Partial Key-Hiding in RSA

Eabhnat Ní Fhloinn[†] **and Michael Purser**

School of Mathematics
Trinity College Dublin
IRELAND

E-mail: [†]evoflynn@maths.tcd.ie

Abstract — **This paper explores the idea of partially exposing sections of the private key in public-key cryptosystems whose security is based on the intractability of factorising large integers. It is proposed to allow significant portions of the private key to be publicly available, reducing the amount of data which must be securely hidden. The "secret" data could be XORed with an individual's biometric reading in order to maintain a high level of security, and we suggest using iris templates for this purpose. Finally, we propose an implementation of this system for RSA, and consider the potential risks and advantages associated with such a scheme.**

Keywords — **Public-key cryptography, Partial key exposure, Partial hiding, RSA, Iris, Biometrics**

I Introduction

The concept of public-key cryptography (PKC) first came to the fore in April 1976, with the publication of Whitfield Diffie's and Martin Hellman's appropriately titled paper "New Directions in Cryptography" [8]. However, it has since emerged that earlier work was secretly done in this area as early as 1970 in the Communications-Electronics Security Group (CESG) of GCHQ in the United Kingdom [9]. The idea behind PKC represented a major breakthrough for modern cryptography. Each user in a public-key system has two keys - a public key, which is openly available, and a private key, which is kept secret at all times. In order to send an encrypted message, an individual merely looks up the public key of their intended recipient, encrypts the message with this key, and sends it over an open channel. Upon receipt, the recipient decrypts the message using his private key. As no other key will decrypt the message, and the private key is always kept secret, this should provide security and privacy to both individuals.

Most of the major public-key algorithms in use today rely for their security on either of two intractable mathematical problems - factorising large integers or the discrete logarithm problem.

Problem I.1 (Integer Factorisation Problem) *Given a positive integer n, find its prime factorisation i.e. write $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.*

Problem I.2 (Discrete Log Problem) *Given integers p, α and y, find x such that $y \equiv \alpha^x \bmod p$.*

Most modern cryptography books cover the security afforded by these problems in detail, notably [18].

a Partial Hiding of Private Key

In public-key systems which rely on the intractability of factorising large numbers, it is generally recommended that private keys be of length at least 1024 bits. These private keys are securely stored and hidden from public access. In this paper, we address the question of whether all these bits need to be secret, or if it would be sufficient to store a portion securely and leave the remainder publicly available.

In mathematical notation, this idea could be expressed by letting P be the private key of length p, S be the bits not hidden and X be the bits hidden. Using the XOR operation to combine these, we have

$$P = S' \oplus X' \quad , \tag{1}$$

where S' and X' represent bit-strings of length p, comprising S or X respectively but padded with zeros in positions where bits were extracted. For example, if P were 10 bits long, and X consisted of the four "middle" bits, then we would have

$$P = S_1 0000 S_2 \oplus 000X000 \quad . \tag{2}$$

It is vital that X is sufficiently large to render a brute force attack impractical. This means that if an attacker has access to the public key, the algorithm and S, he should not be able to determine the private key P by trying all possible values of X.

There are several ways in which the bits of X could be extracted from P: we could remove the least significant bits (LSBs); the most significant bits (MSBs); or a random scattering of bits from throughout the key. Randomly selected bits would seem initially to give a greater level of security. However, if the bits are randomly selected according to some secret rule, then we need to also store their original position in the key in order to recover them later on. Storing these positions, as well as the values of the bits, within the n bits allowance we choose for X reduces the overall number of bits of the private key that can be hidden thus - meaning that we could only store $n/2$ bits of the key and $n/2$ positions, instead of n bits of the private key, which is a clear disadvantage.

Although there has been substantial work done to date in the area of partial key exposure, it would appear that the aim of this work has been to investigate the implications of inadvertant exposure of some part of the private key. We term our scheme *partial hiding*, focusing instead on the potential advantages of deliberately revealing portions of the private key.

While partial hiding in a public-key cryptosystem may prove advantageous in schemes which rely on the intractability of factorising large integers for their security, the same cannot be said for discrete-log based systems. The reason for this is that most schemes which rely solely on the discrete-log problem for their security require that the private key is a random number of length 160 bits. Firstly, this is not overly large and secondly, it needs to be of this length in order to frustrate brute-force attacks on the key. Thus, shortening it in any way would seem to leave it vulnerable to this most basic of attacks. Ultimately, even if partial hiding was possible with such systems, the savings in storage space would be negligible and therefore we will not consider such schemes any further.

