RSA

RSA Public-Key Cryptography

Sec In secret-key cryptography, the **sender's** **plaintext** (i.e., the original unencrypted text) is **encrypted** with a **secret key** to form **ciphertext** (i.e., the encrypted text). The **receiver** uses the ***same*** **secret key** to decode the **ciphertext**, forming the original **plaintext**. This is called **symmetric encryption**. A problem with secret-key cryptography is that the security of the ciphertext is only as good as the security of the secret key, and several copies of that key are "floating around." In an attempt to correct this problem, **public-key cryptography** was proposed by Diffie–Hellman.[1]

In this section, we walk step-by-step through the **RSA Public-Key Cryptography algorithm**.[2,3,4] In particular, we focus on how to generate:

- the **public key** that any sender can use to encrypt plaintext into ciphertext for a particular receiver, and
- the **private key** that only the particular receiver can use to decrypt the ciphertext.

RSA is based on sophisticated mathematics. However, the steps to generate the public and private keys, encrypt messages with the public key and decrypt messages with the private key are straightforward, as we'll show momentarily. Industrial-quality RSA works with enormous prime numbers consisting of hundreds of digits. To keep our explanations simple, we're going to use only small prime numbers in our explanations. Such small-prime-number RSA versions are not secure, but they'll help you understand how RSA works.

1. Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography." IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp. 644–654. November 1976. Accessed November 20, 2021. `https://ee.stanford.edu/~hellman/publications/24.pdf`.
2. "RSA (cryptosystem)." Accessed November 20, 2021. `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`.
3. "RSA Algorithm." Accessed November 20, 2021. `https://simple.wikipedia.org/wiki/RSA_algorithm`.
4. K. Moriarty, B. Kaliski, J. Jonsson and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2." November 2016. Accessed November 20, 2021. `https://tools.ietf.org/html/rfc8017`.

Public-Key Cryptography

Whitfield Diffie and Martin Hellman introduced **public-key cryptography**[5] to address secret-key cryptography's weakness—that the secret key must be known by both the sender and the receiver. They came up with the idea but not an implementation of the scheme.

RSA Public-Key Cryptography

Rivest, Shamir and Adelman were the first to publish a working implementation of public-key cryptography. The scheme, called RSA[6], bears the initials of their last names. RSA is the most widely implemented public-key cryptography scheme in the world.[7] Because RSA can be slow,[8] it's often used in conjunction with symmetric private-key encryption, using RSA only to securely send the secret key.[9,10]

Perf

Historical Notes

Clifford Cocks created a workable public-key scheme several years before the RSA paper was published.[11] His work was classified, so it was not revealed until about 20 years after RSA appeared. The company RSA Security held a patent on the RSA algorithm. In 2000, that patent was coming due for renewal. Rather than renewing it, they placed the algorithm into the public domain.[12]

RSA Algorithm Steps

For a nice video explanation of the RSA algorithm, see the following two-part video presentation:

- The RSA Encryption Algorithm (1 of 2: Computing an Example)[13]
- The RSA Encryption Algorithm (2 of 2: Generating the Keys)[14]

---

5. Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography." IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp. 644–654. November 1976. Accessed November 20, 2021. `https://ee.stanford.edu/~hellman/publications/24.pdf`.
6. R. Rivest; A. Shamir; L. Adleman (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" (PDF). Communications of the ACM. 21 (2): 120–126. Accessed November 20, 2021. `https://people.csail.mit.edu/rivest/Rsapaper.pdf`.
7. Michael Cobb, "RSA algorithm (Rivest-Shamir-Adleman)." Last updated in November 2021. Accessed November 20, 2021. `https://searchsecurity.techtarget.com/definition/RSA`.
8. "RSA (cryptosystem)." Accessed November 20, 2021. `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`.
9. Josh Lake, "What is RSA encryption and how does it work?" March 18, 2021. Accessed November 20, 2021. `https://www.comparitech.com/blog/information-security/rsa-encryption/`.
10. Ron Franklin, "AES vs. RSA Encryption: What Are the Differences?" March 13, 2021. Accessed November 20, 2021. `https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences`.
11. "Clifford Cocks." Accessed January 8, 2021. `https://en.wikipedia.org/wiki/Clifford_Cocks`.
12. "RSA Security Releases RSA Encryption Algorithm into Public Domain." September 6, 2000. Accessed November 20, 2021. `https://web.archive.org/web/20071120112201/http://www.rsa.com/press_release.aspx?id=261`.
13. Woo, Eddie (misterwootube). "The RSA Encryption Algorithm (1 of 2: Computing an Example)," November 4, 2014. `https://www.youtube.com/watch?v=4zahvcJ9glg`.
14. Woo, Eddie (misterwootube). "The RSA Encryption Algorithm (2 of 2: Generating the Keys)," November 4, 2014. `https://www.youtube.com/watch?v=oOcTVTpUsPQ`.

