RSA Laboratories

CryptoBytes

Volume 5, No. 1 Winter / Spring 2002

CONTENTS

I. *Fast Variants of RSA*

Dan Boneh
dabo@cs.stanford.edu

Hovav Shacham
hovav@cs.stanford.edu

ABSTRACT

We survey four variants of RSA designed to speed up RSA [12] decryption and signing. We only consider variants that are backwards compatible in the sense that a system using one of these variants can interoperate with systems using standard RSA.

1. INTRODUCTION

RSA is the most widely deployed public key cryptosystem. It is used for securing web traffic, e-mail, and some wireless devices. Since RSA is based on arithmetic modulo large numbers it can be slow in constrained environments. For example, 1024-bit RSA decryption on a small handheld device such as the PalmPilot III can take as long as 30 seconds. Similarly, on a heavily loaded web server, RSA decryption significantly reduces the number of SSL requests per second that the server can handle. Typically, one improves RSA's performance using special-purpose hardware. Current RSA coprocessors can perform as many as 10,000 RSA decryptions per second (using a 1024-bit modulus) and even faster processors are coming out.

In this paper we survey four simple variants of RSA that are designed to speed up RSA decryption in software. We emphasize backwards compatibility: A system using one of these variants for fast RSA decryption should be able to interoperate with systems that are built for standard RSA. Moreover, existing Certificate Authorities must be able to respond to a certificate request for a variant-RSA public key.

Editor's Note

After a hiatus of over two years, RSA Laboratories is pleased to resume publication of its newsletter. *CryptoBytes* aims to present timely research articles and surveys written by leading specialists on selected topics in cryptography and data security. Ranging from the highly accessible to the broadly technical, the articles in *CryptoBytes* present information in an easily digestible form to a wide audience consisting of specialists and non-specialists from industry and academia. To software engineers, students, IT professionals, scientists, and the many other members of our readership, welcome back!

This year marks the 25th anniversary of the invention of the RSA algorithm at M.I.T. by Profs. Rivest, Shamir, and Adleman. We are devoting this, the Winter/Spring 2002 issue of *CryptoBytes*, to a commemorative exploration of current research on RSA.

The most widespread current use of the RSA algorithm is in the Secure Sockets Layer (SSL) protocol for data protection on the Internet. In the first article in this issue, Dan Boneh and Hovav Shacham discuss several backwards compatible variants of RSA that speed up RSA decryption and signing. From David Pointcheval, we have a survey article guiding the reader through the labyrinthine evolution of RSA-OAEP and related schemes, all of which seek to provide rigorously provable security guarantees to RSA encryption. Finally, Pascal Paillier writes about a new encryption scheme of his devising that makes use of an RSA modulus, i.e., the product of two primes, as its basis, but possesses very different properties. The Paillier encryption scheme is especially attractive, for instance, for the design of electronic voting schemes.

As always, the input of our readers is important to the future success of *CryptoBytes*. We welcome comments, opinions, and proposals for future articles. The *CryptoBytes* editor may be contacted at cryptobytes editor@rsasecurity.com.

Fast Variants of RSA *cont'd from page 1*

We begin the paper with a brief review of RSA. We then describe the following variants for speeding up RSA decryption:

- Batch RSA [8]: do a number of RSA decryptions for approximately the cost of one
- Multi-factor RSA [7, 14]: use a modulus of the form $N = pqr$ or $N = p^2q$
- Rebalanced RSA [16]: speed up RSA decryption by shifting most of the work to the encrypter.

The security of these variants is an open research problem. We cannot show that an attack on these variants would imply an attack on the standardized version of RSA (as described, e.g., in ANSI X9.31). Therefore, when using these variants, one can only rely on the fact that so far none of them has been shown to be weak.

The RSA trapdoor permutation is used for both public key encryption and digital signatures. Since the exact application of RSA is orthogonal to the discussion in this paper we use terminology consistent with the application to public key encryption. All the RSA variants we discuss apply equally well to digital signatures, where they speed up RSA signing.

1.1 Review of the basic RSA system

We review the basic RSA public key system and refer to [10] for more information. We describe three constituent algorithms: key generation, encryption, and decryption.

Key generation: The key generation algorithm takes a security parameter n as input. Throughout the paper we use $n = 1024$ as the standard security parameter. The algorithm generates two $(n/2)$-bit primes, p and q, and sets $N \leftarrow pq$. Next, it picks some small value e that is relatively prime to $\varphi(N) = (p-1)(q-1)$. The value e is called the encryption exponent, and is usually chosen as $e = 65537$. The RSA public key consists of the two integers $\langle N, e\rangle$. The RSA private key is an integer d satisfying $e \cdot d = 1 \bmod \varphi(N)$. Typically, one sends the public key $\langle N, e\rangle$ to a certificate authority (CA) to obtain a certificate for it.

Encryption: To encrypt a message X using an RSA public key $\langle N, e\rangle$, one first formats the bit-string X to obtain an integer M in $Z_N = \{0, \ldots, N-1\}$. This formatting is often done using the PKCS #1 standard [1, 9]. The ciphertext is then computed as $C \leftarrow M^e \bmod N$. (Other methods for formatting X prior to encryption are described elsewhere in this issue.)

Decryption: To decrypt a ciphertext C the decrypter uses its private key d to compute an e'th root of C by computing $M \leftarrow C^d \bmod N$. Since both d and N are large numbers (each approximately n bits long) this is a lengthy computation for the decrypter. The formatting operation from the encryption algorithm is then reversed to obtain the original bit-string X from M. Note that d must be a large number (on the order of N) since otherwise the RSA system is insecure [3, 16].

It is standard practice to employ the Chinese Remainder Theorem (CRT) for RSA decryption. Rather than compute $M \leftarrow C^d \pmod N$, one evaluates:

$$M_p \leftarrow C_p^{d_p} \pmod p \qquad M_q \leftarrow C_p^{d_q} \pmod q.$$

Here $d_p = d \bmod p-1$ and $d_q = d \bmod q-1$. Then one uses the CRT to calculate M from M_p and M_q. This is approximately four times as fast as evaluating $C^d \bmod N$ directly [10, p. 613].

2. BATCH RSA

Fiat [8] showed that, when using small public exponents e_1 and e_2 for the same modulus N, it is possible to decrypt two ciphertexts for approximately the price of one. Suppose C_1 is a ciphertext obtained by encrypting some M_1 using the public key $\langle N, 3\rangle$, and C_2 is a ciphertext for some M_2 using $\langle N, 5\rangle$. To decrypt, we must compute $C_1^{1/3}$ and $C_2^{1/5} \bmod N$. Fiat observed that by setting $A = (C_1^5 \cdot C_2^3)^{1/15}$ we obtain:

(1)
$$C_1^{1/3} = \frac{A^{10}}{C_1^3 \cdot C_2^2} \quad \text{and } C_2^{1/5} = \frac{A^6}{C_1^2 \cdot C_2}.$$

Hence, at the cost of computing a single 15th root and some additional arithmetic, we are able to decrypt both C_1 and C_2. Computing a 15th root takes the same time as a single RSA decryption.

This batching technique is only worthwhile when the public exponents e_1 and e_2 are small (e.g., 3 and 5). Otherwise, the extra arithmetic required is too expensive. Also, one can only batch-decrypt ciphertexts encrypted using the same modulus and *distinct public exponents.* This is essential — it is known [13, Appendix A] that one cannot apply such algebraic techniques to batch the decryption of two ciphertexts encrypted with the same public key (e.g., we cannot batch compute $C_1^{1/3}$ and $C_2^{1/3}$).

Fiat generalized the above observation to the decryption of a batch of b RSA ciphertexts. We have b pairwise relatively prime public keys $e_1, \ldots, e_b$, all sharing a common modulus N. Furthermore, we have b encrypted messages $C_1, \ldots, C_b$, where C_i is encrypted using the exponent e_i. We wish to compute $M_i = C_i^{1/e_i}$ for $i = 1, \ldots, b$. Fiat describes this b-batch processing a binary tree. For small values of b ($b \le 8$), one can use a direct generalization of (1). One sets $e \leftarrow \prod_i e_i$, and $A_0 \leftarrow \prod_i C_i^{e/e_i}$ (where the indices range over $1, \ldots, b$). Then one calculates $A \leftarrow A_0^{1/e} = \prod_{i=1}^{b} C_i^{1/e_i}$. For each i one computes M_i as:

(2)
$$M_i = C_i^{1/e_i} = \frac{A^{\alpha_i}}{C_i^{(\alpha_i - 1)/e_i} \cdot \prod_{j \ne i} C_j^{\alpha_i/e_j}}$$

$$\text{where} \quad \alpha_i = \begin{cases} 1 \bmod e_i \\ 70 \bmod e_j \ (\text{for } j \ne i) \end{cases}$$

This b-batch requires b modular inversions whereas Fiat 's tree-based method requires $2b$ modular inversions, but fewer auxiliary multiplications. Note that since b and the e_i's are small the exponents in (2) are also small.

2.1 Improving the performance of batch RSA

In [13] the authors show how to use batch RSA within the Apache Web server to improve the performance of the SSL handshake. This requires changing the web server architecture. They also describe several natural improvements to batch RSA. We mention a few of these improvements here.

Batch division: Modular inversion is much slower than modular multiplication. Using a trick due to Montgomery we compute all b inversions in the batch algorithm for the cost of a single inversion and a few more multiplications. The idea is to invert x and y by computing $\alpha \leftarrow (xy)^{-1}$ and setting $x^{-1} \leftarrow y \cdot \alpha$ and $y^{-1} \leftarrow x \cdot \alpha$. Thus we obtain the inverses of both x and y at the cost of a single modular inversion and three multiplications.

More generally, we use the following fact [6, p.481]: Let $x_1, \ldots, x_n$ *be elements of* Z_N. *All* n *inverses* $x_1^{-1}, \ldots, x_n^{-1}$ *can be obtained at the cost of one inversion and* $3n$ *multiplications.*

Consequently, only a single modular inversion is required for the entire batching procedure.

Global Chinese Remainder: In Section 1.1 we mentioned that RSA decryption uses the CRT to speed up the computation of $C^d \bmod N$. This idea extends naturally to batch decryption. We run the entire batching algorithm modulo p, and again modulo q, then use the CRT on each of the b pairs $\langle C_i^{1/e_i} \bmod p, C_i^{1/e_i} \bmod q \rangle$ to obtain the b decryptions $M_i = C_i^{1/e_i} \bmod N$.

Simultaneous Multiple Exponentiation: Simultaneous multiple exponentiation [10, §14.6] is a method for calculating $a^u \cdot b^v \bmod m$ without first evaluating a^u and b^v. It requires approximately as many multiplications as does a single exponentiation with the larger of u or v as exponent. Such products of exponents are a large part of the batching algorithm. Simultaneous multiple exponentiation cuts the time required to perform them by close to 30%.