There are several issues which need to be addressed in the proposed idea of partial key hiding. The most important of these is how many bits it is safe to expose, and exactly which bits these might be. However, discussions about security are of no use if the "secret" portions of the private key are not securely stored in such a way that an attacker cannot gain access to them. We suggest the use of biometrics with error-correcting codes to overcome this problem.

b Biometrics

A biometric measures an individual's unique physical or behavioural characteristics in order to recognize or authenticate his identity [16]. Currently, physical rather than behavioural characteristics are more commonly used for biometric identification. Table 1 compares desirable characteristics of various biometrics: *Universality* means that each person must possess the characteristic; *Uniqueness* implies that no two people share an identical trait; *Permanence* refers to the stability of the characteristic, meaning that it neither changes nor can be changed; *Collectability* indicates that the biometric is easily measured and clearly presentable to a sensor; *Performance* assesses the system's accuracy, speed and robustness; *Acceptability* judges the extent to which people are willing to accept the biometric as an identifier; *Circumvention* measures how easy it is to fool the system.

Biometric identification is made up of two stages: *enrolment* and *verification* [26]. Enrolment refers to the process of obtaining several readings of the biometric in question from a new user and extracting notable features to store in a template for comparison purposes later on. Verification consists of acquiring a new reading from the user and comparing the result with the previously-stored template in order to accept or reject the user.

Clearly the accuracy and effectiveness of biometric identification is restricted by numerous issues, such as the precision of the biometric sensor (particularly in a practical day-to-day setting); distortions in the image due to sweat, dirt, moisture content, temperature, lighting or pressure; distortions due to the positioning and orientation of the biometric; perceivable variations between different individuals' biometrics, and so on.

Although we originally considered working with fingerprints, which are perhaps the most widely accepted biometric in use today, we concluded that iris templates were better suited to our purposes. Unlike fingerprints, the iris benefits from a high level of natural protection in the body [6], being unaffected by dirt, weather conditions, sweat and various other external influences which can adversely affect a fingerprint reading. Although it is an internal organ, it is externally visible, allowing for unintrusive imaging from a distance. It also contains an intrinsic polar geometry leading to an ease of applying a natural coordinate system that is lacking in other biometrics [5]. An in-depth report was produced by the U.K. National Physical Laboratory in 2001 [17], following a detailed study of the performance and reliability of various biometrics. In 2.73 million cross-comparisons, there were no false matches reported for the iris recognition technology. This impressive reliability demonstrates that the iris is probably the most suitable biometric for use in public-key systems.

In our system, most of the private key will be openly available, but a small portion of it will be

Biometrics	Face	Finger	Hand	Iris	Retina	Sign	Voice	FTherm
Universality	High	Medium	Medium	High	High	Low	Medium	High
Uniqueness	Low	High	Medium	High	High	Low	Low	High
Permanence	Medium	High	Medium	High	Medium	Low	Low	Low
Collectability	High	Medium	High	Medium	Low	High	Medium	High
Performance	Low	High	Medium	High	High	Low	Low	Medium
Acceptability	High	Medium	Medium	Low	Low	High	High	High
Circumvention	Low	High	Medium	High	High	Low	Low	High

Table 1: Summary of the relative merits of various biometrics under different headings, as outlined by Jain, Hong and Pankanti [11]. "Sign" here refers to a signature; "FTherm" indicates a facial thermogram

stored securely. This secret portion will be XORed with an individual's iris template and thus, the user must submit to an iris reading to retrieve his entire secret key for a transaction. With an appropriate system design, this could provide a high level of security, as it would be extremely difficult for an attacker to recreate the biometric necessary to uncover the "hidden" portion of the private key.

II Related Work

The algorithm currently in use for almost all iris recognition systems was developed by John Daugman [5]. His approach is not feature-based, which would suffer from the disadvantage that readings would be of various lengths and possibly describe different features. Instead he uses a phase sequence description of the iris, meaning that the length is always the same, regardless of the image quality.

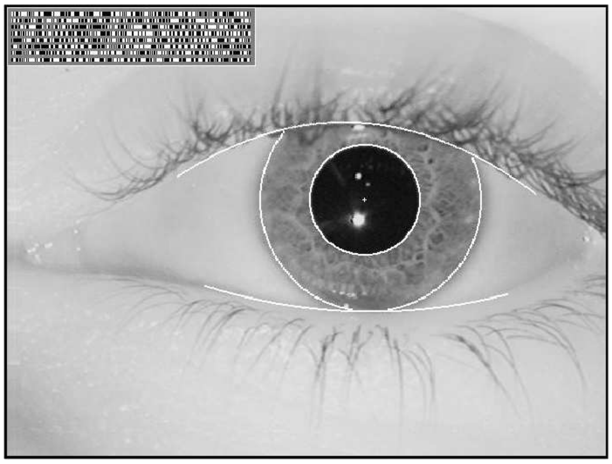

Fig. 1: Example of an iris image segmented using Daugman's algorithm, with the IrisCode (bit stream) visible in the top left corner. The image has been taken monochromatically with near-infrared illumination in the 700-900nm band at a distance of approximately 35cm. [5]

