

# Non-Interactive Provably Secure Attestations for Arbitrary RSA Prime Generation Algorithms

Fabrice Benhamouda   **Houda Ferradi**  
Rémi Géraud   David Naccache

École Normale Supérieure  
Équipe de cryptographie, 45 rue d'Ulm, F-75230 Paris CEDEX 05, France

January 4, 2016

# Context

**The Certification Authority's Cornelian Dilemma:** The Cornelian dilemma is named after French dramatist Pierre Corneille, in whose play *Le Cid* (1636) Rodrigue, is torn between the love of Chimene, or avenging his family, who have been wronged by Chimene's father.

Rodrigue can either seek revenge and lose his beloved, or renounce revenge and lose his honour: thus embodying the Cornelian Dilemma.

All today's Certification Authorities face a Cornelian Dilemma: Either certify potentially insecure RSA keys or... learn the key's factors and hence render these keys insecure.

# This Proposal


Solve the certification authority's Cornelian Dilemma.  
A totally new way to test a match without lighting it!



## Prior Work

- The first *ad hoc* modulus attestation scheme was introduced by Van de Graff and Peralta and consists in proving that  $n$  is a Blum integer without revealing its factors.
- Boyar, Friedl and Lund present a proof that  $n$  is square-free.
- Gennaro, Micciancio and Rabin present a protocol proving that  $n$  is the product of two “quasi-safe” primes<sup>1</sup>
- Camenisch and Michels give an NIZK proof that  $n$  is a product of two safe primes.
- Juels and Guajardo introduce a proof for RSA key generation with verifiable randomness.
- Micali, Boneh, Chan, and Mao describe different protocols proving that  $n$  is the product of two primes  $p$  and  $q$ , without leaking anything on  $p, q$  but their primality.

---

<sup>1</sup>A prime  $p$  is “quasi-safe” if  $p = 2u^a + 1$  for a prime  $u$  and some integer  $a$ . 

# Our Contribution

- A user generates an RSA modulus  $n$  and wants a CA to certify it.
- The CA wants to check that  $n$  was properly generated. We want this done without the CA learning the factors of  $n$ .

A typical application: Users want to check that a PKI's root PK was properly generated before relying on this CA's PKI.

**In general: How to ascertain, before using or certifying an RSA public-key that this public-key was properly generated?**

# Desired Features

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm  $\mathcal{G}$ .
- **Secretless:**  $n$  is provided with an attestation  $\omega_n$  that can be verified by everybody without knowing any secret.
- **Non-interactive:**  $\omega_n, n$  can be checked without any interaction with the creator of  $n$ .
- **Compact:** The size of  $\omega_n$  should be manageable, i.e. polynomial in  $\log n$ .
- **Efficient:** Calculations for creating or verifying an attestation must remain manageable.

## Our approach

The proposed attestation method is based on the following idea: fix  $k \geq 2$ , generate  $k$  random numbers  $r_1, \dots, r_k$  and define  $h_i = \mathcal{H}(i, r_i)$  where  $\mathcal{H}$  denotes a hash function. Let  $p_i = \mathcal{G}(h_i)$  and:

$$N = \prod_{i=1}^k p_i$$

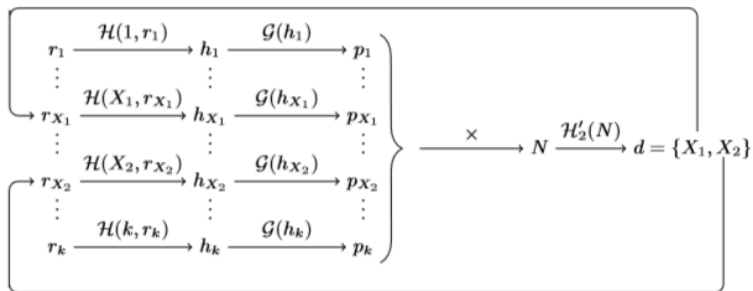
Define  $(X_1, X_2) = \mathcal{H}'_2(N)$ , where  $\mathcal{H}'_2$  is a hash function which outputs two indices  $1 \leq X_1 < X_2 \leq k$ . This defines  $n = p_{X_1} \times p_{X_2}$  and

$$\omega_n = \{r_1, r_2, \dots, r_{X_1-1}, \star, r_{X_1+1}, \dots, r_{X_2-1}, \star, r_{X_2+1}, \dots, r_k\}$$

Here, a  $\star$  denotes a placeholder used to skip one index. The data  $\omega_n$  is called the *attestation* of  $n$ . The algorithm  $\mathcal{A}$  used to obtain  $\omega_n$  is called an *attestator*. The attestation process is illustrated in the next slide.

# The Approach Used to Generate and Validate an Attestation

The choice of the  $r_i$  determines  $N$ , which is split into two parts:  $n$  and  $N/n$ . Splitting is determined by  $d$ , which is the digest of  $N$ , and is hence unpredictable for the opponent.