2.2 Performance of batch RSA

Table 1 lists the running time for stand-alone batch-RSA decryption, using OpenSSL 0.9.5 on a machine with a 750 MHZ Pentium III and 256 MB RAM, running Debian Linux. In all experiments, the smallest possible values for the encryption exponents e_i were used.

batch	key size		
size	768	1024	2048
(unbatched)	4.67	8.38	52.96
2	3.09	5.27	29.43
4	1.93	3.18	16.41
8	1.55	2.42	10.81

Table 1: RSA decryption time, in milliseconds, as a function of batch and key size

With standard 1024-bit keys, batching improves performance significantly. With $b = 4$, RSA decryption is accelerated by a factor of 2.6; with $b = 8$, by a factor of almost 3.5. Note that a batch size of more than eight is probably not useful for common applications, since waiting for many decryption requests to be queued can significantly increase latency.

batch	Server load		
size	16	32	48
(unbatched)	105	98	98
2	149	141	134
4	218	201	187
8	274	248	227

Table 2: SSL handshakes per second as a function of batch size. 1024 bit keys.

We also mention batch-RSA performance as a component of a larger system — a web server handling SSL traffic. An architecture for such a system is described in [13]. Table 2 gives the number of full SSL handshakes per second that the batch-RSA web server can handle, when bombarded with concurrent HTTP HEAD requests by a test client. Here "server load" is the number of simultaneous connection threads the client makes to the server. Under heavy load, batch RSA can improve the number of full SSL handshakes per second by a factor of approximately 2.5.

2.3 The downside of batch RSA

Batch RSA can lead to a significant improvement in RSA decryption time. Nevertheless, there are a few difficulties with using the batching technique:

- When using batch RSA, the decryption server must maintain at least as many RSA certificates as there are distinct keys in a batch. Unfortunately, current Certificate Authorities charge per certificate issued regardless of the public key in the certificate. Hence, the cost of certificates might outweigh the benefits in performance.
- For optimal performance, batching requires RSA public keys with very small public exponents ($e = 3,5,7,11,\ldots$). There are no known attacks on the resulting system, but RSA as usually deployed uses a larger public exponent ($e = 65537$).

3. MULTI-FACTOR RSA

The second RSA variant is based on modifying the structure of the RSA modulus. Here there are two proposals. The first [7] uses a modulus of the form $N = pqr$. When N is 1024 bits, each prime is approximately 341 bits. We refer to this as multi-prime RSA. The second proposal [14] uses an RSA modulus of the form $N = p^2q$ and leads to an even greater speedup. Both methods are fully backwards compatible since there is no known efficient algorithm to distinguish such RSA public keys from standard RSA keys (where $N = pq$).

3.1 Multi-prime RSA: $N = pqr$

We begin with multi-prime RSA [7]. We describe key generation, encryption, and decryption. We then discuss the performance of the scheme and analyze its security.

Key generation: The key generation algorithm takes as input a security parameter n and an additional parameter b. It generates an RSA public/private key pair as follows:

Step 1: Generate b distinct primes $p_1, \ldots, p_b$ each $\lfloor n/b \rfloor$-bits long. Set $N \leftarrow \prod_{i=1}^{b} p_i$. For a 1024-bit modulus we can use at most $b = 3$ (i.e., $N = pqr$), for security reasons discussed below.

Step 2: Pick the same e used in standard RSA public keys, namely e = 65537. Then compute $d = e^{-1} \bmod \varphi(N)$. As usual, we must ensure that e is relatively prime to $\varphi(N) = \prod_{i=1}^{b}(p_i\text{-}1)$. The public key is $\langle N,e \rangle$; the private key is d.

Encryption: Given a public key $\langle N,e \rangle$, the encrypter encrypts exactly as in standard RSA.

Decryption: Decryption is done using the Chinese Remainder Theorem (CRT). Let $r_i = d \bmod p_i\text{-}1$. To decrypt a ciphertext C, one first computes $M_i = C^{r_i} \bmod p_i$ for each i, $1 \le i \le b$. One then combines the M_i's using the CRT to obtain $M = C^d \bmod N$. The CRT step takes negligible time compared to the b exponentiations.

Performance. We compare the decryption work using the above scheme to the work done when decrypting a normal RSA ciphertext. Recall that standard RSA decryption using CRT requires two full exponentiations modulo $n/2$-bit numbers. In multi-prime RSA decryption requires b full exponentiations modulo n/b bit numbers. Using basic algorithms computing $x^d \bmod p$ takes time $O(\log d \log^2 p)$. When d is on the order of p the running time is $O(\log^3 p)$. Therefore, the asymptotic speedup of multi-prime RSA over standard RSA is simply:

$$\frac{2 \cdot (n/2)^3}{b \cdot (n/b)^3} = b^2/4.$$

For 1024-bit RSA, we can use at most $b = 3$ (i.e., $N = pqr$), which gives a theoretical speedup of approximately 2.25 over standard RSA decryption. Our experiments (implemented using the GMP bignum library) show that in practice we get a speedup of a factor of 1.73 over standard RSA.

Security. The security of multi-factor RSA depends on the difficulty of factoring integers of the form $N = p_1 \ldots p_b$ for $b > 2$. The fastest known factoring algorithm (the number field sieve) cannot take advantage of this special structure of N. However, one has to make sure that the prime factors of N do not fall within the range of the Elliptic Curve Method (ECM), which is analyzed in [15]. Currently, 256-bit prime factors are considered within the bounds of ECM, since the work to find such factors is within range of the work needed for the RSA-512 factoring project [5]. Consequently, for 1024-bit moduli one should not use more than three factors.

3.2 Multi-power RSA: $N = p^2 q$

One can further speed up RSA decryption using a modulus of the form $N = p^{b-1}q$ where p and q are n/b bits each [14]. When N is 1024-bits long we can use at most $b = 3$, i.e., $N = p^2 q$. The two primes p,q are then each 341 bits long.

Key generation: The key generation algorithm takes as input a security parameter n and an additional parameter b. It generates an RSA public/private key pair as follows:

Step 1: Generate two distinct $\lfloor n/b \rfloor$-bit primes, p and q, and compute $N \leftarrow p^{b-1} \cdot q$.

Step 2: Use the same public exponent e used in standard RSA public keys, namely $e = 65537$. Compute $d \leftarrow e^{-1}$ mod $(p-1)(q-1)$.

Step 3: Compute $r_1 \leftarrow d$ mod $p - 1$ and $r_2 \leftarrow d$ mod q-1. The public key is $\langle N,e \rangle$; the private key is $\langle p,q,r_1,r_2 \rangle$.

Encryption: Same as in standard RSA.

Decryption: To decrypt a ciphertext c using the private key $\langle p,q,r_1,r_2 \rangle$ one does:

Step 1: Compute $M_1 \leftarrow C^{r_1}$ mod p and $M_2 \leftarrow C^{r_2}$ mod q. Thus $M_1^e = C$ mod p and $M_2^e = C$ mod q.

Step 2: Using Hensel lifting [6, p. 137] construct an M_1' such that $(M_1')^e = C$ mod p^{b-1}. Hensel lifting is much faster than a full exponentiation modulo p^{b-1}.

Step 3: Using CRT, compute an $M \in Z_N$ such that $M = M_1'$ mod p^{b-1} and $M = M_2$ mod q. Then $M = C^d$ mod N is a proper decryption of C.

Performance. We compare the work required to decrypt using multi-power RSA to that required for standard RSA. For multi-power RSA, decryption takes two full exponentiations modulo (n/b)-bit numbers, and b-2 Hensel liftings. Since the Hensel-lifting is much faster than exponentiation, we focus on the time for the two exponentiations. As noted before, a full exponentiation using basic modular arithmetic algorithms takes cubic time in the size of the modulus. So, the speedup of multi-power RSA over standard RSA is approximately:

$$\frac{2 \cdot (n/2)^3}{2 \cdot (n/b)^3} = b^3/8$$

For 1024-bit RSA, b should again be at most three (i.e., $N = p^2 q$), giving a theoretical speedup of about 3.38 over standard RSA decryption. Our experiments (implemented using GMP and taking $e = 65537$) show that in practice we get a speedup by a factor of 2.30 over standard RSA.

Security. The security of multi-power RSA depends on the difficulty of factoring integers of the form $N = p^{b-1}q$. As for multi-prime RSA, one has to make sure that the prime factors of N do not fall within the capabilities of ECM (and the ECM improvement for $N = p^2q$ [11]). Consequently, for 1024-bit moduli one can use at most $b = 3$, i.e., $N = p^2q$. In addition, we note that the Lattice Factoring Method (LFM) [4], designed to factor integers of the form $N = p^u \cdot q$ for large u, cannot efficiently factor integers of the form $N = p^2q$ when N is 1024 bits long.

4. REBALANCED RSA

In standard RSA, encryption and signature verification are much faster than decryption and signature generation. In some applications, one would like to have the reverse behavior. For example, when a cell phone needs to generate an RSA signature that will be later verified on a fast server one would like signing to be easier than verifying. Similarly, SSL web browsers (doing RSA encryption) typically have idle cycles to burn whereas SSL web servers (doing RSA decryption) are overloaded. In this section we describe a variant of RSA that enables us to rebalance the difficulty of encryption and decryption: we speed up RSA decryption by shifting the work to the encrypter. This variant is based on a proposal by Wiener [16] (see also [2]). Note that we cannot simply speedup RSA decryption by using a small value of d since as soon as d is less than $N^{0.292}$ RSA is insecure [16,3]. The trick is to choose d such that d is large (on the order of N), but $d \bmod p-1$ and $d \bmod q-1$ are small numbers. As before, we describe key generation, encryption, and decryption.

Key generation: The key generation algorithm takes two security parameters n and k where $k \leq n/2$. Typically $n = 1024$ and $k = 160$. It generates an RSA key as follows:

Step 1: Generate two distinct $(n/2)$-bit primes p and q with $\gcd(p-1, q-1) = 2$. Compute $N \leftarrow pq$.

Step 2: Pick two random k-bit values r_1 and r_2 such that

$\gcd(r_1, p-1) = 1$, and $\gcd(r_2, q-1) = 1$ and $r_1 = r_2 \bmod 2$.

Step 3: Find a d such that $d = r_1 \bmod p-1$ and $d = r_2 \bmod q-1$.

Step 4: Compute $e \leftarrow d^{-1} \bmod \varphi(N)$. The public key is $\langle N, e\rangle$; the private key is $\langle p, q, r_1, r_2\rangle$.

We need to explain how to find d in Step 3. One usually uses the Chinese Remainder Theorem (CRT). Unfortunately, $p-1$ and $q-1$ are not relatively prime (they are both even) and consequently the theorem does not apply. However, $(p-1)/2$ is relatively prime to $(q-1)/2$. Furthermore, $r_1 = r_2 \bmod 2$. Let $a = r_1 \bmod 2$. Then using CRT we can find an element d' such that

Now, observe that the required d in Step 3 is simply $d = 2d' + a$. Indeed, $d = r_1 \bmod p-1$ and $d = r_2 \bmod q-1$.