An image of the user's eye is obtained and the iris is segmented from the overall image, as illustrated in Fig. 1. An "IrisCode" is then calculated based on a phase sequence description of the iris. This IrisCode is 256 bytes in length and is accompanied by a further 256 "masking" bytes which provide information as to which bits of the image have been corrupted and should be ignored. In order to perform iris recognition, the IrisCodes from two irises are compared using a test of statistical independence. If two different irises are used, this test will be passed - if two IrisCodes computed from the same iris are compared, the test will be failed. Thus, in order to successfully identify an individual, this statistical test needs to be *failed*.

The use of error-correcting codes to correct iris readings and compare them with previously stored templates was suggested by Davida, Frankel and Matt [7]. This scheme increases security by storing the template's check digits along with a hash of the template, but currently their implementation appears to lack sufficient error-tolerance. We believe that this could be improved by introducing the use of erasures and propose to investigate this further in the near future.

In 1998, Boneh, Durfee and Frankel [2] explored several different attacks on RSA following partial exposure of the private key. Interestingly, for low-exponent RSA, they showed that the most significant half of the bits of the private key d are automatically "leaked" and that this has no adverse effect on security. Their results for low-exponent RSA also show that only a quarter of the LSBs of d are sufficient for an attacker to obtain all of d.

However, Steinfeld and Zheng [27] went on to show that this result only holds if no more than the least significant bit of p and q is equal, where p and q are primes such that the modulus $N = pq$. This led them to propose a system for low-exponent RSA in which the m LSBs of p and q are identical. In this case, they prove that, if the system is secure with no bits exposed, it is secure if up to $2m$ LSBs of d are exposed. They go on to suggest that this result may be used to some advantage to reduce the "computational bottleneck" of the decryption operation in RSA.

Hinek, Low and Teske [10] investigate multi-prime RSA (i.e. 3 primes or more) in relation to partial exposure of the private key. They claim that the BDF attack is completely ineffective for

3-prime RSA as the number of solutions is exponential in the bit-size of the RSA modulus.

Most recently, Blömer and May presented a paper entitled "New Partial Key Exposure Attacks on RSA" [1], in which they produce even stronger results for RSA than those suggested by BDF. In addition, they also produce attacks on partial key exposure in the Chinese Remainder Theorem (CRT) version of RSA [21], suggesting that for low-exponent RSA, half the bits of $d_p = d \bmod (p-1)$ are sufficient to factorize the modulus N.

Canetti et al [3] proposed the idea of *All-Or-Nothing Transforms* (AONTs) and *Exposure-Resilient Functions* (ERFs) to protect a secret key even when *almost all* of the secret key has been exposed. An AONT is defined [3] as an efficiently computable transformation T on strings such that

- for any string x, given *all* of $T(x)$, one can efficiently recover x
- there exists some threshold l such that any polynomial-time adversary that (adaptively) learns all but l bits of $T(x)$ obtains *no* information about x.

From this, an ERF is described as being "a deterministic function whose output appears random even if *almost all* the bits of the input are revealed." Instead of storing a secret key as usual, the key is stored with the AONT applied to it. If an AONT can be created with the threshold value l far lower than the size of the output of the AONT, then security against almost total exposure can be obtained. In order to accomplish this, the AONT is based on a suitable ERF. There are some similarities to be noted between this idea and our scheme, as we XOR the secret portion of the private key with the iris template and then store this "transformation".

III RSA Cryptosystem

RSA came to public attention in 1977, deriving its name from its three inventors Rivest, Shamir and Adleman [23]. It can be used both for encryption and digital signatures, and its security relies on both the intractability of factorising large integers and the discrete log problem.

Two large distinct prime numbers p and q are chosen randomly. For maximum security, p and q should be of the same length. The product $N = pq$ is computed and this value N is called the *modulus*. Then the encryption key, e, is chosen at random such that e is relatively prime to $(p-1)(q-1)$. Using the Euclidean Algorithm, the decryption key, d, is computed such that

$$ed \equiv 1 \bmod (p-1)(q-1) \ . \tag{3}$$

Now e and N form the public key, while d is the private key. It is of great importance that p and q be kept secret. They can be discarded at this stage if so desired - but retaining their values can help to speed up private key operations in the algorithm, using the Chinese Remainder Theorem [21].

