Magnetic RSA

Rémi Géraud-Stewart[1,2] and David Naccache[2]

[1] QPSI, Qualcomm Technologies Incorporated, USA
rgerauds@qti.qualcomm.com
[2] ÉNS (DI), Information Security Group
CNRS, PSL Research University, 75005, Paris, France
david.naccache@ens.fr

Abstract. In a recent paper Géraud-Stewart and Naccache [GSN21] (GSN) described an non-interactive process allowing a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a modulus n is the product of two randomly generated primes (p, q) of about the same size. A heuristic argument conjectures that $\mathcal{P}$ cannot control p, q to make n easy to factor.
GSN's protocol relies upon elementary number-theoretic properties and can be implemented efficiently using very few operations. This contrasts with state-of-the-art zero-knowledge protocols for RSA modulus proper generation assessment.
This paper proposes an alternative process applicable in settings where $\mathcal{P}$ co-generates a modulus $n = p_1q_1p_2q_2$ with a certification authority $\mathcal{V}$. If $\mathcal{P}$ honestly cooperates with $\mathcal{V}$, then $\mathcal{V}$ will only learn the sub-products $n_1 = p_1q_1$ and $n_2 = p_2q_2$.
A heuristic argument conjectures that at least two of the factors of n are beyond $\mathcal{P}$'s control. This makes n appropriate for cryptographic use provided that *at least one party* (of $\mathcal{P}$ and $\mathcal{V}$) is honest. This heuristic argument calls for further cryptanalysis.

1 Introduction

Several cryptographic protocols rely on the assumption that an integer $n = pq$ is hard to factor. This includes for instance RSA [RSA78], Rabin [Rab79], Paillier [Pai99] or Fiat–Shamir [FS86]. One way to ascertain that n is such a product is to generate it oneself; however, this becomes a concern when n is provided by a third-party. This scenario appears e.g. with Fiat–Shamir identification, or in the context of certificate authentication: carelessly using an externally-provided n may compromise security if n happens to be cryptographically weak. Naturally, one cannot ask for the factors of n to check n.

The state of the art in the matter are the zero-knowledge procotols of Auerbach–Poettering [AP18] and Camenisch–Michels [CM99], which prove that a given n is the product of safe primes of prescribed size. While correct and very useful, these protocols are difficult to implement and analyze, and have high computational costs. This motivates the search for simpler and more efficient solutions.

In [GSN21], Géraud-Steward and Naccache (GSN) introduced an alternative protocol that nearly achieves the same functionality with fewer operations and communication. GSN is also simpler to understand and thus to implement. However, GSN relies on a number-theoretical conjecture that remains so see further analysis.

This paper proposes an alternative protocol, applicable in settings where a user $\mathcal{P}$ co-generates a modulus $n = p_1q_1p_2q_2$ with a certification authority (CA, denoted $\mathcal{V}$). If $\mathcal{P}$ is honest then $\mathcal{V}$ will only learn the sub-products $n_1 = p_1q_1$ and $n_2 = p_2q_2$. A new heuristic argument claims that at least two of the four prime factors of n are beyond $\mathcal{P}$'s control, thereby making n proper for cryptographic use.

1.1 Difference with GSN

It is important to underline the difference between the construction proposed in this paper and GSN. This paper's reason to be is the following: In GSN it is conjectured that when generating an Esther modulus, $\mathcal{P}$ cannot manipulate p and $\mathcal{P}$ cannot manipulate q. What would happen if one day a method allowing to manipulate only one of the factors is found? In such a case the method proposed in this paper is conjectured to still survive provided that the CA still does its job.

$\mathcal{P}$ can control	GSN	This paper
no factor	resists	resists
one factor	broken	resists
two factors	broken	broken

Table 1. Consequences of different cryptanalysis scenarios.

2 Preliminaries & Building Blocks

Notations. This paper uses the following notations: $a|b$ denotes the concatenation of the bitstrings a and b. If c is an integer, $\|c\| = \lceil \log_2 c \rceil$ denotes the size of c, i.e. the minimal number of bits needed to write c.

Certification. A certification authority (CA, here $\mathcal{V}$) possesses a long-term signing key allowing $\mathcal{V}$ to vouch for a third party's public key. They do so by producing a certificate, which contains identifying information and which is signed using the $\mathcal{V}$'s private key. Schematically, this is an algorithm $\mathsf{Certify}(n)$ that can only be run by $\mathcal{V}$, and which outputs a publicly-verifiable certificate. The corresponding verification algorithm is denoted Verify.

