RSA

Notes for *Serious Cryptography* chapter 10

Jeffrey Goldberg
jeffrey@goldmark.org

April 4, 2022
Last revised: May 8, 2024

From factoring problem to crypgoraphy

- Our goal is to turn the (believed) NP problem of factoring into useful cryptography.
- Despite all of the math we've gone through over the past months, there is still more to learn.
- We need to understand properties of modular arithmetic when the modulus is a composite number.

Composite moduli

Multiplication table

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Table 1: Multiplication modulo 15

Multiplication table

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Table 1: Multiplication modulo 15

1. There should be a column and a row for 0, but I left that for space.

Multiplication table (again)

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Table 2: Multiplication modulo 15: Highlighting some rows

Multiplication table (yet again)

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Table 3: Multiplication modulo 15: Highlighting other rows

Rows with 1

- All of the green rows contain 1.
- None of the yellow rows contain 1.

Example 1

- $2 \times 8 \equiv 1$
- There is no x such that $3 \times x \equiv 1$.

Rows with 0

- All of the yellow rows contain 0.
- None of the green rows contain 0.

Example 2

- $10 \times 3 \equiv 0$
- There is no $x > 0$ such that $7 \times x \equiv 0$.

Rows with all of the non-zero elements

- All of the green rows contain all of the numbers 1 through 14.
- None of the yellow rows contain all of the numbers 1 through 14.

Example 3

- The row for 4 contains all of the numbers 1 through 14.
- The row for 6 contains the numbers 3, 6, 9, 12, and 0; missing 1, 2, 4, 5, 7, 8, 10, 11, 13, and 14.

Coprime or not

- The green rows are for multiplying numbers that are coprime with 15: $\{1, 2, 4, 8, 11, 13, 14\}$
- The yellow rows are for multiplying numbers that are *not* coprime with 15: $\{3, 5, 6, 9, 10, 12\}$

Relatively prime

Definition 4 (Coprime)

Two integers a and b are **coprime** if and only if the largest integer that divides both of them is 1.

Other ways to express the same concept include "a and b are relatively prime" and "$\gcd(a, b) = 1$".

Relatively prime

Definition 4 (Coprime)

Two integers a and b are **coprime** if and only if the largest integer that divides both of them is 1.

Other ways to express the same concept include "a and b are relatively prime" and "$\gcd(a, b) = 1$".

1. When we are in the world of integers, we can use the word "divide" to mean integer division with no remainder.
2. We are in (a part of) the world of integers.

Examples for 15

Example 5

- 3 and 15 are *not* coprime because 3 divides both of them.
- 10 and 15 are *not* coprime because 5 divides both of them.
- 8 and 15 *are* coprime because 1 is the largest number that divides both of them.

Coprime with 15

n	$\gcd(n, 15)$	coprime with 15?
1	1	✓
2	1	✓
3	3	×
4	1	✓
5	5	×
6	3	×
7	1	✓
8	1	✓
9	3	×
10	5	×
11	1	✓
12	3	×
13	1	✓
14	1	✓

Table 4: Which numbers $0 < n < 15$ are coprime with 15

Composite moduli

Making a group

Green for group

If we remove the yellow rows (and corresponding columns) we get a group.

$\times$	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11
7	7	14	13	4	11	2	1	8
8	8	1	2	11	4	13	14	7
11	11	7	14	2	13	1	8	4
13	13	11	7	1	14	8	4	2
14	14	13	11	8	7	4	2	1

Table 5: Operation table for $\mathbb{Z}_{15}^{\times}$

The group $\mathbb{Z}_{15}^{\times}$

Definition 6 ($\mathbb{Z}_{15}^{\times}$)

The group $\mathbb{Z}_{15}^{\times}$ is the set of elements, $\{1, 2, 4, 7, 8, 11, 13, 14\}$, along with the operation multiplication modulo 15.

Definition 6 ($\mathbb{Z}_{15}^{\times}$)
The group $\mathbb{Z}_{15}^{\times}$ is the set of elements, $\{1, 2, 4, 7, 8, 11, 13, 14\}$, along with the operation multiplication modulo 15.

1. Note that the members of the group do *not* include 0, 3, 5, 6, 9, 10, and 12.

$\mathbb{Z}_{15}^{\times}$ AND ITS GROUP PROPERTIES

$\mathbb{Z}_{15}^{\times}$ is an abelian (commutative) group because for all elements of the group a, b and c

- It is **closed**. $a \times b$ is also in the group.
- There is an **identity element**, which in this case is 1, such that $a \times 1 \equiv a$.
- Every element has an **inverse**: For every a there is a member of the group that we will call 'a^{-1}' such that $a \times a^{-1} \equiv 1$.
- The operation is commutative, making the group **abelian**: $a \times b \equiv b \times a$
- The operation is **associative**: $(a \times b) \times c \equiv a \times (b \times c)$.