If we let m be the message we wish to encrypt and c be the corresponding ciphertext, then encryption merely consists of

$$m^e \bmod N = c \ . \tag{4}$$

Decryption is accomplished by raising the ciphertext c to the power of the private key d which gives us

$$\begin{aligned} c^d &= m^{ed} \\ &= m^{1+k(p-1)(q-1)} \\ &= m \bmod N \ . \end{aligned} \tag{5}$$

In the case of digital signatures, the sender typically calculates the hash $H(m)$ of his message and then sends the message together with the signature

$$H(m)^d \bmod N \ . \tag{6}$$

In order to validate the signature, the recipient raises it to the power $e \bmod N$ to get

$$\left(H(m)^d\right)^e = H(m) \bmod N \ , \tag{7}$$

which he compares with the hash of the received message. If these are equal, the signature is taken to be valid.

The intractability of factorising large integers means that the modulus N cannot be factorised in this algorithm if it is correctly implemented. In the encryption process, the discrete log problem ensures that, even though the sender knows that $c^d = m \bmod N$, and possesses c, m and N, he cannot determine d, the receiver's private key. Discrete logs also feature in the security of the digital signature scheme, as the signature $H(m)^d$ is sent to the receiver along with $H(m)$. Were it not for the discrete log problem, the receiver would be able to calculate the sender's private key d from this information.

a Frustrating General Attacks on RSA

Possible attacks on RSA are well-documented [13] and strict guidelines as to the implementation of the algorithm are in place to counteract these attacks. RSA Laboratories' latest recommendations [12] estimate that RSA moduli of minimum size $n = 1024$ bits are "safe" until at least 2010 against such powerful factorising algorithms as the quadratic sieve [20] and the number field sieve [19]. However, they also note that *"A larger minimum may well be appropriate for enterprise and root keys, a distinction that is already reflected in practice through the common use of 2048-bit root keys today."*

It has been established that factorising the modulus, N, and computing the decryption key d from the public key (N, e) are computationally equivalent [23]. For this reason, it is also important that the primes p and q, whose product creates the modulus, are carefully selected so that factorising N is infeasible. It is important that p and q be large and about the same bitlength, to frustrate the elliptic curve factoring algorithm [15]. At the same time, the difference between p and q should not be too small, to avoid factorisation by trial division. Otherwise, if $p - q$ is small, then

$$\begin{aligned} p &\approx q \\ \Rightarrow p &\approx \sqrt{N} \end{aligned} \tag{8}$$

and an attacker could factorise N with Fermat's factorisation technique, using the fact that $x^2 - y^2 = (x+y)(x-y)$ in order to find $x^2 - y^2 = N$. The requirement for p and q to be "strong primes" is overcome provided p and q are sufficiently large [24].

While it is possible to choose a small value for the encryption key e, the same is not true for the decryption key d. An attack due to Wiener [28] shows that if d is small compared to the modulus ($d < N^{1/4}$), then it can be successfully computed from the public key (N, e), using continued fractions. This attack is not possible if the decryption key is about the same size as the modulus N.

b *Chosen Parameters for Partial Hiding*

Let $N = pq$ be an RSA modulus of size $n = \log_2 N = 2048$ bits, with p and q each 1024 bits long. We wish to choose a "small" exponent e, but common values previously used, such as $e = 3$ or $e = 17$, are no longer considered safe [14]. Thus, we let $e = 2^{16} + 1 = 65537$. From the paper produced by Boneh, Durfee and Frankel [2, pp 6], we know that the most significant 1024 bits of the private key d are automatically "leaked" in low-exponent RSA such as this, but this does not affect the security of the system.

We now base our system on an adaptation of that proposed by Steinfeld and Zheng [27], setting the m least significant bits (LSBs) of p and q to be equal. They claim that if low-exponent RSA in this form is secure with *no* bits exposed, then it is secure if up to $2m$ LSBs are exposed. In addition to this, we know (from the Boneh-Durfee-Frankel paper) that the most significant half of the bits can also be exposed in this case, with no adverse effect on security. Thus, if we let

$$m = \frac{n}{4}(1 - \epsilon) \tag{9}$$

with small ϵ in a secure system, we can expose the $n/2$ (=1024) MSBs of d and the $(n/2)(1 - \epsilon)$ ($= 1024 - 1024\epsilon$) LSBs of d and still have a secure system.