Steps 1–5 below use small integer values to explain how the RSA algorithm generates a public-key/private-key pair. Next, *Step 6* uses the public key to encrypt plaintext into ciphertext. Then, *Step 7* uses the private key to decrypt the ciphertext back to the original plaintext. The steps we show are based on the original RSA paper[15] and the RSA Algorithm Wikipedia page.[16]

RSA Algorithm Step 1—Choose Two Prime Numbers

Choose two different prime numbers, `p` and `q`. We'll use small prime numbers for this discussion—`p = 13` and `q = 17`. This will keep the calculations manageable in our discussions and on your computer using C++ with its limited-range, built-in integer data types. In commercial-grade RSA cryptography systems, these prime numbers typically are hundreds of digits each and chosen at random. For a sense of how large the integers in RSA can be, visit the RSA Numbers webpage

```
https://en.wikipedia.org/wiki/RSA_numbers
```

which shows integers from 100 to 617 digits in length. The C++ integer data types `int`, `long` and `long long` cannot hold integers this large. Libraries like Boost.Multiprecision[17] are required to accommodate arbitrary-sized integers.

RSA Algorithm Step 2—Calculate the Modulus (n), Which Is Part of Both the Public and Private Keys

Calculate the **modulus `n`**, which is simply the product of `p` and `q`:

```
n = p * q
```

Based on `p = 13` and `q = 17`, `n` is `221`. As you'll see, `n` is part of both the public and private keys. The `p` and `q` values are kept private.

RSA Algorithm Step 3—Calculate the Totient Function

Calculate $\Phi(n)$—pronounced "phi of `n`"—which is **Euler's totient function**.[18] This is calculated simply as:

```
Φ(n) = (p - 1) * (q - 1)
```

Given `p = 13` and `q = 17`, $\Phi(n)$ is

```
Φ(n) = 12 * 16 = 192
```

This number is used in the calculations that determine the **encryption exponent** (**`e`**) and **decryption exponent** (**`d`**), which will help us **encrypt the plaintext** and **decrypt the ciphertext**, respectively.

---

15. R. Rivest; A. Shamir; L. Adleman (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" (PDF). Communications of the ACM. 21 (2): 120–126. `https://people.csail.mit.edu/rivest/Rsapaper.pdf`.
16. "RSA Algorithm." Accessed January 6, 2021. `https://simple.wikipedia.org/wiki/RSA_algorithm`.
17. John Maddock and Christopher Kormanyos, "Boost.Multiprecision," Accessed November 20, 2021. `https://www.boost.org/doc/libs/1_72_0/libs/multiprecision/doc/html/index.html`.
18. "Euler's totient function." Accessed January 7, 2021. `https://en.wikipedia.org/wiki/Euler%27s_totient_function`.

RSA Algorithm Step 4—Select the Public-Key Exponent (e) for Encryption Calculations

Next, we **choose an exponent, e, for encryption**, which is subject to the following rules:

- `1 < e < Φ(n)`
- `e` must be **coprime** with `Φ(n)`.

Two integers are **coprime** if they have no common factors other than 1.

In our example, the integers that satisfy the first rule for `Φ(n) = 192` are the values `2`–`191`. The prime factorization of 192 is

```
192 = 2 * 2 * 2 * 2 * 2 * 2 * 3
```

The value for `e` must be coprime with `Φ(n)`, so we must eliminate from consideration for `e` any prime factors and all their multiples. Thus, the value `2` and the other even integers from `2`–`190` are eliminated, as are the value `3` and its multiples. This leaves the following odd values as possible values for `e`:

```
  5   7  11  13  17  19  23  25  29  31  35  37  41  43  47  49
 53  55  59  61  65  67  71  73  77  79  83  85  89  91  95  97
101 103 107 109 113 115 119 121 125 127 131 133 137 139 143 145
149 151 155 157 161 163 167 169 173 175 179 181 185 187 191
```

Any of these values can be used as the public encryption key's exponent (`e`). For our continuing discussion, we'll choose `37`, so **our public key is `(37, 221)`**.