In Step 4, we must justify why d is invertible modulo $\varphi(N)$. Recall that $\gcd(r_1, p-1) = 1$ and $\gcd(r_2, q-1) = 1$. It follows that $\gcd(d, p-1) = 1$ and $\gcd(d, q-1) = 1$.

$$d' = \frac{r_1 - a}{2} \pmod{\frac{p-1}{2}} \text{ and } d' = \frac{r_2 - a}{2} \pmod{\frac{q-1}{2}}.$$

Consequently, $\gcd(d, (p-1)(q-1)) = 1$. Hence, d is invertible modulo $\varphi(N) = (p-1)(q-1)$.

For security reasons described below we take $k = 160$, although other larger values are acceptable. Note that e is very large — on the order of N. This is unlike standard RSA, where e typically equals 65537. All Certificate Authorities we tested were willing to generate certificates for such RSA public keys.

Encryption: Encryption using the public key $\langle N, e\rangle$ is identical to encryption in standard RSA. The only issue is that since e is much larger than in standard RSA, the encrypter must be willing to accept such public keys. At the time of this writing all browsers we tested were willing to accept such keys. The only exception is Microsoft's Internet Explorer (IE) — IE allows a maximum of 32 bits for e.

Decryption: To decrypt a ciphertext C using the private key $\langle p,q,r_1,r_2 \rangle$ one does:

Step 1: Compute $M_1 \leftarrow C^{r_1} \bmod p$ and $M_2 \leftarrow C^{r_2} \bmod q$.

Step 2: Using the CRT compute an $M \in Z_N$ such that $M = M_1 \bmod p$ and $M = M_2 \bmod q$. Note that $M = C^d \bmod N$. Hence, the resulting M is a proper decryption of C.

Performance. We compare the work required to decrypt using the above scheme to that required using standard RSA. Recall that decryption time for standard RSA with CRT is dominated by two full exponentiations modulo $(n/2)$-bit numbers. In the scheme presented above, the bulk of the decryption work is in the two exponentiations in Step 1, but in each of these the exponent is only k bits long. Since modular exponentiation takes time linear in the exponent's bit-length, we get a speedup of $(n/2)/k$ over standard RSA. For a 1024-bit modulus and 160-bit exponent ($k = 160$), this gives a theoretical speedup of about 3.20 over standard RSA decryption. Our experiments (implemented using GMP) show that in practice we get a speedup by a factor of 3.06 over standard RSA.

Security. It is an open research problem whether RSA using values of d as above is secure. Since d is large, the usual small-d attacks [16,3] do not apply. The best known attack on this scheme is described in the following lemma [2].

Lemma. Let $\langle N,e \rangle$ be an RSA public key with $N = pq$. Let $d \in Z$ be the corresponding RSA private exponent satisfying $d = r_1 \bmod p - 1$ and $d = r_2 \bmod q - 1$ with $r_1 < r_2$. Then given $\langle N,e \rangle$ an adversary can expose the private key d in time $O(\sqrt{r_1} \log r_1)$.

The above attack shows that, to obtain security of 2^{80}, we must make both r_1 and r_2 be at least 160 bits long. Consequently, for security reasons k should not be less than 160.

5. CONCLUSIONS

We surveyed four variants of RSA designed to speed up RSA decryption and be backwards-compatible with standard RSA. Table 3 gives the decryption speedup factors for each of these variants using a 1024-bit RSA modulus. Batch RSA is fully backwards-compatible, but requires the decrypter to obtain and manage multiple public keys and certificates. The two multi-factor RSA techniques are promising in that they are fully backwards-compatible. The rebalanced RSA method gives a large speedup, but only works with peer applications that properly implement standard RSA, and so are willing to accept RSA certificates with a large encryption-exponent e. Currently, Internet Explorer rejects all RSA certificates where e is more than 32 bits long. Multi-factor RSA and rebalanced RSA can be combined to give an additional speedup. All these variants can take advantage of advances in algorithms for modular arithmetic (e.g., modular multiplication and exponentiation) on which RSA is built.

Method		Speedup	Comment
Batch RSA,	$b = 4$	2.64	Requires multiple certificates
Multi-prime,	$N = pqr$	1.73	
Multi-power,	$N = p^2q$	2.30	e = 65537
Rebalance,	$k = 160$	3.06	Incompatible with Internet Explorer

Table 3: Comparison of RSA variants. Experimental speedup factors for 1024-bit keys.

ACKNOWLEDGEMENTS

The authors thank Ari Juels for his comments on preliminary versions of this paper.

REFERENCES

[1] M. Bellare and P. Rogaway. "Optimal Asymmetric Encryption." In A. De Santis, ed, *Proceedings of Eurocrypt '94* vol. 950 of *Lecture Notes in Computer Science* (*LNCS*), pp. 92–111. Springer-Verlag,1994.

[2] D. Boneh. "Twenty Years of Attacks on the RSA Cryptosystem." *Notices of the American Mathematical Society* 46(2): 203–213, Feb. 1999.

[3] D. Boneh and G. Durfee. "Cryptanalysis of RSA with Private Key d Less than $n^{0.292}$." *IEEE Transactions on Information Theory* 46(4):1339–1349, Jul. 2000. Early version in *Proceedings of Eurocrypt '99*

[4] D. Boneh, G. Durfee, and N. Howgrave-Graham. "Factoring $N = p^r q$ for Large r." *Proceedings of Crypto '99*, vol. 1666 of *LNCS,* pp. 326–337. Springer-Verlag, 1999.

[5] S. Cavallar, B. Dodson, A. K. Lensra, W. Lioen, P. Montgomery, B. Murphy, H. Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffet, C. Putnam, P. Zimmermann, "Factorization of a 512-Bit RSA Modulus," *Proceedings of Eurocrypt 2000* vol.1807 of *Lecture Notes in Computer Science* (*LNCS*), pp. 1–11, Springer-Verlag, 2000.

[6] H. Cohen. *A Course in Computational Algebraic Number Theory,* vol. 138 of *Graduate Texts in Mathematics* Springer-Verlag, 1996.

[7] T. Collins, D. Hopkins, S. Langford, and M. Sabin. *Public Key Cryptographic Apparatus and Method.* US Patent #5,848,159. Jan. 1997.

[8] A. Fiat. "Batch RSA." In G.Brassard, ed., *Proceedings of Crypto '89,* vol. 435 of (*LNCS*) pp. 175–185. Springer-Verlag, 1989.

[9] RSA Labs. *Public Key Cryptography Standards* (*PKCS*), *Number 1* Version 2.0. Version 2.1 draft is available at

http://www.rsalabs.com/pkcs/pkcs-1/index.html

[10] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[11] E. Okamoto, R. Peralta. "Faster Factoring of Integers of a Special Form," IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, E79-A, n.4 (1996).

[12] R. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Commun. of the ACM* 21(2): 120 –126. Feb. 1978.

[13] H. Shacham and D. Boneh. "Improving SSL Handshake Performance via Batching." In D. Naccache, ed., *Proceedings of RSA 2001* vol. 2020 of *LNCS,* pp. 28 – 43. Springer-Verlag, 2001.

[14] T. Takagi. "Fast RSA-type Cryptosystem Modulo $p^k q$." In H. Krawczyk, ed., *Proceedings of Crypto '98,* vol. 1462 of *LNCS,* pp. 318–326. Springer-Verlag, 1998.

[15] R. Silverman and S. Wagstaff, Jr. "A Practical Analysis of the Elliptic Curve Factoring Algorithm." *Math. Comp*. 61(203):445–462. Jul. 1993.

[16] M. Wiener. "Cryptanalysis of Short RSA Secret Exponents." *IEEE Trans. on Info. Th*. 36(3):553–558. May 1990.

II. How to Encrypt Properly with RSA

David Pointcheval

Dépt d'Informatique, ENS – CNRS, 45 rue d'Ulm,
75230 Paris Cedex 05, France

David.Pointcheval@ens.fr
http: //www.di.ens.fr /users /pointche

ABSTRACT

In 1993, Bellare and Rogaway formalized the concept of a random oracle, imported from complexity theory for cryptographic purposes. This new tool allowed them to present several asymmetric encryption and signature schemes that are both efficient and provably secure (in the random oracle model). The Optimal Asymmetric Encryption Padding (OAEP) is the most significant application of the random oracle model to date. It gives an efficient RSA encryption scheme with a strong security guarantee (semantic security against chosen-ciphertext attacks). After Bleichenbacher's devastating attack on RSA-PKCS #1 v1.5 in 1998, RSA-OAEP became the natural successor (RSA-PKCS #1 v2.0) and thus a *de facto* international standard. Surprisingly, Shoup recently showed that the original proof of security for OAEP is incorrect. Without a proof, RSA-OAEP cannot be trusted to provide an adequate level of security. Luckily, shortly after Shoup's discovery a formal and complete proof was found in joint work by the author and others that reaffirmed the strong level of security provided by RSA-OAEP. However, this new security proof still does not guarantee security for key sizes used in practice due to the inefficiency of the security reduction (the reduction to inverting RSA takes quadratic time). Recent alternatives to OAEP, such as OAEP+, SAEP+, and REACT, admit more efficient proofs and thus provide adequate security for key sizes used in practice.

1. ASYMMETRIC ENCRYPTION

In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation [*RiShAd78*]. A trapdoor permutation primitive is a function f that anyone can compute efficiently; however, inverting f is hard unless we are also given some "trapdoor" information. Given the trapdoor information, inverting f becomes easy. Naively, a trapdoor permutation defines a simple public key encryption scheme: the description of f is the public key and the trapdoor is the secret key. Unfortunately, encryption in this naive public key system is deterministic and hence cannot be secure, as discussed below.

Before we can claim that a cryptosystem is secure (or insecure) we must precisely define what security actually means. The formalization of security notions started around the time when RSA was proposed and took several years to converge (see [*Go97*] for a survey on this topic). Today, the accepted security requirement for an encryption scheme is called "semantic security against an adaptive chosen-ciphertext attack" [*RaSi91*] or IND-CCA for short. To understand this concept we point out that security is always defined in terms of two parameters: (1) the attacker's capabilities, namely what the attacker can do during the attack, and (2) the attacker's goals, namely what the attacker is trying to do.

1. Attacker's capabilities: The strongest attacker capability in the standard model is called "adaptive chosen-ciphertext attack" and is denoted by (CCA) [*RaSi91*]. This means that the adversary has the ability to decrypt any ciphertext of his choice except for some challenge ciphertext (imagine the attacker is able to exploit a decryption box that will decrypt anything except for some known challenge ciphertext).

2. Attacker's goal: The standard security goal is called "semantic security" [*GoMi84*] (also known as "indistinguishability of ciphertexts"), and is denoted by (IND). Roughly speaking, the attacker's goal is to deduce just one bit of information about the decryption of some given ciphertext. We say that a system is semantically secure if no efficient attacker can achieve this goal. We note that a deterministic encryption algorithm can never give semantic security.