We must now decide on an optimum value for ϵ. We want $\epsilon < 0.5$, as Steinfeld and Zheng found that there is a considerable reduction in the cost of computation if this is true. However, we also need to ensure that $2^{\frac{\epsilon n}{4}}$ is too large to exhaustively search for $(\epsilon n/4)$ unknown bits of p or q.

Based on Silverman's estimates [25] on the costs of breaking cryptographic keys, with a budget of \$10 million dollars, 2^{56} takes < 5 minutes to crack; 2^{80} takes 600 months; 2^{96} takes 3 million years; 2^{128} takes 10^{16} years. From Table 2, we observe that our optimum value for ϵ is 0.25, as we reject 2^{64} as too low a value for security. The current standards for symmetric cryptography suggest to use 128-bits for AES, so this value for ϵ would provide comparable security. Thus, if we choose $\epsilon = 0.25$, we get $m = 384$, and a value of 2^{128}, which would seem to suit our purposes well while still being sufficiently large to be considered safe.

This means that the 384 LSBs of p and q will be equal. In order to generate these primes, simply find p in the usual fashion, fix the 384 LSBs of q to be identical to p, and produce a prime q of this form. Steinfeld claims that this is expected to be as efficient as the standard independent primes generation algorithm for random RSA moduli, where each candidate for q is chosen independently of p as a random odd integer.

If we let $\epsilon = 0.25$, then the 1024 MSBs and the 768 LSBs of d can be exposed for low-exponent RSA, and if the original system was secure, then this new system should also be. This means we need to keep 256 bits securely hidden at all times, in bit positions from 769 to 1024 inclusive, assuming an offset of 1 and working from right to left.

IV CRT RSA Partial Hiding

While considering improvements of the RSA system, it is natural to attempt to implement these using the Quisquater-Couvreur [21] version of RSA, which uses the Chinese Remainder Theorem to speed up calculations in the scheme. In this approach, two separate quantities

$$\begin{aligned} d_p &= d \bmod (p-1) \ , \\ d_q &= d \bmod (q-1) \end{aligned} \tag{10}$$

are used to encrypt and decrypt each message. This means that if we have either p or q, we should be able to recreate all the remaining information needed from our public information. Thus, while we need to retain the values of p and q, we cannot make these publicly known in their entirety. Therefore, both of these values need to be at least partially hidden.

There are two main approaches to partial hiding to be considered:

Size of N	Size of p (or q)	Value of ϵ	Value of m	Value of $2^{\frac{\epsilon n}{4}}$
2048	1024	0.5	256	2^{256}
2048	1024	0.25	384	2^{128}
2048	1024	0.125	448	2^{64}

Table 2: Possible values for ϵ

1. Partially store p and q. Partially store d_p and d_q. Restore these hidden portions to the "known" public portions of d_p and d_q each time.

2. Partially store p and q. Partially store d. Generate $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ each time using both partially stored and publicly known information.

If we consider option 1, there are certain restrictions that must be imposed immediately - it is important that less than 50% of the LSBs (or MSBs) of d_p or d_q are made public, as Blomer-May [1] detail an attack on CRT RSA if more than this amount is exposed, for low public exponent e, such as $e = 2^{16} + 1$.

If we deal with an RSA modulus of size $n = 2048$, then p and q are of length 1024 bits and d_p and d_q are approximately this size also. The 384 LSBs of p and q are identical in our system; the 640 MSBs are not. However, Coppersmith [4] describes a method of factoring N given the MS half of the bits of p. Thus, it would appear to make sense to securely hide the 640 MSBs of p and q using the iris template. We must also store over half of the bits of d_p and d_q, which means that, in total, we would need to store over 2304 bits of data in this scheme. If we wish to use iris templates to encrypt this data, we encounter problems at this point, as an iris template consists of only 2048 bits of data, and so a direct XOR with the RSA information is not possible. Thus, we can only continue with this particular approach if we reduce the size of our modulus and deal with 1024-bit RSA in place of 2048-bit RSA, or possibly consider other biometrics.

If we turn our attention now to the second approach, we again choose to partially hide the 640 MSBs of both p and q. As suggested above for RSA, we store only the 256 "middle" bits of d, which gives us a total of 1536 bits to be stored, which could easily be hidden by XORing with an iris template.