GSN with prescribed MSBs. The protocol in [GSN21] proves that a modulus $n = pq$ has exactly two factors of about the same size, and checks that the most significant bits of n feature a given pattern N_H.

We use the following notation: $\mathsf{Esther}(n, N_H)$ returns True if n is accepted by the GSN protocol with most significant bits N_H.

Finally, using the techniques discussed in [GSN21], there is an randomized modulus generation algorithm G taking as input N_H with $\|N_H\| = 2\|n\|/3$ and returning n such that $\mathsf{Esther}(n, N_H) = \mathsf{True}$.

Secure channel with shared randomness. We assume that all communications between parties happen through secure channels; since our protocol requires a shared session randomness, we can hit two birds with one stone as follows: $\mathcal{P}$ and $\mathcal{V}$ run an authenticated Diffie–Hellman key exchange [DH76] to obtain a shared value K. Then they use a secure key derivation algorithm to obtain symmetric keys (ensuring the channel's confidentiality and integrity). They further use this key derivation algorithm to generate two numbers $N_{H,1}$ and $N_{H,2}$. These numbers are under the complete and sole control of neither party and are kept secret. We write generically $N_{H,1}|N_{H,2} := \mathsf{Exchange}$.

3 The New Modulus Co-Generation Process

The intuition behind the proposed protocol is illustrated (see Figures 1 and 2) by an analogy consisting in attaching (multiplying) four "magnets" (primes p_1, q_1, p_2, q_2) to form a larger "magnet" (n).

The magnet generator $\mathcal{P}$ may cheat and provide, instead of an elementary magnet, a simple piece of iron. We assume that an iron-magnet link is too weak and breakable but that a magnet-magnet link is resistant. We hence ask $\mathcal{P}$ to generate two magnet-magnet pieces. Should $\mathcal{P}$ cheat, we would still have two magnets in n and hence have at least a part of n (sub-factor) which is hard to factor.

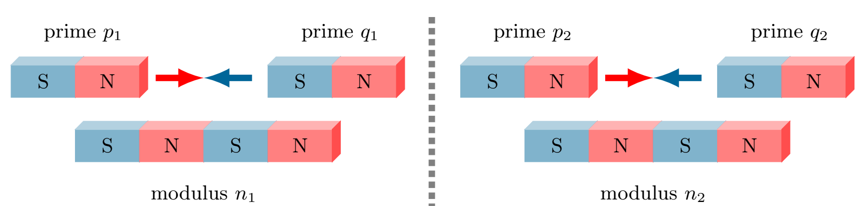

Fig. 1. The magnet analogy: preparing n_1 and n_2.

3.1 Protocol

As mentioned in Section 2 we assume that all exchanges between $\mathcal{P}$ and $\mathcal{V}$ are properly secured against passive attackers — this is standard and the precise

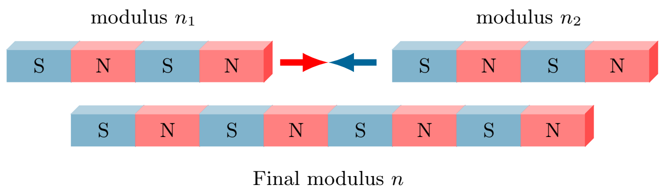

Fig. 2. The magnet analogy: forming $n = n_1 n_2$. Even if p_1 and p_2 are weak primes (e.g., with $p_i - 1$ smooth), the existence of q_1 and q_2 in n makes n cryptographically useful.

way in which this is achieved in practice is beyond the scope of our paper. The protocol is then the following, where a failure of any "$\stackrel{?}{=}$" results in a failure of the overall protocol.

Prover $\mathcal{P}$		**Verifier $\mathcal{V}$** (CA)
$N_{H,1} \vert N_{H,2} := \mathsf{Exchange}$		$N_{H,1} \vert N_{H,2} := \mathsf{Exchange}$
$n_1 := G(N_{H,1})$	$\xrightarrow{n_1}$	$\mathsf{Esther}(n_1, N_{H,1}) \stackrel{?}{=} \mathsf{True}$
$n_2 := G(N_{H,2})$		
$\gcd(n_1, n_2) \stackrel{?}{=} 1$	$\xrightarrow{n_2}$	$\mathsf{Esther}(n_2, N_{H,2}) \stackrel{?}{=} \mathsf{True}$
$n := n_1 n_2$		$n := n_1 n_2$
$\mathsf{Verify}(n, c) \stackrel{?}{=} \mathsf{True}$	$\xleftarrow{c}$	$c := \mathsf{Certify}(n)$

3.2 Rationale

The rationale behind this procedure is the following:

- In case an attack on G allowing to generate vulnerable moduli is found, it is likely that such an attack would not affect *both factors* but only one which would remain controllable by the attacker. The process ascertains that at least one "good" factor in n_1 and one "good" factor in n_2 would result in n being partially factoring-resistant (except with respect to the CA $\mathcal{V}$) with respect to external attackers and hence cryptographically useful.
- We require the prescribed patterns $N_{H,i}$ to be secret[3] and random to avoid partially factoring n given the information that its factors n_1 and n_2 share known bit patterns.