# The Attestator $\mathcal{A}$

**Input:**  $r_1, \dots, r_k$

**Output:**  $n, \omega_n$

$N \leftarrow 1$

**for all**  $i = 1$  to  $k$  **do**

$h_i \leftarrow \mathcal{H}(i, r_i)$

$p_i \leftarrow \mathcal{G}(h_i)$

$N \leftarrow N \times p_i$

**end for**

$(X_1, X_2) \leftarrow \mathcal{H}'_2(N)$

$\omega_n \leftarrow \{r_1, \dots, r_{X_1-1}, *, r_{X_1+1}, \dots, r_{X_2-1}, *, r_{X_2+1}, \dots, r_k\}$

$n \leftarrow p_{X_1} \times p_{X_2}$

**return**  $n, \omega_n$

# The Validator $\mathcal{V}$

**Input:**  $n, \omega_n$

**Output:** True or False

$N \leftarrow n$

**for all**  $r_i \neq \star$  in  $\omega_n$  **do**

$h_i \leftarrow \mathcal{H}(i, r_i)$

$p_i \leftarrow \mathcal{G}(h_i)$

$N \leftarrow N \times p_i$

**end for**

$(X_1, X_2) \leftarrow \mathcal{H}'_2(N)$

**if**  $r_{X_1} = \star$  and  $r_{X_2} = \star$  and  $\#\{r_i \in \omega_n \text{ s.t. } r_i = \star\} = 2$  **then**

**return** True

**else**

**return** False

**end if**

# The Issue: Efficiency!

Selecting only two primes out of  $k$  makes attacks easy. Indeed, the attacker must only bet on two indices amongst  $k$ , i.e. his success probability is  $\frac{2}{k(k-1)}$ .

In addition, this is the success probability per trial and attestations are not an interactive experiments, hence with sufficient computing power this can be broken even if  $k$  is made very large (say 1,000,000).

To overcome this we propose two approaches:

- Use moduli with more than  $\ell = 2$  prime factors.
- Use more than  $u = 1$  modulus for signing or encrypting the same message.

Different  $(\ell, u)$  parameters allow to reach sufficient security.

# Multi-Factor Moduli

It suffices to have only **two** properly generated factors to make  $n$  secure for RSA encryption and signature.

**Advantage:** Security grows quickly with the number of factors (details in the paper).

**Disadvantage:** Working with bigger moduli slows-down multiplications quadratically (and exponentiations cubically). We hence buy security at the price of slower execution.

## Using Several Moduli

It suffices to have only **one** properly generated modulus to perform secure RSA encryption and signature.

**Advantage:** Again; security grows quickly with the number of moduli (details in the paper).

**Disadvantage:** Working with more moduli slows-down calculations linearly. We hence buy again security at the price of slower execution.

**Note:** For signature we sign  $u$  times the same message using different  $u$  moduli. But for encryption we share a secret key  $\kappa = \kappa_1 \oplus, \dots, \oplus \kappa_u$  and encrypt each share  $\kappa_j$  with a different modulus (the idea is that at least one share gets protected by a properly generated modulus).

# The Attestator $\mathcal{A}$ for More Than $u = 1$ Modulus

Variant consisting in generating  $u = \ell/2$  bi-factor moduli for some even number  $\ell$  of factors.

**Input:**  $r_1, \dots, r_k$

**Output:**  $\mathbf{n} := (n_1, \dots, n_u), \omega_{\mathbf{n}}$

$N \leftarrow 1$

**for all**  $i \leftarrow 1$  to  $k$  **do**

$h_i \leftarrow \mathcal{H}(i, r_i)$

$p_i \leftarrow \mathcal{G}(h_i)$

$N \leftarrow N \times p_i$

**end for**

$(X_1, \dots, X_{2u}) \leftarrow \mathcal{H}'_{2u}(N)$

$\omega_{\mathbf{n}} \leftarrow \{r_1, \dots, r_{X_1-1}, *, r_{X_1+1}, \dots, r_{X_{u\ell}-1}, *, r_{X_{u\ell}+1}, \dots, r_k\}$

**for all**  $j \leftarrow 1$  to  $u$  **do**

$n_j \leftarrow p_{X_{2j}} \times p_{X_{2j+1}}$

**end for**

**return**  $\mathbf{n} := (n_1, \dots, n_u), \omega_{\mathbf{n}}$

# General Attestation Scheme

Attestator  $\mathcal{A}$  for the General Attestation Scheme ( $u \geq 1, \ell \geq 2$ )

**Input:**  $r_1, \dots, r_k$

**Output:**  $\mathbf{n} := (n_1, \dots, n_u), \omega_{\mathbf{n}}$

$N \leftarrow 1$

**for all**  $i \leftarrow 1$  to  $k$  **do**

$h_i \leftarrow \mathcal{H}(i, r_i)$

$p_i \leftarrow \mathcal{G}(h_i)$

$N \leftarrow N \times p_i$

**end for**

$(X_1, \dots, X_{u\ell}) \leftarrow \mathcal{H}'_{u\ell}(N)$

$\omega_{\mathbf{n}} \leftarrow \{r_1, \dots, r_{X_1-1}, *, r_{X_1+1}, \dots, r_{X_{u\ell}-1}, *, r_{X_{u\ell}+1}, \dots, r_k\}$

**for all**  $j \leftarrow 1$  to  $u$  **do**

$n_j \leftarrow p_{X_{(\ell-1)j+1}} \times \dots \times p_{X_{\ell j}}$

**end for**

**return**  $\mathbf{n} := (n_1, \dots, n_u), \omega_{\mathbf{n}}$

# Parameters

Refer to the tables in the handouts.



# Compressing The Attestation