We must now consider whether we have introduced additional vulnerability to the system by partially revealing p and q, and if so, to what extent this affects the overall system. As before, bits 1-768 and 1025-2048 of d are made public, with only bits 769-1024 securely hidden. We are proposing to reveal bits 1-384 of p (and q). If it is possible to determine either the least significant half or the most significant half of the bits of d_p as a result, then our system is not secure.

p is prime, so the least significant bit of p will always equal 1. Thus, to obtain $(p-1)$, we must simply change this least significant bit to 0. Let p' be the 384 LSBs of p that are publicly available. We now show in a general way that there is no obvious link between the LSBs of a random number, x, calculated $\bmod(p-1)$ and $\bmod(p'-1)$. This ensures that the LSBs of d_p could not be directly calculated from the partial knowledge of p and d that we have exposed.

Let

$$p = t(2^{384}) + p' \ , \tag{11}$$

where p and t are not publicly known. Clearly, this means that

$$p - 1 = t(2^{384}) + p' - 1 \ . \tag{12}$$

If we now choose a random value x, larger than $(p-1)$, we can calculate $x \bmod (p-1)$ and $x \bmod (p'-1)$ and observe if there are any similarities in the LSBs. Now let

$$a = x \bmod (p-1) = x - k(p-1) \tag{13}$$

$$b = x \bmod (p'-1) = x - k'(p'-1) \ , \tag{14}$$

for some k, k'. Thus,

$$a - b = (k' - k)(p' - 1) - kt(2^{384}) \ . \tag{15}$$

Clearly, $kt(2^{384})$ only affects the MSBs. The LSBs are controlled by $(k'-k)(p'-1)$. As we know p', we also know k', but k is unknown and unguessable provided x is significantly larger than $(p-1)$. Thus, the LSBs of a and b differ by the product of $(p'-1)$ and a pseudo-random integer $(k'-k)$. Provided that this is large enough, it cannot be guessed or found by some brute force approach.

These calculations are done knowing the entire value of x - in CRT RSA, an attacker would have only a portion of d, making these calculations even more uncertain. Thus, it would appear that the LSBs of d_p cannot be directly calculated in this fashion.

V Attacks on Partial Exposure

The most basic attack on partial hiding in RSA consists simply of a brute force approach, where an attacker tries all possible combinations for the hidden portion of the private key, d. We are proposing to hide bits 769-1024 of a 2048-bit d. This means that only 256 bits must be uncovered to break the system. However, an attacker would need to calculate the values of all 256 of these bits before being in a position to judge if any were correct. We believe that the parameters we have chosen for this system are secure under current computing power.

Our scheme is not vulnerable to any of the attacks mentioned in [2], as it is based on the Steinfeld-Zheng system, which is resistant to these attacks. However, Blömer and May [1] produced a paper recently which contains several attacks which may be applicible to such a scheme, depending on the parameters used. We will now look at each of the relevant attacks in detail and discuss whether they might pose a threat to the system we have proposed.

a Blömer-May Attack 1

We begin with their "strongest" attack - one which works for all $e < N^{\frac{7}{8}}$. In order for this attack to work, a certain portion of the LSBs of d must be exposed. Blömer and May work with a modulus N of bit-size $n = 1000$ and use varying bit-sizes of 300, 400 and 500 bits for e. This means that a minimum of 725, 782 and 834 LSBs of d respectively are needed in order for the attack to work. These values are calculated from the following theorem, proven in their paper:

Theorem V.1 (Blömer-May) *For every $\epsilon > 0$, there exists N_0 such that for every $N \geq N_0$, the following holds:*
Let (N, e) be an RSA public key with $\alpha = \log_N(e) \leq \frac{7}{8}$. Let d be the private key. Given d_0, M satisfying $d = d_0 \bmod M$ with

$$M \geq N^{\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}+\epsilon}, \tag{16}$$

then N can be factored in polynomial time.

If we apply this theorem to our proposed system, we can determine whether there is a threat posed to our scheme by this attack. Our modulus N is of size $n = 2048$ bits. We have set $e = 2^{16} + 1 = 65537$. We expose the 768 LSBs of d - we must now calculate what is the minimum number of LSBs required in order for the attack to proceed.

If N is of length 2048 bits, the smallest such number is 2^{2047}. From the theorem, if M is sufficiently large (where M denotes the number of LSBs known), then our system can be broken. In order to obtain the minimum number of LSBs needed, set $\epsilon \approx 0$ and neglect this term completely. We need to calculate the value of $\alpha = \log_N(e)$.

$$\begin{aligned} \alpha &= \log_{2^{2047}}(65537) \\ &= 0.0078163273 \ . \end{aligned} \tag{17}$$

We can now determine M using (16):

$$\begin{aligned} M &= \left(2^{2047}\right)^{\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}} \\ &= 2^{1039.3167} \ . \end{aligned} \tag{18}$$

