2^{h+k+m}$. And combine with $2^{n-a-b-1} - 1 > 2^{n-a-b-2}$. Thus we have:

$$\pi(N_2) - \pi(N_{22}) > 2^{n-b-h-k-m-3}$$

Although each user only needs 2 primes for the RSA cryptosystem, if the generating range $[N_{22}, N_2]$ has very few primes, $2F[Qz + u_0] + 1$ then the attacker will exhaust the known identifier B to find 2 primes of the RSA modulo. From there, we also give the corresponding thresholds of the magnitude h, k of the auxiliary primes F, Q that meet the RSA parameter standards for bit lengths of 2048, 3072, 4096, 8192. The basic observation here is that the security corresponds to the standard key length, not fixed (in the standard key length [4, 5] it is fixed at 80 bits based on the complexity of the problem of finding collisions according to the birth date paradox of the SHA1 hash function with a length of 160 bits). Below we will evaluate the security level according to the SECRET and TOP SECRET levels to comply with the newly issued RSA parameter standard s equal to 128, 160 and 192 bits respectively. Thus, we need to have:

$$\begin{aligned}\pi(N_2) - \pi(N_{22}) &\geq 2^s \Leftrightarrow 2^{n-b-h-k-m-3} \geq 2^s \\ &\Leftrightarrow n - b - h - k - m - 3 \geq s \\ &\Leftrightarrow n - m - s - 67 \geq h + k, \text{ with } b = 64\end{aligned}$$

Thus, we have the following system of inequalities:

$$\begin{cases} n - m - s - 67 \geq h + k \\ h \geq 0.5n \end{cases}$$

Furthermore, to comply with the secure RSA parameter standard, in FIPS 186-5 [6] according to section A.1.1 (page 33-34), the corresponding primes F and Q have a minimum bit size of h_1 and k_1 we have the following system of inequalities:

$$\begin{cases} n - m - s - 67 \geq h + k \\ h \geq 0.5n \\ h \geq h_1 \\ k \geq k_1 \end{cases} \Leftrightarrow \begin{cases} 0.5n \leq h \leq n - m - s - 67 - k_1 \\ k_1 \leq k \leq n - m - s - 67 - h \end{cases} \quad (4)$$

Following way to choose the bit length h of F and the bit length k of Q : First, based on the length of the RSA parameter, choose s , then randomly choose h that satisfies the condition of

the first inequality $0.5n \leq h \leq n - m - s - 67 - k_1$, then randomly choose a value k that satisfies the condition of the second inequality $k_1 \leq k \leq n - m - s - 67 - h$.

TABLE I. BIT LENGTH OF AUXILIARY PRIMES

RSA modulus size and prime numbers	S	k_1	m	h, k
2048/1024	128	140	10	$512 \leq h \leq 679$ $140 \leq k \leq 307$
3072/1536	128	170	11	$768 \leq h \leq 1160$ $170 \leq k \leq 562$
4096/2048	160	200	11	$1024 \leq h \leq 1160$ $200 \leq k \leq 786$
8192/4096	196	255	12	$2048 \leq h \leq 3565$ $255 \leq k \leq 1773$

Clearly, in FIPS 186-5 [6] only the minimum lengths for each of the auxiliary primes are given as dependent on RSA modulus $nlen$, where $nlen$ is the length of the modulus in bits. Note that $nlen$ is also called the key size. The lengths of the auxiliary primes may be fixed or randomly chosen, subject to the restrictions in Table A.1 in FIPS 186-5 [6]. Specifically: $nlen = 2048$, then h and k are greater than 140 bits; $nlen = 3072$, then h and k are greater than 170 bits; $nlen = 4096$, then h and k are greater than 200 bits; and there is no such case for $nlen = 8192$. In our paper, the lower bounds of the auxiliary primes are the same as in FIPS 186-5 [6]; however, to ensure that the solution generates prime numbers p and q within a unique range, it is necessary to add the maximum lengths for each of the auxiliary primes, which are dependent on the RSA modulus $nlen$, as shown in Table I of the paper. In this table, we also include the case where $nlen = 8192$.

C. Unique RSA prime number generation algorithm by identifier.

In the RSA cryptosystem, each set of security parameters corresponding to an identifier requires generating two prime numbers

p and q . We generate prime number q in the interval $[N_1, N_{11}]$ and generate prime number p in the interval $[N_{22}, N_2]$ with conditions that satisfy the RSA parameter standards (regarding the auxiliary prime number and the distance between two prime numbers) as presented above.