RSA Algorithm Step 5—Select the Private-Key Exponent (d) for Encryption Calculations

The final step is to **determine the private key's exponent, `d`, for decryption**. We must choose a value for `d` such that

```
(d * e) mod Φ(n) = 1
```

In our example, the first value of `d` for which this is true is `109`. We can check whether the preceding calculation produces `1` by plugging in the values of `d`, `e` and `Φ(n)`:

```
(109 * 37) mod 192
```

The value of `109 * 37` is `4033`. If you multiply `192` by `21`, the result is `4032`, leaving a remainder of `1`. So, `109` is a valid value for `d`. There are many potential values of `d`—each is `109` plus a multiple of the totient (`192`). For instance, `301` (`109 + 1 * 192`):

```
(301 * 37) mod 192
```

`301 * 37` is `11137`, which has the remainder `1` when divided by `192`—`192 * 58` is `11136`, leaving a remainder of `1`. So values for `d` such as the following will work:

```
109  301  493  685  877  ...
```

We chose `109`, so our private key is `(109, 221)`.

RSA Algorithm Step 6—Encrypting a Message with RSA

Once you have the public key, it's easy to encrypt a message using RSA. **Given a plaintext integer message (`M`) to encrypt into ciphertext (`C`) and a public key consisting of two positive integers `e` (for encrypt) and `n`—commonly represented as `(e, n)`—a message sender can encrypt `M` with the calculation:**

$$C = M^e \bmod n$$

The value of `M` must be in the range `0 <= M < n`. Otherwise, you must break the message into values within that range and encrypt each separately.

Let's encrypt the `M` value `122` using our public key `(37, 221)`:

$$C = 122^{37} \bmod 221$$

The value 122^{37} is an enormous number, but you can perform this calculation using the Wolfram Alpha website at

```
https://www.wolframalpha.com/input/
```

Enter the calculation as follows (the ^ represents exponentiation in Wolfram Alpha):

```
122^37 mod 221
```

You'll see that the result is `5`, which is our ciphertext.

RSA Algorithm Step 7—Decrypting a Message with RSA

It's also easy to decrypt a message if you have the private key. **Given a ciphertext integer message (`C`) to decrypt into the original plaintext message (`M`) and a private key consisting of two positive integers `d` and `n`—commonly represented as `(d, n)`—a message receiver can decrypt `C` with the following calculation:**

$$M = C^{d} \bmod n$$

Let's decrypt the `C` value `5` using our public key `(109, 221)`:

$$C = 5^{109} \bmod 221$$

Once again, the value 5^{109} is an enormous number, but you can perform this calculation using Wolfram Alpha by entering the calculation as follows:

```
5^109 mod 221
```

You'll see that the result is `122`, which indeed is our plaintext.

Note that `n` is part of *both* the public key and the private key. You'll also see that the exponent `d`'s value is based on the exponent `e` and the modulus value `n`.

Encrypting and Decrypting Strings

Suppose you wish to use RSA to encrypt a plaintext message, such as

```
Damn the torpedoes, full speed ahead!
```[19]

As you know, the RSA algorithm encrypts *only* integer messages in the range `0 <= M < n`. To encrypt the preceding message, you must map the characters to integer values.

One way to convert characters to integers is to use each character's numeric value in the underlying character set. You might, for example, assume ASCII characters, which have integer values in the range 0–127. If a character's integer value is less than `n`, you can encrypt that value as shown previously. You can store each resulting ciphertext integer in an integer array. If you try to display those ciphertext integers as characters, you may see some strange symbols. For instance, the ciphertext integers might represent special characters, such as newlines or tabs, or might be outside the ASCII range.

19. David Glasgow Farragut—an American Civil War Union officer and the first full admiral in the U.S. Navy. Accessed January 8, 2021. `https://en.wikipedia.org/wiki/David_Farragut`.