Thus, a minimum of 1040 LSBs would be needed for this attack to work, and as we only expose 768 LSBs, our system would seem to be secure against this approach.

b Blömer-May Attack 2

The next attack described by Blömer and May to which our scheme may be vulnerable involves a provable attack for almost all $e < N^{\frac{1}{2}}$. Here, a vulnerability is introduced if

$$N^{\alpha+\frac{1}{2}+\epsilon} \leq M \leq 2N^{\alpha+\frac{1}{2}+\epsilon} \ , \tag{19}$$

where $0 < \alpha, \epsilon < 1/2$ and M represents the known LSBs again. If this is not the case, then the attack will not work. Calculating as before for our system, taking $\alpha, \epsilon \approx 0$ for the smallest number of bits acceptable, we get

$$N^{\frac{1}{2}} \leq M \leq 2N^{\frac{1}{2}} \ . \tag{20}$$

This in turn gives us

$$2^{1023.5} \leq M \leq 2^{1024.5} \tag{21}$$

so $M = 2^{1024}$. Thus, the 1025 LSBs of d would need to be exposed to make our system vulnerable to this attack, whereas we only expose the 768 LSBs of d.

c Blömer-May Attack 3

The final Blömer and May attack which may introduce a vulnerability to our scheme is based on the Chinese Remainder Theorem (CRT) version of RSA [21]. The attack enables an adversary to recreate d given the least significant half of the bits of d_p and works for low-exponent RSA such as $e = 2^{16} + 1$. In our system, the least significant half of the bits of d_p is only of length 512 bits. We expose the 768 LSBs of d, but in order to determine the value of d_p, it is necessary to know p. Although an attacker knows that the 384 LSBs of p and q are equal, and has access to the value of $N = pq$, these values are too large for him to successfully decipher p from this knowledge. The correctness of each bit that would be guessed would depend on the correctness of the bit before it. Clearly, the LSB will always be revealed - but we believe that the value of any further bits is still protected. Thus, our system is secure against this form of attack.

d Wiener's Continued Fractions Attack

Wiener's original attack [28] relies on continued fractions to find the value of d. It is successful for short secret exponents, which are up to one-quarter the size of the modulus, N. Our secret exponent is of length 2048 bits, but only 256 of these are securely hidden. We must investigate whether a continued fractions approach could expose a vulnerability in this scheme.

Let our secret exponent, d, consist of three sections: let d_0 represent the known 1024 MSBs, d_1 the unknown "middle" 256 bits and d_2 the known 768 LSBs. Then we have

$$d = d_0 2^{1024} + d_1 2^{768} + d_2 \quad . \tag{22}$$

This means we can adapt Wiener's notation and write:

$$e \left(d_0 2^{1024} + d_1 2^{768} + d_2 \right) = K \ \mathrm{lcm}(p-1, q-1) + 1 \ , \tag{23}$$

for some K. This gives us

$$e \left(d_0 2^{1024} + d_1 2^{768} + d_2 \right) = \frac{k}{g} \left(p-1 \right) \left(q-1 \right) + 1 \ , \tag{24}$$

where $G = \gcd(p-1, q-1)$, $k = K/\gcd(K, G)$ and $g = G/\gcd(K, G)$. At this point, Wiener manipulates the formula to read:

$$\frac{e}{pq} = \frac{k}{dg}(1-\delta) \text{ where } \delta = \frac{p+q-1-\frac{g}{k}}{pq} \quad . \tag{25}$$

If d is small, it can now be found using continued fractions. However, in our case, an attacker would need to be able to separate d_1 in (24) from the rest of the expression in a similar form in order to be able to launch an attack in this fashion. If d could be expressed as a product in place of a sum, in which the unknown d_1 was less than a quarter of the bits of N, then possibly an adaptation of Wiener's attack could be used, but we do not believe this can be done.

VI Conclusion and Future Work

In this paper, we propose a version of RSA in which part of the private key is publicly available, without allowing an attacker to uncover the remainder of the private key. We suggest using an individual's iris reading to securely encrypt the "secret" portion of the private key, in order to increase security. Two different implementations are described: one involving general RSA; the other using the Quisquater-Couvreur version. We also explore several possible attacks on these systems and show that they are resistant to all of these attacks.

Partially hiding the private key allows the decryption process to be completed more quickly in RSA, which may be of some advantage in implementations such as smart cards. It also allows us to use a biometric such as an iris reading to encrypt the hidden portion, as there would be insufficient information in an iris reading to directly XOR it with an entire private key. Clearly, this system could also be implemented using a different biometric, or combination of biometrics, an issue that we may explore further in the future. We also propose to further investigate the use of erasures in error-correcting codes to increase the error-correcting capacity of schemes like that of Davida, Frankel and Matt [7].