3.3 Security

Let us now discuss three different scenarios.

- $\mathcal{P}$ *is honest:* A third party receiving n gets no useful information on the factors of n. $\mathcal{V}$ only learns the sub-factors n_i which are insufficient for allowing $\mathcal{V}$ to sign messages on $\mathcal{P}$'s behalf nor decrypt ciphertexts sent to $\mathcal{P}$.

[3] With respect to the external world.

– $\mathcal{P}$ *is dishonest and* $\mathcal{V}$ *is honest:* $\mathcal{P}$ attempts to generate a modulus which is easy to factor. The exchange of commitments at the beginning of the protocol reduces[4] $\mathcal{P}$'s control over the sub-moduli n_i. It is conjectured that this constraint prevents $\mathcal{P}$ from fixing the n_i to some constant value while passing the Esther test or prevents $\mathcal{P}$ from simultaneously increasing the smoothness of both $p_i - 1$ and $q_i - 1$. $\mathcal{P}$ may however reveal his own random tape of G or use a low-entropy random tape and thereby enable everybody to re-generate the factors but this is tantamount to purposely revealing the factors[5], a misbehavior against which nothing can be done in general.
– $\mathcal{P}$ *and* $\mathcal{V}$ *are both dishonest*: This scenario does not make any practical sense, e.g., $\mathcal{P}$ could just publish all his factors or $\mathcal{V}$ could certify any easy to factor modulus.

$\mathcal{P}$	$\mathcal{V}$ (CA)	$\mathcal{V}$ learns	Externals learn	n is:
honest	any behavior	n_1, n_2	no factors of n	usable
dishonest	honest	all factors of n	only weak factors of n	usable
dishonest	dishonest	all factors of n	all factors of n	broken

Table 2. Consequences of different honesty settings. The table shows that is if *at least* one party remains honest then n remains safe for cryptographic use with respect to the external world.

3.4 Efficiency

The global computational cost of our protocol is dominated by two calls to G, as other phases essentially have the cost of a few full-sized modular multiplications. The cost of running G is typically cubic (up to logarithmic factors), and therefore our protocol's complexity is $O(k \log^3 n \log\log n)$, where n is the modulus and k is the number of primality testing rounds determining the primality probability. The additional $\log\log n$ factor is the density of primes among integers of size $\log n$.

3.5 Hierarchical Attestation

Note that the scheme can be made hierarchical: i.e. a PKI where CAs of level i magnetically co-generate moduli with CAs of level $i-1$ and certify them. In other words, the modulus n_i (used by the father node to certify the son's n_{i+1}) is magnetically co-generated with the grandparent node and certified using the grandparent's n_{i-1}. Note that n_0 must be attested using some other means, for instance [GSN21] or [CM99].

[4] Reduces but does not eliminate! Indeed, $\mathcal{P}$ must be given the possibility to inject randomness into the random tape of G.
[5] Rather than just weakening the key - the risk we want to defend against

4 Conclusion

This paper introduced an inexpensive interactive process allowing a CA to co-generate a modulus $n = p_1q_1p_2q_2$ with a user $\mathcal{P}$. Under the conjecture formulated in [GSN21], n contains at least two uncontrolled prime factors which makes it appropriate for most factoring-based cryptographic applications. Because this assumption is core to security, we encourage the community to carefully scrutinize this protocol before considering its adoption.

References

AP18. Benedikt Auerbach and Bertram Poettering. Hashing solutions instead of generating problems: On the interactive certification of RSA moduli. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 403–430. Springer, 2018.

CM99. Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 1999.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

GSN21. Rémi Géraud-Stewart and David Naccache. Elementary attestation of cryptographically useful composite moduli. In Diana Maimut, Andrei-George Oprina, and Damien Sauveron, editors, *Proceedings of the 13th International Conference on Security for Information Technology and Communications (SECITC 2020), November 19-20*, Lecture Notes in Computer Science. Springer, 2021.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

Rab79. Michael O. Rabin. Digitalized signatures and public key functions as intractable as intractable as factorization. *MIT Laboratory of Computer Sciences*, 21, 1979.

RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.