An encryption scheme that is semantically secure under an adaptive chosen-ciphertext attack is said to be IND-CCA secure. IND-CCA security implies that even with full-access to the decryption oracle, the attacker is not able to deduce one bit of information about the decryption of a given challenge ciphertext. IND-CCA may seem very strong, but such attacks are possible in some real world scenarios. In fact, CCA-like attacks have been used to break practical implementations, as we will see later. Furthermore, semantic security is required for high confidentiality, namely when the message space is limited (such as "yes" or "no," "buy" or "sell"). As a consequence, IND-CCA is accepted as the required security level for practical encryption schemes

One can obtain many other security notions by combining different attacker goals with various attacker capabilities. For example, another security goal is called "non-malleability" [*DoDwNa00*, *BeSh99*]. Here the attacker is given some ciphertext and his goal is to build another ciphertext such that the plaintexts are meaningfully related. Non-malleability is known to be equivalent to semantic-security under an adaptive chosen-ciphertext attack [*BeDePoRo98*]. For this reason, IND-CCA security is sometimes called non-malleability. Similarly, one can also consider different attacker capabilities based on the oracles given to the attacker [*NaYu89*, *RaSi91*, *Bl98*, *GoHaSc99*, *OkPo01*]. As mentioned above, the most powerful attacker capability in the "classical" model is the decryption oracle itself, which decrypts any ciphertext (except the challenge ciphertext). This "classical" model gives the cryptographic engine to the adversary as a black box to which he can make queries and receive correct answers in constant time. It thus excludes timing attacks [*Ko96*], simple and differential power analyses [KoJaJu99] as well, and other differential fault analyses [BiSh97, *BoDeLi97*].

2. THE RSA-BASED CRYPTOSYSTEMS

2.1 The Plain-RSA

The RSA permutation, proposed by Rivest, Shamir and Adleman [*RiShAd78*], is the most well known trapdoor permutation. Its one-wayness is believed to be as strong as integer factorization. The RSA setup consists of choosing two large prime numbers p and q, and computing the RSA modulus $n = pq$. The public key is n together with an exponent e (relatively prime to $\varphi(n)=(p-1)(q-1)$). The secret key d is defined to be the inverse of e modulo $\varphi(n)$. Encryption and decryption is defined as follows:

$$E_{n,e}(m) = m^e \bmod n \qquad D_{n,d}(c) = c^d \bmod n.$$

This primitive does not provide by itself an IND-CCA secure encryption scheme. Under a slightly stronger assumption than the intractability of the integer factorization, it gives a cryptosystem that is only one-way under chosen-plaintext attacks — a very weak level of security. Semantic security fails because encryption is deterministic. Even worse, under a CCA attack, the attacker can fully decrypt a challenge ciphertext $C = m^e \bmod n$ using the homomorphic property of RSA:

$$E_{n,e}(m_1)\, E_{n,e}(m_2) = E_{n,e}(m_1\, m_2) \bmod n.$$

To decrypt $C = m^e \bmod n$ using a CCA attack do: (1) compute $C' = C \cdot 2^e \bmod n$, (2) give C' (≠ C) to the decryption oracle, and (3) the oracle returns $2m \bmod n$ from which the adversary can deduce m.

To overcome RSA this simple CCA attack, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption. This randomizes the ciphertext and eliminates the homomorphic property.

2.2 The RSA-PKCS #1 v1.5 Encryption

A widely deployed padding for RSA-based encryption is defined in the PKCS #1 v1.5 standard: for any modulus $2^{8(k-1)} \leq n < 2^{8k}$, in order to encrypt an ℓ byte-long message m (for $\ell \leq k$-11), one randomly chooses a k-3-ℓ byte-long random string r (with only non-zero bytes). Then, one defines the k-byte long string $M = 02 \,||\, r \,||\, 0 \,||\, m$ (see Figure 1) which is thereafter encrypted with the RSA permutation, $C = M^e \bmod n$. When decrypting a ciphertext C, the decryptor applies RSA inversion by computing $M = C^d \bmod n$ and then checks that the result M matches the expected format $02 \,||\, * \,||\, 0 \,||\, *$. If so, the decryptor outputs the last part as the plaintext. Otherwise, the ciphertext is rejected.

0	2	non-zero bytes more than 8 bytes	0	m

Figure 1: PKCS #1 v1.5 Format

Intuitively, this padding seems sufficient to rule out the above weaknesses of the plain RSA system, but without any formal proof or guarantee. Surprisingly, in 1998, Bleichenbacher [*Bl98*] showed that a simple active attack can completely break RSA-PKCS #1. This attack applies to real systems such as a Web server using SSL v3.0. These servers often output a specific "failure" message in case of an invalid ciphertext. This enables an attacker to test whether the two most significant bytes of a challenge ciphertext C are equal to `02'. If so, the attacker learns the following bound on the decryption of C:

$$2.20^{8(k-2)} \leq C^d \bmod n < 3.2^{8(k-2)}.$$

Due to the random self-reducibility of the RSA permutation, in particular the homomorphism

$$C\, s^e = M^e\, s^e = (Ms)^e \bmod n,$$

the complete decryption of C can be recovered after a relatively small number of queries. Only a few million queries are needed with a 1024-bit modulus.

Bleichenbacher's attack had an impact on many practical systems and standards bodies, which suddenly became aware of the importance of formal security arguments. Nevertheless, the weak PKCS #1 v1.5 padding is still used in the TLS protocol [*TLS*]. The TLS specification now appears to defend against Bleichenbacher's attack using a technique for which no proof of security has yet been published. Certain simple attacks are still possible (for example, plaintext-checking attacks [*OkPo01*] can be easily run, even if they seem ineffective). The lesson here is that standards should rely as much as possible on fully analyzed constructions and avoid ad-hoc techniques.

3. THE OPTIMAL ASYMMETRIC ENCRYPTION PADDING

For some time, people have tried to provide security proofs for cryptographic protocols in the "reductionist" sense [*BlM84*]. To do so, one presents an algorithm that uses an effective adversary as a sub-program to break some underlying hardness assumption (such as the RSA assumption, or the intractability of the integer factorization). Such an algorithm is called a "reduction." This reduction is said to be efficient, roughly speaking, if it does not require too many calls to the sub-program.

3.1 The Random Oracle Model

A few years ago, a new line of research started with the goal of combining provable security with efficiency, still in the "reductionist" sense. To achieve this goal, Bellare and Rogaway [*BeRo93*] formalized a heuristic suggested by Fiat and Shamir [*FiSh86*]. This heuristic consisted in making an idealized assumption about some objects, such as hash functions, according to which they were assumed to behave like truly random functions. This assumption, known as the "random oracle model," may seem strong, and lacking in practical embodiments. In fact, Canetti *et al.* [*CaGoHa98*] gave an example of a signature scheme which is secure in the random oracle model, but insecure under any instantiation of the random oracle.

However, one can also consider random-oracle-based proofs under the assumption that the adversary is generic, whatever the actual implementation of the hash function or other idealized algorithms may be. In other words, we may assume that the adversary does/can not use any specific weakness of the hash functions used in practice. Thanks to this ideal assumption, several efficient encryption and signature schemes have been analyzed [*BeRo94*, *BeRo96*, *PoSt00*].

We emphasize that even formal analyses in the random oracle model are not strong security proofs, because of the underlying ideal assumption. They do, however, provide strong evidence for security and can furthermore serve as the basis for quite efficient schemes. Since people do not often want to pay more than a negligible price for security, such an argument for practical schemes is more useful than formal security proofs for inefficient schemes.

3.2 Description of OAEP

At the time Bleichenbacher published his attack on RSA-PKCS #1 v1.5, the only efficient and "provably secure" encryption scheme based on RSA was the Optimal Asymmetric Encryption Padding (OAEP) proposed by Bellare and Rogaway [*BeRo94*]. OAEP can be used with any trapdoor permutation f. To encrypt a message m using the encryption scheme f-OAEP, first apply the OAEP procedure described in Figure 2. Here r is a random string and G,H are hash functions. The resulting values [s||t] are then encrypted using f, namely $C = f(s,t)$.

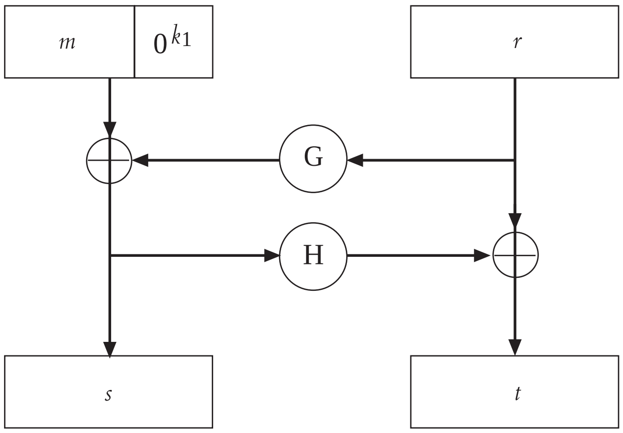


Figure 2: OAEP Padding

Bellare and Rogaway proved that OAEP padding used with any trapdoor permutation f provides a semantically secure encryption scheme. By adding some redundancy (the constant value 0^{k_1} at the end of the message, as shown in Figure 2), they furthermore proved it to be *weakly* plaintext-aware. Plaintext-awareness is a property of encryption schemes in the random oracle model which means that there exists a plaintext-extractor able to simulate the decryption oracle on any ciphertext (valid or not) designed by the adversary. The *weak* part in the definition proposed by Bellare and Rogaway was that the plaintext-extraction was just required to work while the adversary had not received any valid ciphertext from any source. Unfortunately, the adaptive chosen-ciphertext attack model gives the adversary a full-time access to the decryption oracle, even after receiving the challenge ciphertext about which the adversary wants to learn information. This challenge is a valid ciphertext. Therefore, semantic security together with *weak* plaintext-awareness only implies the semantic security against non-adaptive chosen-ciphertext attacks (*a.k.a.* lunchtime attacks [*NaYu89*], or *indifferent* chosen-ciphertext attacks), where the decryption oracle access is limited until the adversary has received the challenge ciphertext.

In 1998, Bellare, Desai, Rogaway and the author [*BeDePoRo98*] corrected this initial definition of plaintext-awareness, requiring the existence of a plaintext-extractor able to simulate the decryption oracle on any ciphertext submitted by the adversary, even after seeing some valid ciphertexts not encrypted by the adversary himself. This stronger definition is a more accurate model of the real world, where the adversary may have access to ciphertexts via eavesdropping. We furthermore proved that this new property (which can only be defined in the random oracle model) actually provides the encryption scheme with the strongest security level, namely semantic security against (adaptive) chosen-ciphertext attacks (IND-CCA). However, no one ever provided OAEP with such a new plaintext-extractor. Therefore, even if everybody believed in the strong security level of OAEP, it had never been proven IND-CCA under the one-wayness of the permutation alone.