Algorithm 1 below performs the generation of a prime number $2F[Qz + u_0] + 1$ in the range $[D, E]$ with identifier B for a set of RSA parameters and with z satisfying:

$$\frac{D-1}{2FQ} - \frac{u_0}{Q} \leq z \leq \frac{E-1}{2FQ} - \frac{u_0}{Q} \quad (5)$$

Algorithm 1: Algorithm for generating unique RSA prime numbers by identifier.

Input: n - Number of bits of the prime number to generate

B - RSA parameter set identifier

Output: Prime number $N = 2F[Qz + u_0] + 1$ satisfying RSA parameter standard.

Step 1: Randomly generate bit lengths h and k of F and Q according to (4).

Step 2: Generate a prime number Q of length k bits using the Cuberoot algorithm with auxiliary primes according to section A.1.1 (page 33-34).

Step 3: Generate a prime number F of length h bits using the Cuberoot algorithm with auxiliary primes according to section A.1.1 (page 33-34).

Step 4: Determine the boundary $[D, E]$ according to the identifier B .

Step 5: Generate a prime number N of length n bits as follows:

Step 5.1: Calculate the integer solution (u_0, v_0) of the equation $vQ - uF = 1$

$$v_0 = Q^{-1} \bmod F$$

$$u_0 = \frac{v_0 Q - 1}{F}$$

Step 5.2: Choose z randomly within the interval according to (5).

Step 5.3: Calculate $N = 2F[Qz + u_0] + 1$

Step 5.4: Randomly select a in the interval $[2, N - 1]$. Check that the following two equalities are satisfied at the same time:

$$(i) \gcd(a^{(N-1)/F}, N) = 1$$

$$(ii) a^{N-1} \equiv 1 \pmod N$$

If not satisfied then $z = z + 1$, then check if z is in the interval (5) then go back to Step 3, otherwise go back to Step 2.

Step 6: Return N

We used the following system for speed evaluation: CPU: Intel Core i5 3.5GHz; RAM: 8 GB; OS: Ubuntu 22.02; Java: Oracle JDK 8; EJBCA: EJBCA Enterprise Edition 6.7; Crypto Provider: Bouncy Castle 1.54. The original EJBCA 6.7 generates prime numbers during the RSA key pair generation process, using the SecureRandom (JVM) mechanism; employing the probabilistic prime number generation algorithm (Miller-Rabin) implemented by JCA or Bouncy Castle.

For speed comparison, we integrated the deterministic prime number generation algorithm according to section A.1.3 in FIPS 186-5 [6] into EJBCA 6.7. And then, we also integrated the deterministic prime number generation algorithm according to Algorithm 1 in our paper. The average time to generate an RSA key pair differs only in the time it takes to generate the two prime numbers p and q . Therefore, we can compare them through the average time to generate an RSA key pair as shown in Table II below.

TABLE II. COMPARISON OF BIT LENGTH OF AUXILIARY PRIMES

RSA key size	Probabilistic Primes in EJBCA 6.7	Provable Primes in FIPS 186-5 [6]	Algorithm 1 in the paper
2048	$\approx 150 - 400 \text{ ms}$	$\approx 300 - 700 \text{ ms}$	$\approx 1.1 - 1.5 \text{ s}$
3072	$\approx 500 - 900 \text{ ms}$	$\approx 1.0 - 1.3 \text{ s}$	$\approx 3.5 - 5.3 \text{ s}$
4096	$\approx 1.5 - 3 \text{ s}$	$\approx 1.9 - 4.1 \text{ s}$	$\approx 4.5 - 7.9 \text{ s}$

Thus, according to Table II, the average speed of generating an RSA key pair using our Algorithm 1 appears to be slower. However, it is still acceptable due to the trade-off for greater security.

IV. CONCLUSION

This work addresses a long-standing operational gap in RSA deployments within EJBCA-based PKIs: SKI-level checks prevent duplicate public keys but do not stop “shared-prime” failures, where distinct moduli incidentally reuse a prime and become factorable via batched GCD. We proposed two complementary controls that lift uniqueness from the public key to RSA’s secret parameters. First, a privacy-preserving duplication check commits to the primes (via one-way digests) inside the CSR and rejects enrollments that collide, blocking prime reuse without exposing p or q . Second, an identity-scoped RSA generation method deterministically maps a subject identifier to disjoint prime-search intervals and enforces well-spaced primes, preserving cryptographic strength and compatibility with HSM/PKCS#11 workflows and standard modulus sizes (2048 - 8192 bits). Together, these measures provide defense-in-depth against shared-prime compromises and materially strengthen operational guarantees of key uniqueness in real-world PKIs.