Another possible area of future study is the Rabin encryption scheme [22], which also relies on the intractability of factorising large numbers for security. It may be advantageous to use partial hiding of the private key in relation to this cryptosystem. We could also explore the possibility of partially hiding the factors in this case.

References

[1] J. Blömer and A. May. New partial key exposure attacks on RSA. In *Advances in Cryptology - Proc. of Crypto '03*, vol. 2729 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[2] D. Boneh, G. Durfee and Y. Frankel. Exposing an RSA private key given a small fraction of its bits, 1998. Full version of work presented at Asiacrypt '98, available at *http://crypto.stanford.edu/∼dabo/abstracts/bits_of_d.html*

[3] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *Advances in Cryptology - Proc. of Eurocrypt '00*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 453-469. Springer-Verlag, 2000.

[4] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233-260, 1997.

[5] J. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(11):1148-1161, November 1993.

[6] J. Daugman. Biometric personal identification system based on iris analysis. U.S. Patent Number 5291560, March 1994. Patent application made in July 1991.

[7] G.I. Davida, Y. Frankel and B.J. Matt. On the relation of error correction and cryptography to an offline biometric based identification scheme. In *Proc. of WCC99*, Workshop Coding and Cryptography, 1999.

[8] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22(6):644-654, November 1976.

[9] J.H. Ellis. *The Possibility of Secure Non-Secret Digital Encryption.* Communications-Electronics Security Group, GCHQ, Cheltenham, Gloucestershire, U.K., January 1970.

[10] M.J. Hinek, M.K. Low and E. Teske. On some attacks on multi-prime RSA. In *Selected Areas in Cryptography: 9th Annual Int'l Workshop, 2002*, vol. 2595 of *Lecture Notes in Computer Science*, pp. 385-404. Springer-Verlag, 2003.

[11] A. Jain, L. Hong and S. Pankanti. Biometric identification. *Comm. of the ACM*, 43(2):91-98, February 2000.

[12] B.S. Kaliski, Jr. TWIRL and RSA key size. Technical note, RSA Laboratories, May 2003. Available on RSA Homepage *http://www.rsasecurity.com/rsalabs/technotes/twirl.html*

[13] B.S. Kaliski, Jr and Y.L. Yin. The secure use of RSA. *Cryptobytes*, 1(3):7-13, Autumn 1995.

[14] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255-293, 2001.

[15] H.W. Lenstra, Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126:649-673, 1987.

[16] S. Liu and M. Wattenberg. A practical guide to biometric security technology. *IEEE IT Professional*, 3(1), January 2001.

[17] T. Mansfield, G. Kelly, D. Chandler and J. Kane. *Biometric Product Testing Final Report.* National Physical Laboratory, UK, March 2001. CESG contract X92A/4009309. *http://www.cesg.gov.uk/technology/biometrics*

[18] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[19] J.M. Pollard. Factoring with cubic integers. In A.K. Lenstra and H. W. Lenstra Jr, eds, *The Development of the Number Field Sieve*, vol. 1554 of *Lecture Notes in Math.*, pp. 4-10. Springer-Verlag, 1993.

[20] C. Pomerance. The quadratic sieve factoring algorithm. In *Advances in Cryptology - Proc. of Eurocrypto '84*, vol. 209 of *Lecture Notes in Computer Science*, pp. 169-182. Springer-Verlag, 1985.

[21] J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18(21):905-907, 1982.

[22] M.O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. *MIT/LCS/TR-212*, 1979. MIT Laboratory for Computer Science.

[23] R.L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120-126, February 1978.

[24] R.L. Rivest and R. D. Silverman. Are "strong" primes needed for RSA? Technical report, RSA Laboratories, December 1998.

[25] R.D. Silverman. A cost-based security analysis of symmetric and asymmetric key lengths. RSA Laboratories' Bull. 13, November 2001. Revised edition. Original publication April 2000.

[26] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy and B.V.K. Vijaya Kumar. Biometric Encryption. In R.K. Nichols, ed, *ICSA Guide to Cryptography*, chapter 22. Mc-Graw-Hill, 1999.

[27] R. Steinfeld and Y. Zheng. An advantage of low-exponent RSA with modulus primes sharing least significant bits. In *Proc. of RSA Conf. 2001, Cryptographer's Track*, vol. 202 of *Lecture Notes in Computer Science*, pp. 52-62. Springer-Verlag, 2001.

[28] M.J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Information Theory*, 36(3):553-558, May 1990.