3.3 The OAEP Security Analyses

In fact, the only formally proven security result about OAEP was its semantic security against lunchtime attacks, assuming the one-wayness of the underlying permutation. Until very recently OAEP was widely believed to also be IND-CCA.

3.3.1 Shoup's Result

Shoup [*Sh01*] recently showed that it was quite unlikely that OAEP is IND-CCA assuming only the one-wayness of the underlying trapdoor permutation. In fact, he showed that if there exists a trapdoor one-way permutation g for which it is easy to compute $g(x \oplus a)$ from $g(x)$ and a, then OAEP cannot be IND-CCA secure for an arbitrary trap-door permutation f. Referring to this special property of g as "XOR malleability," let us briefly present Shoup's counter-example. Let $s||t$ denote the output of the OAEP transformation on a plaintext message m. Define the one-way permutation f as $f(s||t) = s || g(t)$. Then encrypting m using f-OAEP gives the ciphertext $C = [s || g(t)]$.

What Shoup showed is that under these conditions the adversary can use C to construct a ciphertext C' of a plaintext message m' that is closely related to the message m. In particular, for any string δ, the adversary can construct C' which is the encryption of $m' = m \oplus \delta$. Thus, the scheme is malleable and hence not IND-CCA — giving C' to the decryption oracle will reveal $m' = m \oplus \delta$, from which the adversary can obtain m.

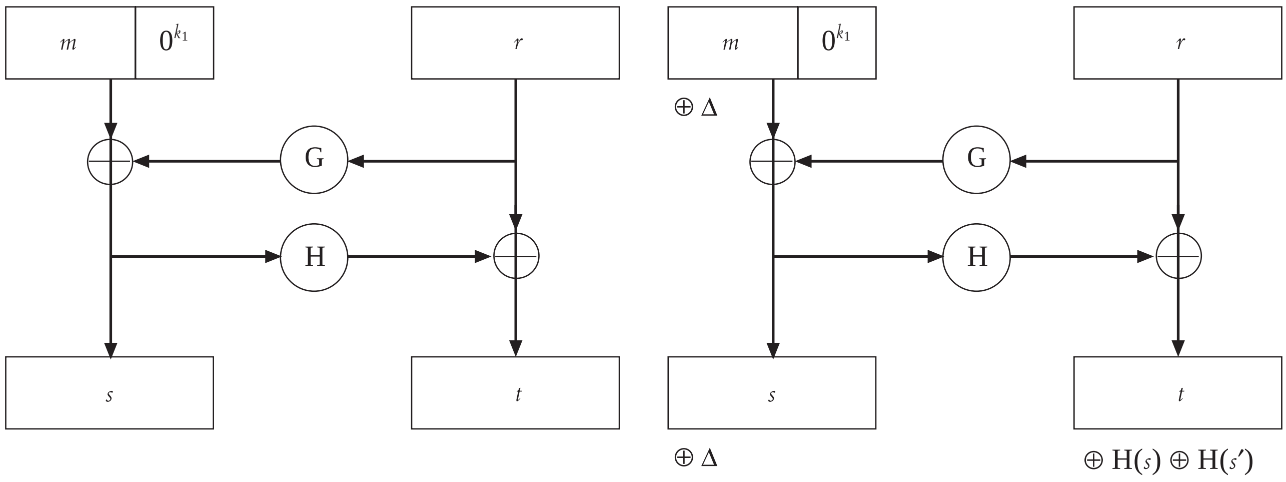


Figure 3: Shoup's Attack

To construct C', the idea is for the adversary to exploit the explicit appearance of s in the ciphertext C. The adversary first computes $s' = s \oplus \Delta$, where $\Delta = \delta || 0^{k1}$; essentially, Δ is simply a padded rendering of δ. The adversary then computes $D = H(s) \oplus H(s')$ using explicit knowledge of s and s' and access to the random oracle for H. Finally, by exploiting the "XOR malleability" of g, the adversary computes $g(t')$, where $t' = t \oplus D$. It is easy to see now that $C' = s' || g(t')$ is a valid encryption of the message m'. Hence, the non-malleability of f-OAEP is broken.

This observation shows that it is unlikely that one can prove that f-OAEP is IND-CCA secure for arbitrary trapdoor permutations f by assuming only the one-wayness of f.

3.3.2 Repairing the OAEP Proof of Security

To construct a valid ciphertext C' in the above attack it seems that the adversary has to query the hash function H at $H(s)$. But this seems to imply that given C the adversary can figure out the value s used to create C (recall that s is the left hand side of $f^{-1}(C)$). Thus, it appears that in order to mount Shoup's attack the adversary must be able partly to invert f – given $f(s, t)$, the adversary must be able to expose s.

We say f is partial-domain one-way if no efficient algorithm can deduce s from $C = f(s, t)$. For such trapdoor permutations f, one could hope that Shoup's attack will fail and that f-OAEP is IND-CCA secure. Fujisaki, Okamoto, Stern and the author [FuOkPoSt01] formally proved this fact: If f is partial-domain one-way, then f-OAEP is IND-CCA secure. We note that partial-domain one-wayness is a stronger property than one-wayness: a function might be one-way but still not partial-domain one-way.

Fortunately, the homomorphic properties of RSA enable us to prove that the RSA permutation is partial-domain one-way if and only if RSA is one-way. More precisely, an algorithm that can expose half of $\text{RSA}^{-1}(C)$ given C can be used to completely invert the RSA permutation. Altogether, this proves the widely believed IND-CCA security of RSA-OAEP assuming that RSA is a trapdoor permutation. For security parameters ε and t (whose formal definitions are omitted here), we obtain the following result [FuOkPoSt01]:

Let A be a CCA-adversary against the "semantic security" of RSA-OAEP with running time bounded by t and advantage ε. Then, the RSA function can be inverted with probability greater than approximately $\varepsilon^2/4$ within time bound $2t$.

Unfortunately, the security reduction from an RSA-inversion into an attack is quite inefficient for practical sizes (more precisely, it is quadratic in the number of oracle queries). Hence, this reduction is meaningless unless one uses a modulus large enough so that the RSA-inversion (or the factorization) requires much more than 2^{150} computational effort. With current factorization techniques [LeLe93, CaDoLe00], one needs to use a modulus of length more than 4096 bits to make the reduction meaningful (see [LeVe00] for complexity estimates of the most efficient factoring algorithms). Viewed another way, this reduction shows that a 1024-bit modulus just provides a provable security level of 2^{40}, which is clearly inadequate given currently prevalent levels of computing power. (We note, however, that this does not meant that there is an attack with this low complexity, only that one cannot be ruled out by the available proofs of security.)

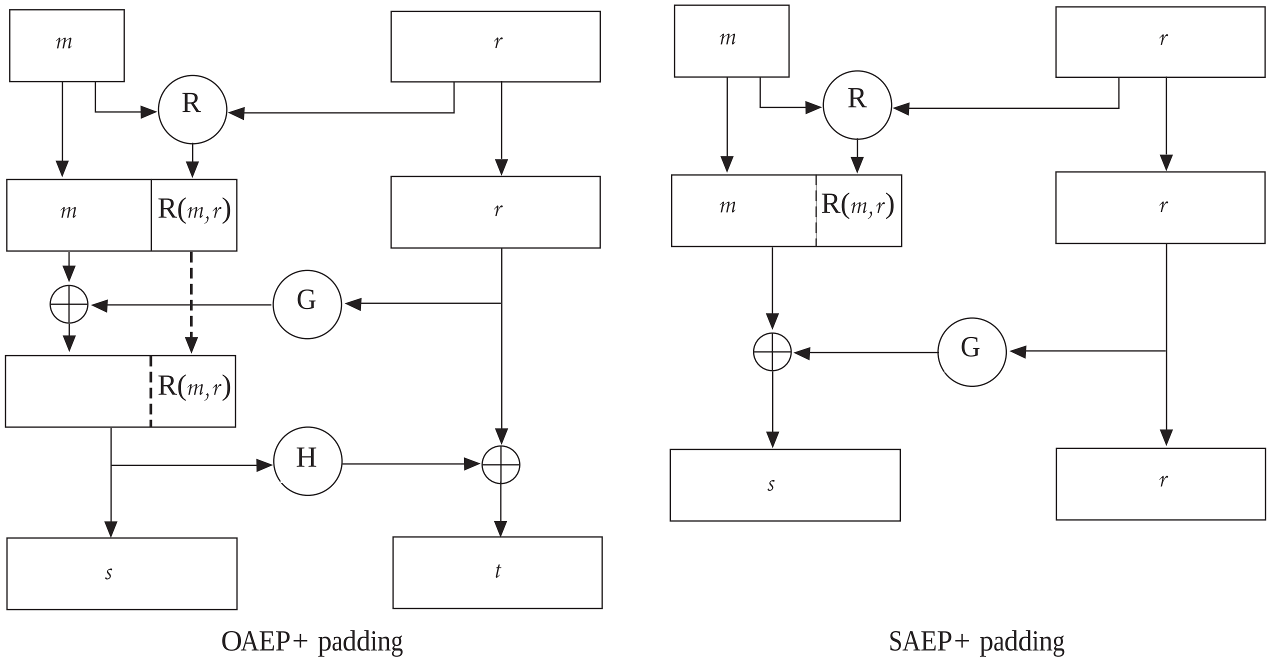


Figure 4: OAEP+ and SAEP+ padding

4. OAEP ALTERNATIVES

4.1 The OAEP+ Padding

Shoup also proposed a formal security proof of RSA-OAEP with a much more efficient security reduction, but in the particular case where the encryption exponent e is equal to 3. However, many people believe that the RSA trapdoor permutation with exponent 3 may be weaker than with greater exponents. Therefore, he also proposed a slightly modified version of OAEP, called OAEP+ (see Figure 4), which can be proven secure under the one-wayness of the permutation alone. It uses the variable redundancy $R(m,r)$ instead of the constant 0^{k_1}. It is thus a bit more intricate than the original OAEP. The security reduction for OAEP+ is efficient, but still runs in quadratic time.

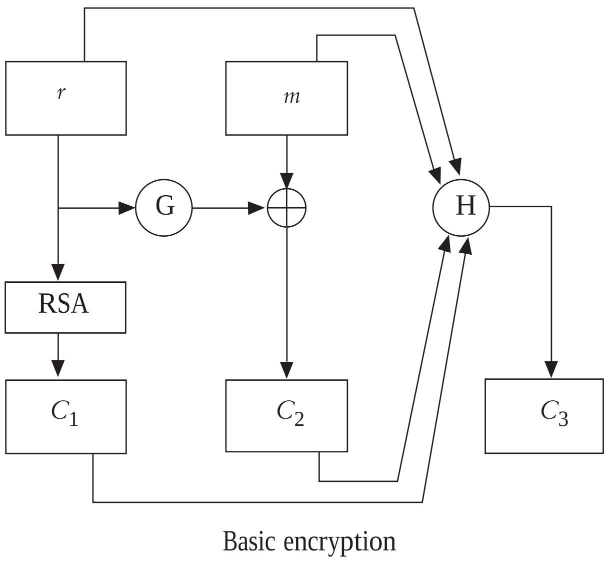
4.2 SAEP+ Padding

Boneh [Bo01] recently proposed a new padding scheme, SAEP+, to be used with the Rabin primitive [Ra78] or RSA. It is simpler than OAEP, hence the name Simplified Asymmetric Encryption Padding: whereas OAEP is a two-round Feistel network, SAEP+ is a single-round. SAEP+ has a linear time reduction for the Rabin system (i.e., $e = 2$). For larger exponents, SAEP+ has a quadratic time reduction. Hence, for larger exponents ($e > 2$), SAEP+ does not guarantee security for practical parameters (less than two thousand bits).