We discussed integration points for EJBCA, assessed storage/latency overheads, and analyzed security and privacy trade-offs, including renewal behavior and revocation handling. While our approach assumes honest key-generation endpoints and careful handling of prime commitments, it remains deployable with today’s tooling and complements existing SKI-based screening. Limitations include inter-CA collision detection at ecosystem scale and residual risks from weak entropy sources. Future work includes federated commitment registries across CAs, zero-knowledge set membership for even stronger privacy, hardware-enforced commitment generation, and migration paths that maintain comparable uniqueness guarantees in hybrid or post-quantum issuance workflows.

ACKNOWLEDGMENT

We thank the EJBCA open-source community for implementation insights, and the PKI operations engineers who shared valuable feedback on enrollment, renewal, and HSM integration workflows. We are grateful to our colleagues for comments on earlier drafts and to the anonymous reviewers for suggestions that improved clarity and scope. Any remaining errors are our own.

REFERENCES

- [1] PrimeKey, "13 Hardcore EJBCA PKI options explained", 2015. Access time: 26/4/2026, <https://blog.ejbca.org/2015/12/13-hardcore-ejbca-pki-options-explained.html>.
- [2] IETF, RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008.
- [3] 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. 21st USENIX Security Symposium*, Bellevue, WA, USA, pp. 205-220, 2012.
- [4] A. K. Lenstra, "Key lengths," in *The Handbook of Information Security*, 2004.
- [5] Keylength.com, "Cryptographic key length recommendation." Access time: 26/4/2026, <https://www.keylength.com>.
- [6] National Institute of Standards and Technology, *FIPS PUB 186-5: Digital Signature Standard (DSS)*, 2023. DOI: 10.6028/NIST.FIPS.186-5.
- [7] I. Soprounov, "A short proof of the prime number theorem for arithmetic progressions", 2010. Access time: 26/4/2026, <https://www.researchgate.net/publication/251779481>.
- [8] D. Kaminsky, "MD5 to be considered harmful someday," *Cryptology ePrint Archive*, Report 2004/357, 2004.
- [9] Phong, T. Q., Chi, D. D., Huy, T. D., & Diep, N. N. On some issues affecting the security of RSA and ECDSA digital signature schemes. *Journal of Science and Technology on Information Security*, vol.1, no.18, pp.38-46, 2023. DOI: 10.54654/isj.v1i18.884.
- [10] V. T. Cường, N. T. Trung, và L. Đ. Hùng, "Cài đặt thuật toán sinh tham số RSA 2048 bit trên thiết bị PKI Token," *Journal of Science and Technology on Information security*, vol.1, no.11, pp. 53–60, 2020. DOI: 10.54654/isj.v1i11.72.

ABOUT THE AUTHOR



Dinh Quoc Tien

Workplace: Institute of Cryptography Science and Technology, Vietnam Government Information Security Commission.

Email: tienqd77@gmail.com

Education: Academy of Cryptography Techniques

Recent research direction: Cryptography, Information Security.

Tên tác giả: **Đinh Quốc Tiến**

Cơ quan làm việc: Viện Khoa học và Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: tienqd77@gmail.com

Quá trình đào tạo: Kỹ sư mật mã, Học viện Kỹ thuật mật mã, Thạc sĩ mật mã tại Học viện Kỹ thuật mật mã. Tiến sĩ mật mã tại Viện Khoa học – Công nghệ quân sự.

Hướng nghiên cứu hiện nay: Mật mã, An toàn thông tin, Hệ thống thông tin.



Nguyen Le Minh

Workplace: Institute of Cryptography Science and Technology, Vietnam Government Information Security Commission.

Email: nguyenminhhymta2000@gmail.com

Education: Military Technical Academy.

Recent research direction: Cryptography, Information Security.

Tên tác giả: **Nguyễn Lê Minh**

Cơ quan làm việc: Viện Khoa học và Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: nguyenminhhymta2000@gmail.com

Quá trình đào tạo: Kỹ sư An toàn thông tin, Học viện Kỹ thuật quân sự.

Hướng nghiên cứu hiện nay: Mật mã, An toàn thông tin, Hệ thống thông tin.