4.3 The REACT Construction

Another alternative to OAEP is the REACT construction, proposed by Okamoto and the author [OkPo01] (see Figure 5). It provides an IND-CCA encryption scheme from any weakly secure one (more precisely, a one-way primitive, against plaintext-checking attacks), such as the RSA primitive. Therefore, the RSA-REACT scheme is IND-CCA secure under the RSA assumption.

Furthermore, the security reduction is very efficient, since it is in linear time without any loss in the success probability, whatever the exponent. Consequently, it guarantees perfect equivalence with RSA inversion for moduli which require just a bit more than 2^{70} effort to be factored. This is the case for 1024 bit-long moduli, the minimal currently advised key size. In comparison to previous proposals, REACT is a full scheme and not just a pure padding applied to the message before the RSA function.



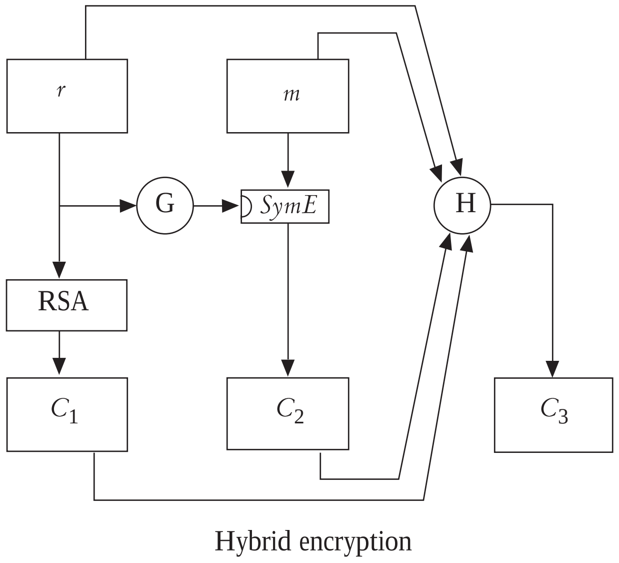


Figure 5: REACT

Consequently, the ciphertext is a bit longer. However, even when used for key transport, it allows integration of a symmetric encryption scheme (SymE) to achieve very high encryption rates, as shown in the hybrid construction. In the specific case of RSA, REACT can be optimized, as explained below.

4.4 Simple RSA

In an ISO report [Sh01b], Shoup suggested a possible alternative, based on ideas from Bellare and Rogaway [BeRo93] that provide a secure encryption scheme from any trapdoor one-way permutation f. Roughly speaking, "simple RSA," as it is called, consists of first encrypting a random string r using f to obtain C_0 (thus $C_0 = r^e \bmod n$), and then parsing $G(r)$ as $k_0 \,||\, k_1$, where G is some hash function (modeled by a random oracle). Thereafter, one encrypts the message m using a symmetric encryption scheme with the key k_0 to get C_1 (*e.g.*, $C_1 = m \oplus k_0$), and authenticates the ciphertext with a MAC function H using the key k_1 to get a tag $T = H(k_1, C_1)$. The ciphertext is the triple (C_0, C_1, T). This construction is a special case of REACT, optimized for RSA, and hence is IND-CCA under the RSA assumption. It provides a very efficient linear time reduction. Moreover, thanks to the random self-reducibility of RSA (which can only be used with this latter construction, but cannot with the OAEP and SAEP variants), this construction provides a high security level even when encrypting many plaintexts [BaPoSt00, BeBoMi00].

5. CONCLUSION

RSA-OEP is a practical RSA encryption scheme with provable security in the random oracle model. For practical security, the cost of the reductions cannot simply be shown to be polynomial time (as in asymptotical analyses), since the reduction efficiency directly impacts the security parameters needed for the scheme. Hence, when evaluating cryptographic constructions, one must take into account the efficiency of the security proof. Inefficient proofs of security do not give security guarantees for real world parameters.

Only OAEP with exponents 2 or 3, SAEP+ with exponent 2, and RSA-REACT (or the optimization "simple RSA") with any exponent, admit formal proofs with linear time reductions in the random oracle model. Hence only these schemes guarantee semantic security against chosen-ciphertext attacks for practical modulus sizes (even less than 1024 bits). The provable security for other padding schemes is meaningful only for much larger moduli (more than 4096 bits).

ACKNOWLEDGEMENTS

I warmly thank my co-authors, Mihir Bellare, Anand Desai, Eiichiro Fujisaki, Tatsuaki Okamoto, Phil Rogaway and Jacques Stern for the interesting works we did on asymmetric encryption, as well as Dan Boneh, Pierre-Alain Fouque, Victor Shoup and Yves Verhoeven for the fruitful discussions we had.

REFERENCES

[BaPoSt00] O. Baudron, D. Pointcheval, and J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. *In Proc. of the 27th ICALP*, LNCS 1853, pages 499-511. Springer-Verlag, Berlin, 2000

[BeBoMi00] M. Bellare, A. Boldyreva, and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In *Eurocrypt '00*, LNCS 1807, pages 259-274. Springer-Verlag, Berlin, 2000.

[BeDePoRo98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26-45. Springer-Verlag, Berlin, 1998.

[BeRo93] M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62-73. ACM Press, New York, 1993.

[BeRo94] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption — How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92-111. Springer-Verlag, Berlin, 1995.

[BeRo96] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399-416. Springer-Verlag, Berlin, 1996.

[BeSh99] M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto '99*, LNCS 1666, pages 519-536. Springer-Verlag, Berlin, 1999.

[BiSh97] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Crypto '97*, LNCS 1294, pages 513-525. Springer-Verlag, Berlin, 1997.

[Bl98] D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In *Crypto '98*, LNCS 1462, pages 1-12. Springer-Verlag, Berlin, 1998.

[BlM84] M. Blum, S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo Random Bits, *SIAM Journal on Computing*, Vol.13, pages 850-864, 1984.

[Bo01] D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In *Crypto '01*, LNCS 2139, pages 275-291. Springer-Verlag, Berlin, 2001.

[BoDeLi97] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Eurocrypt '97*, LNCS 1233, pages 37-51. Springer-Verlag, Berlin, 1997.

[CaGoHa98] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209-218. ACM Press, New York, 1998.

[CaDoLe00] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA Modulus. In *Eurocrypt '00*, LNCS 1807, pages 1-18. Springer-Verlag, Berlin, 2000.

[DoDwNa00] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391-437, 2000.

[FiSh86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186-194. Springer-Verlag, Berlin, 1987.

[FuOkPoSt01] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is Secure under the RSA Assumption. In *Crypto '01*, LNCS 2139, pages 260-274. Springer-Verlag, Berlin, 2001.

[Go97] O. Goldreich. On the Foundations of Modern Cryptography. In *Crypto '97*, LNCS 1294, pages 46-74. Springer-Verlag, Berlin, 1997. Also appeared in *CryptoBytes*, 3(2):1-5, Autumn 1997.

[GoMi84] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270-299, 1984.

[GoHaSc99] C. Hall, I. Goldberg, and B. Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. In *Proc. of ICICS '99*, LNCS, pages 2-12. Springer-Verlag, 1999.

[Ko96] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Crypto '96*, LNCS 1109, pages 104-113. Springer-Verlag, Berlin, 1996.

[KoJaJu99] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Crypto '99*, LNCS 1666, pages 388-397. Springer-Verlag, Berlin, 1999.

[LeLe93] A. Lenstra and H. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.

[LeVe00] A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In *PKC '00*, LNCS 1751, pages 446-465. Springer-Verlag, Berlin, 2000.

[NaYu89] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *Proc. of the 21st STOC*, pages 33-43. ACM Press, New York, 1989.

[OkPo01] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT-RSA '01*, LNCS 2020, pages 159-175. Springer-Verlag, Berlin, 2001.

[PoSt00] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361-396, 2000.

[Ra78] M. O. Rabin. Digitalized Signatures. In R. Lipton and R. De Millo, editors, *Foundations of Secure Computation*, pages 155-166. Academic Press, New York, 1978.

[RaSi91] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433-444. Springer-Verlag, Berlin, 1992.

[RiShAd78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120-126, February 1978.

[Sh01] V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239-259. Springer-Verlag, Berlin, 2001.

[Sh01b] V. Shoup. Editor's Contribution on Public Key Encryption. ISO / IEC JTC 1 / SC27. February 13, 2001.

[TLS] T. Dierks and C. Allen. The TLS Protocol, January 1999. RFC 2246.

III. Composite-Residuosity Based Cryptography: An Overview

Pascal Paillier

Cryptography Group, Gemplus

pascal.paillier@gemplus.com

1. INTRODUCTION

There are three main families of public key cryptosystems based on computational number theory. The first family includes RSA and related variants (Rabin-Williams, LUC, Dickson, elliptic curve embodiments of RSA-like KMOV). The trapdoor for these relies on the extraction of roots over finite Abelian groups of some secret order. Root extraction is easy when the group order is known, but believed to be hard without that knowledge. Finding the group order is as hard as factoring a large integer.

The second family is based on Diffie-Hellman-type schemes (ElGamal and variants, Cramer-Shoup) which exploit properties of exponentiation over finite cyclic groups. Here, the trapdoor depends on the knowledge of the discrete logarithm of some public group element and again, computing this secret information from the description of the group alone is believed to be hard.

Finally, the third family is based on high degree residuosity classes (Goldwasser-Micali, Benaloh, Naccache-Stern, Okamoto-Uchiyama and variants). The trapdoor in these schemes combines the extraction of residuosity classes over certain groups with the intractability of computing their order. Because residuosity classes are additive, such cryptosystems look like discrete-log based ones, but the trapdoor is closer in nature to those for factoring-based systems.

We review here one particular cryptosystem belonging to this last family that was proposed by the author of this paper at Eurocrypt '99. The system, based on composite residuosity classes, has recently gained a certain degree of popularity mainly as a building block for cryptographic protocols. We summarize some of these constructions and provide state-of-the-art references to composite-residuosity-based cryptographic tools.

2. DESCRIPTION OF THE SCHEME

We first briefly recall basic facts on composite residues, referring the reader to [8] for more details.

- We set $n = pq$ where p and q are large primes and denote by $\phi(n)$ and $\lambda = \lambda(n)$ the Euler and Carmichael functions of n respectively. Then $\phi(n) = (p\text{-}1)(q\text{-}1)$ and $\lambda = \text{lcm}(p\text{-}1, q\text{-}1)$.
- It is a well-known fact that the group $\mathbb{Z}^*_{n^2}$ has $\phi(n^2) = n\phi(n)$ elements. Furthermore, $w^\lambda = 1 \bmod n$ and $w^{n\lambda} = 1 \bmod n^2$ for any element $w \in \mathbb{Z}^*_{n^2}$.
- We say that an integer z is an n-residue modulo n^2 if there exists some $y \in \mathbb{Z}^*_{n^2}$ such that $z = y^n \bmod n^2$. The set of n-residues forms a subgroup of $\mathbb{Z}^*_{n^2}$ of order $\phi(n)$. Each n-residue in $\mathbb{Z}^*_{n^2}$ has exactly n roots of degree n.
- We denote by B the set of elements of order $n\alpha$ for some $\alpha \in [1, \lambda]$.
- Let $g \in \mathbb{Z}^*_{n^2}$ and consider the mapping over $Z_n \times \mathbb{Z}^*_{n^2}$ defined by:

  $$(x,y) \mapsto g^x y^n \bmod n^2.$$

  Then, when $g \in B$, this map is one-to-one.
- Let $g \in B$. We define the <u>n-residuosity class</u> of an element $w \in \mathbb{Z}^*_{n^2}$ with respect to g as the unique integer $x \in [1, n]$ for which there exists $y \in \mathbb{Z}^*_{n^2}$ s.t. $w = g^x y^n \bmod n^2$.

  Following the notation of Benaloh [2], we denote the class of w by $[w]_g$. Note that $[w]_g = 0$ if and only if w is an n-residue. Additionally,

  $$\forall w_1, w_2, \qquad [w_1 \cdot w_2]_g = [w_1]_g + [w_2]_g.$$

  Hence, the class function $w \mapsto [w]_g$ is an additive homomorphism for any $g \in B$.
- Consider the subgroup of Z_{n^2} defined by $S_n = \{u : u = 1 \bmod n\}$. For $u \in S_n$ define:

  $$L(u) = (u - 1)/n.$$

  Then for $u \in S_n$ we have $L(u^r)/L(u) = r$. Hence, discrete-log is easy in the group S_n.

We are now ready to define the composite residuosity cryptosystem. Let $n = pq$ and $g \in B$. The public key is the pair (n,g) while the factors (p,q) are the private key. The cryptosystem is as follows.

Encryption:

Plaintext: $0 \leq m < n$
select a random $0 < r < n$
Ciphertext: $c = g^m r^n \bmod n^2$

Decryption:

Ciphertext: $0 < c < n^2$

Plaintext: $m = \dfrac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$

We discuss the security of this system below. This cryptosystem is useful for distributed computations due to its additive homomorphism. That is, for all m_1, m_2 :

$$D(\ E(m_1) \cdot E(m_2) \ mod \ n^2\ ) = m_1 + m_2 \ mod \ n$$

In other words, given the two ciphertexts $E(m_1)$, $E(m_2)$ it is easy to construct the encryption of $m_1 + m_2$.

The additive property is particularly useful when designing threshold crypto-systems and distributed protocols in general. It also allows full self-randomization of encryptions in the sense that any ciphertext can be transformed into another without affecting the plaintext.

3. THE CLASS PROBLEM

We discuss several complexity assumptions needed for the security of the above cryptosystem.

The problem of distinguishing the set of n-residues from non n-residues in Z_{n^2} is denoted by $CR[n]$. This problem has a random-self reduction in Z_{n^2} (reduce $CR[n]$ for a worst case element $x \in Z_{n^2}$ to a random element). The assumption that $CR[n]$ is polynomial-time intractable is referred to as the Decisional Composite Residuosity Assumption (DCRA).

The cryptosystem above is semantically secure (against chosen-plaintext attacks) assuming the DCRA assumption.

The n-Residuosity Class Problem in base g, denoted $Class[n,g]$, is defined as the problem of computing the class function in base g. This is exactly the problem of decrypting a given ciphertext in the cryptosystem above.

As before, one can show that $Class[n,g]$, is random self-reducible over its inputs. Moreover, for any $w \in Z^*_{n^2}$ and $g_1 g_2 \in B$, we have

$$[w]_{g_1} = [w]_{g_2}\ [g_2]_{g_1} \bmod n$$

which implies that $Class[n,g]$ is also random self-reducible over $g \in B$. Hence, $Class[n]$ is a computational number-theoretic problem which only depends on n, very much like factorization for instance. The Class problem is related to other standard hard number theoretic problems. For example, the decryption procedure shows that:

$$\frac{L(w^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} = [w]_g \bmod n.$$

Hence, if we can factor n and obtain λ then we can solve $Class[n]$. Therefore, we write:

$$Class[n] \Leftarrow Fact[n].$$

One can show a slightly stronger statement. $Class[n]$ can be solved just given the ability to compute n'th roots in Z_n. Computing n'th roots in Z_n is called the RSA problem with public exponent $e = n$ and is denoted by $RSA[n,n]$. Therefore, we have:

$$Class[n] \Leftarrow RSA[n,n].$$

In summary, the computational hierarchy behind composite residuosity is

$$CR[n] \Leftarrow Class[n] \Leftarrow RSA[n,n] \Leftarrow Fact[n].$$

We conjecture that $Class[n]$ is polynomial-time intractable; by analogy with the DCRA, this conjecture is called Computational Composite Residuosity Assumption (CCRA for short). We know that if the DCRA is true then the CCRA is true, but the converse implication remains open.

A careful study of $CR[n]$ and $Class[n]$ is essential in future research, because very few things are known about these problems today. We note that the encryption scheme shown above is one-way relative to the CCRA and semantically secure (against chosen-plaintext attacks) relative to the DCRA.

In [4], Catalano, Gennaro and Howgrave-Graham examined the bit security of our scheme. They showed that given a random element $w \in \mathbb{Z}^*_{n^2}$, predicting the least significant bit of $[w]_g$ is as hard as computing $[w]_g$ completely. Moreover, they proved that the scheme simultaneously hides $|n| - b$ bits of $[w]_g$ under the assumption that computing classes remains hard over $\{w : [w]_g < 2^b\}$. By encrypting random-padded messages, the authors deduced from their results a way to construct the first encryption scheme hiding $O(|n|)$ plaintext bits. Note that although their modified version of the class problem seems to remain hard in this context, further research on its connections with the original class problem (as well as on possible breakthroughs) is required to validate this approach.

4. CRYPTOGRAPHIC APPLICATIONS

We now give some cryptographic applications of composite residuosity. Without being exhaustive, composite residuosity finds applications in such different fields as encryption, signatures, distributed protocols such as voting schemes and ZK proofs. It is worthwhile noting that among all residuosity-based schemes, taking $g = 1+kn$ for some k leads to higher encryption rates as $g^m = 1+kmn$. Because of random self-reducibility, this choice does not affect the security level.

4.1 A Subgroup Variant

We give here a slightly modified encryption scheme in which the ciphertext space is restricted to the subgroup $\langle g \rangle$. Indeed, assuming that g is of order $n\alpha$, we have for any $w \in \langle g \rangle$,

$$[w]_g = \frac{L(w^\alpha \bmod n^2)}{L(g^\alpha \bmod n^2)} \bmod n.$$

This motivates the following cryptosystem.

Encryption:

Plaintext: $0 \leq m < n$
select a random $0 < r < n$
ciphertext: $c = g^{m+rn} \bmod n^2$

Decryption:

Ciphertext: $0 < c < n^2$

Plaintext: $m = \dfrac{L(c^\alpha \bmod n^2)}{L(g^\alpha \bmod n^2)} \bmod n$

This time, the secret key is α instead of λ. The most expensive operation while decrypting is the modular exponentiation $c^a \bmod n^2$, which can be accelerated arbitrarily by an adequate selection of α. In practice, α should be typically set to a 320-bit divisor of λ such that $\alpha = \alpha_p\alpha_q$ where α_p divides p-1 but not q-1 and α_q divides q-1 but not p-1. This can be met using an appropriate key generation algorithm.

In this subgroup variant, one-wayness does not rely on the composite residuosity class problem, because the ciphertext is known to lie in $\langle g \rangle$. The problem consisting in computing residuosity classes in this context is called Partial Discrete Logarithm Problem and is a weaker instance of the class problem. Similarly, we call Decisional Partial Discrete Logarithm Problem the problem of distinguishing n-residues given the public information. The semantic security of the encryption scheme is equivalent to this problem.

4.2 Extended Variant

In [15], Damgård and Jurik introduced a modified cryptosystem in which computations are performed modulo n^{s+1} where $s \geq 1$. Clearly, the original scheme is contained by setting $s = 1$. Damgård and Jurik's extended scheme relies on the observation that for any $g \in Z^*_{n^{s+1}}$ such that n^s divides the order of g modulo n^{s+1}, the function defined over $Z_{n^s} \times Z^*_n$ by

$$(x,y) \mapsto g^x y^{n^s} \bmod n^{s+1}$$

is one-to-one. As a result, n^s-residuosity classes are easily definable in this context and present the same features than in the original system. The particular choice $g = 1 + n$ (it is easily shown that the order of $1 + n$ modulo n^{s+1} is n^s) provides the advantage of reducing the key size without modifying the system's properties (including security). The final observation is that computing m from $w = (1 + n)^m \bmod n^{s+1}$ is easy. Define like in the original setting $L(x) = (x - 1)/n$. Clearly,

$$L\left((1+n)^m \bmod n^{s+1}\right) = \left(m + \binom{m}{2} n + \cdots + \binom{m}{s} n^{s+1}\right) \bmod n^s$$

Damgård and Jurik then give an inductive method to compute $m_i = m \bmod n^i$ for successive values of $i \in [1,s]$. A simple alternative to their method is obtained by observing that $(1 + n)^{n^i} = 1 + n^{i+1} \bmod n^{i+2}$ for any i, so we actually have the more direct induction

$$m_{i+1} = m_i + L(w(1+n)^{-m_i} \bmod n^{i+2}),$$

which also allows us to recover $m = m_s$. Damgård and Jurik's cryptosystem is described as follows.

Key Generation: choose an RSA modulus $n = pq$. The public key is n while the secret key is (p,q).

Encryption: given a plaintext $m < n^s$, choose a random $r < n^{s+1}$ and let the ciphertext be

$$c = (1 + n)^m r^{n^s} \bmod n^{s+1}.$$

Decryption: compute d such that $d = 1 \bmod n^s$ and $d = 0 \bmod \lambda$ (this may also be saved as some secret key material). Given the encryption c, compute $c^d = (1+n)^m \bmod n^{s+1}$ and apply the above algorithm to recover m.

The one-wayness of this scheme is based on the assumption that the class function is hard to compute in this context without knowledge of (p,q). Similarly, semantic security is achieved if and only if distinguishing n^s-residues in $Z^*_{n^{s+1}}$ is intractable. These assumptions were called Generalised (Decisional) Composite Residuosity Assumption or G(D)CRA and conjectured true by the authors. It is easily seen that original assumptions imply Damgård and Jurik's generalized assumptions.

4.3 Digital Signatures

Trapdoor permutations are extremely rare objects: we refer the reader to [12] for an exhaustive documentation on these. Here, we show how composite residuosity allows to design a trapdoor permutation. As before, n stands for the product of two large primes and $g \in B$.

Encryption:

plaintext $m < n^2$
split m into $m = m_1 + n m_2$
ciphertext $c = g^{m_1} m_2^n \bmod n^2$

Decryption:

ciphertext $c < n^2$

compute $m_1 = \dfrac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n,$

compute $c' = c g^{-m_1} \bmod n,$
compute $m_2 = c'^{\,n^{-1} \bmod \lambda} \bmod n,$
plaintext $m = m_1 + n m_2$.

As easily seen in the decryption procedure, we require the extraction of an n-root modulo n. Because of this additional step, we get that this permutation is one-way if and only if $RSA[n,n]$ is hard. Like with any other trapdoor permutation, digital signatures are obtained by using the cryptosystem backwards : denoting by μ: $\{0,1\}^* \mapsto \{0,1\}^k$ some padding function with $k = |n^2|$, we obtain a signature scheme as follows. For a given message m, the signer computes the signature (s_1, s_2) where

$$s_1 = \frac{L(\mu(m)^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n,$$

$$s_2 = (\mu(m) g^{-s_1})^{n^{-1} \bmod \lambda} \bmod n$$

and the verifier checks that

$$\mu(m) = g^{s_1} s_2^n \bmod n^2.$$

4.4 A Distributed Version

In [15], Damgård and Jurik devised a distributed cryptosystem allowing threshold decryption among a set of servers. Fouque, Poupard and Stern independently proposed a similar technique in [6]. This threshold variant is an adaptation of Shoup's distributed RSA [17] whose main part allows a set of servers to collectively and efficiently raise an input number to a secret exponent modulo an RSA modulus. On input c, each server returns a share of the result, together with a proof of correctness. Given sufficiently many correct shares, these can be efficiently combined to compute c^d mod n, where d is the secret exponent. Damgård and Jurik transplanted this method in the case of a shared exponentiation modulo n^{s+1}.

Assume that there are l decryption servers, and that a minimum of k of these are needed to make a correct decryption.

Key Generation: pick a pair of primes p and q satisfying $p = 2p'+1$ and $q = 2q'+1$ for primes p' and q'. Set $n = pq$, $m = p'q'$ and decide on some $s > 0$, so that the plaintext space is Z_{n^s}. Then pick an number d to satisfy $d = 1 \bmod n^s$ and $d = 0$ mod m. Now choose a polynomial

$$f(X) = \sum_{i=0}^{k-1} a_i X^i \bmod n^s m$$

by picking random coefficients $a_i \in Z_{n^s m}$ for i ranging from 1 to k–1 and $a_0 = d$. The secret share of server i is $s_i = f(i)$ for $i \in [1, l]$ while the public key is n. To verify the actions of the decryption servers, the system requires the following fixed public values: v, generating the cyclic group of squares in $Z^*_{n^{s+1}}$ and for each decryption server a verification key $v_i = v^{\Delta s_i} \bmod n^{s+1}$ where $\Delta = l!$. The entire key setup may be executed by a trusted party, or distributed among servers using suitable multiparty computation techniques.

Encryption: to encrypt a message m, a random $r < n^{s+1}$ is picked and the ciphertext is computed as $c = (1+n)^m r^{n^s}$ mod n^{s+1}.

Share Decryption: the server i computes $c_i = c^{2\Delta s_i}$ where c is the ciphertext and provides a zero-knowledge proof that $\log_{c^4}(c_i^2) = \log_v(v_i)$ which shows that he has indeed raised c to his secret exponent s_i.

Share Combining: given a subset S of k (or more) shares with a correct proof, the result is obtained by combining the shares into

$$c' = \prod_{i \in S} c_i^{2\lambda^s_{0,i}} \bmod n^{s+1} \text{ where } \lambda^s_{0,i} = \Delta \prod_{j \in S-i} \frac{-i}{i-j}.$$

We then get

$$c' = c^{4\Delta^2 f(0)} = c^{4\Delta^2 d} = (1+n)^{4\Delta^2 m} \bmod n^{s+1}.$$

The plaintext m is retrieved by applying the induction formula described in the extended variant and multiplying the result by $(4\Delta^2)^{-1} \bmod n^s$.

The authors then showed that this threshold version is as secure as their extended variant in the random oracle model, provided that some trusted player performs the share combining stage. More recently, Damgård and Koprowski proposed a new threshold RSA technique [16] applicable *mutatis mutandis* to the present setting. Its main advantage over [15] resides in that no trusted dealer is needed.

4.5 Other Applications

Boneh and Franklin [3] introduced a traitor tracing scheme in which black box tracing is achieved using the subgroup variant.

Pointcheval and the author of these lines [10] proposed security-enhanced cryptosystems provably semantically secure against chosen-ciphertext attacks in the random oracle model.

In [14], Poupard and Stern use the subgroup encryption scheme to devise proofs of knowledge for the factorization of a public composite integer. In [13], the same authors further achieve fair encryption of secret keys, a clever and efficient approach to key recovery systems.

Yung and I considered self-escrowed public-key infrastructures [11], in which a joint use of Paillier and ElGamal encryption schemes leads to a simplified implantation of PKI properties.

Cramer, Damgård and Nielsen [5] propose a way of basing multiparty computation protocols on homomorphic threshold crypto-systems instead of using secret sharing schemes. Their general construction is shown to reach a better efficiency in that fewer bits are needed to be transmitted between parties, while security against cheating is preserved for any minority of cheaters.

Galbraith [7] recently showed how to securely design a composite-residuosity-based encryption scheme on non-specific elliptic curves over rings. This implicitly provided an answer to the quest of [9].

More recently, Baudron and Stern [1] exploited our scheme's homomorphic property to design a new auction protocol where bids are submitted non-interactively and bidders are not required to interact with each other.

Even more recently, Cramer and Shoup used our scheme to propose an encryption scheme secure against active adversaries in the standard model [6]. They based their scheme's security on the decisional composite residuosity assumption.

5. IMPLEMENTATION ISSUES

5.1 Efficiency

The reader may find in [8] some tips about practical aspects of computations required by composite residuosity-based cryptosystems, as well as various implementation strategies for increased performance. We just recall here the main tricks: decryption allows Chinese remaindering; preprocessing can be used advantageously in both encryption and decryption; small values for g or setting $g = 1+n$ (which does not affect security at all) would greatly improve encryption rates, provided that $g \in B$ still holds. In the same spirit as with RSA, simple randomization techniques may help protect hardware or software implementations against side-channel attacks.

REFERENCES

1. Olivier Baudron and Jacques Stern. Non-interactive private auctions. In *Financial Crypto'01*, Lecture Notes in Computer Science, pages 300-313. Springer-Verlag, 2001.

2. Josh Cohen Benaloh. Verifiable Secret-Ballot Elections. PhD thesis, Yale University, 1988.

3. Dan Boneh and Matthew Franklin. An efficient public key traitor tracing scheme. In *Crypto '98*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

4. Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. In Birgit Pfitzmann, editor, *Eurocrypt* '01, volume 2045 of Lecture Notes in Computer Science, pages 229-243. Springer-Verlag, 2001.

5. Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. In Bart Preneel, editor, *Eurocrypt* '00, volume 1807 of Lecture Notes in Computer Science, pages 280-300. Springer-Verlag, 2000.

6. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. Available on IACR ePrint Archive

   (http://eprint.iacr.org/2001/085).

7. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography 2000*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

8. Steven D. Galbraith. Elliptic curve Paillier schemes. Journal of Cryptology, 2001. to appear.

9. Pascal Paillier. Public-key cryptosystems based on discrete logarithms residues. In *Eurocrypt* '99, volume 1592 of Lecture Notes in Computer Science, pages 223-238. Springer-Verlag, 1999. European patent number 9900341.

10. Pascal Paillier. Trapdooring discrete logarithms on elliptic curves over rings. In T. Okamoto, editor, Asiacrypt'00, volume 1976 of Lecture Notes in Computer Science, pages 573-584. Springer-Verlag, 2000.

11. Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In K. Y. Lam and E. Okamoto, editors, *Asiacrypt* '99, volume 1716 of Lecture Notes in Computer Science, pages 165-179. Springer-Verlag, 1999.

12. Pascal Paillier and Moti Yung. Self-escrowed public-key infrastructures. In JooSeok Song, editor, *ICICS* '99, volume 1787 of Lecture Notes in Computer Science, pages 257-268. Springer-Verlag, 1999.

13. Jacques Patarin and Louis Goubin. Trapdoor one-way permutations and multivariate polynomials. In *ICICS* '97, volume 1334 of Lecture Notes in Computer Science, pages 356-368. Springer-Verlag, 1997.

14. Guillaume Poupard and Jacques Stern. Fair encryption of RSA keys. In Bart Preneel, editor, *Eurocrypt* '00, volume 1807 of Lecture Notes in Computer Science, pages 172-189. Springer-Verlag, 2000.

15. Guillaume Poupard and Jacques Stern. Short proofs of knowledge for factoring. In *PKC* '00, volume 1751 of Lecture Notes in Computer Science, pages 147-166. Springer-Verlag, 2000.

16. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC* '01, volume 1992 of Lecture Notes in Computer Science, pages 119-136. Springer-Verlag, 2001.

17. Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In Bart Preneel, editor, *Eurocrypt* '00, volume 1807 of Lecture Notes in Computer Science, pages 152-165. Springer-Verlag, 2000.

18. Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Eurocrypt* '00, volume 1807 of Lecture Notes in Computer Science, pages 207-220. Springer-Verlag, 2000.

Reader's notes

About RSA Laboratories

An academic environment within a commercial organization, RSA Laboratories is the research center of RSA Security Inc., the company founded by the inventors of the RSA public-key cryptosystem. Through its research program, standards development, and educational activities, RSA Laboratories provides state-of-the-art expertise in cryptography and security technology for the benefit of RSA Security and its customers.

Please see www.rsasecurity.com/rsalabs for more information.

Newsletter Availability and Contact Information

CryptoBytes is a free publication and all issues, both current and previous, are available at www.rsasecurity.com/rsalabs/cryptobytes. While print copies may occasionally be distributed, publication is primarily electronic.

For more information, please contact:

cryptobytes-editor@rsasecurity.com.

RSA SECURITY®

RSA Security Inc.
www.rsasecurity.com

RSA Security Ireland Limited
www.rsasecurity.ie

©2002 RSA Security Inc. All rights reserved.
RSA and RSA Security are registered trademarks of RSA Security Inc. All other trademarks are the property of their respective owners.

CRYPTOBYTES VOLUME 5, NO. 1 0202