$\mathbb{Z}_{15}^{\times}$ AND ITS GROUP PROPERTIES

$\mathbb{Z}_{15}^{\times}$ is an abelian (commutative) group because for all elements of the group a, b and c

- It is **closed**. $a \times b$ is also in the group.
- There is an **identity element**, which in this case is 1, such that $a \times 1 \equiv a$.
- Every element has an **inverse**: For every a there is a member of the group that we will call 'a^{-1}' such that $a \times a^{-1} \equiv 1$.
- The operation is commutative, making the group **abelian**: $a \times b \equiv b \times a$
- The operation is **associative**: $(a \times b) \times c \equiv a \times (b \times c)$.

1. By now, many of you should be able to recite the properties of a group in your sleep. And perhaps I have put you to sleep. But we also have newcomers.

Totients

How many coprimes?

- There are eight numbers less than 15 that are coprime with 15.
- This is typically written "$\phi(15) = 8$', using the small Greek letter 'ϕ' (phi) or the variant form 'φ'.
- $\phi(n)$ is called the "totient of n".

Totient

Definition 7 (Totient)

The totient of a number n, written $\phi(n)$, is the number of numbers which are coprime with n and less than n.

Totient

Definition 7 (Totient)

The totient of a number n, written $\phi(n)$, is the number of numbers which are coprime with n and less than n.

1. The term “totient” was introduced in 1879 with no explanation by Sylvester [Syl79]. Seems to be some sort of total on analogy to “quotient.”

Totient of 15

- $\phi(15) = 8$
- $15 = 3 \times 5$
- $\phi(15) = (3-1)(5-1)$
- That is not a coincidence, and we will return to it.

Fermat's Little Theorem

Little vs Last

- Fermat's *Last* Theorem is famous because he never proved it.
- Fermat never proved the "little" theorem either.
- Fermat's Little Theorem was extended and proved by Euler.
- Unlike Fermat's Last Theorem, Fermat's *Little* Theorem is enormously useful.

Little vs Last

- Fermat's *Last* Theorem is famous because he never proved it.
- Fermat never proved the "little" theorem either.
- Fermat's Little Theorem was extended and proved by Euler.
- Unlike Fermat's Last Theorem, Fermat's *Little* Theorem is enormously useful.

1. The key to proving Fermat's *Last* Theorem in 1994 rested on an important theorem about elliptic curves over rational fields.

Fermat's Little Theorem

Theorem 8 (Fermat's Little Theorem)

If p is prime and a is not a multiple of p then

$$a^{p-1} \equiv 1 \pmod{p}$$

FLT WITH $p - 1 = 6$

a	1	2	3	4	5	6
a^{7-1}	1	64	729	4096	15 625	46 656
$a^{7-1} \pmod 7$	1	1	1	1	1	1

Table 6: Illustrating FTL with exponent $7 - 1$

Multiply by a

$$a^{p-1} \equiv 1 \pmod{p} \qquad \text{(Theorem 8)}$$

$$a^{p-1} \cdot a \equiv 1 \cdot a \pmod{p} \qquad \text{(Multiply each side by } a\text{)}$$

$$a^{p} \equiv a \pmod{p} \qquad \text{(Exponentiation rules!)}$$

Also Fermat's Little Theorem

Theorem 9 (Also Fermat's Little Theorem)

If p is prime and a is not a multiple of p then

$$a^p \equiv a \pmod{p}$$

FLT WITH $p = 7$

a	1	2	3	4	5	6
a^7	1	128	2187	16 384	78 125	279 936
$a^7 \pmod 7$	1	2	3	4	5	6

Table 7: Illustrating Fermat's Little Theorem with $p = 7$

Historical note

- Chinese mathematicians were long aware of this, perhaps as far back as 500 BCE.
- Fermat rediscovered it and said to have a proof in 1640.
- Leibniz proved it before 1683 in an unpublished manuscript.
- Euler published a proof in 1740.
- Euler proved an important generalization of FLT in the same year.

Historical note

- Chinese mathematicians were long aware of this, perhaps as far back as 500 BCE.
- Fermat rediscovered it and said to have a proof in 1640.
- Leibniz proved it before 1683 in an unpublished manuscript.
- Euler published a proof in 1740.
- Euler proved an important generalization of FLT in the same year.

1. There was some destruction of books (and killing of scholars) during the Qin dynasty, but modern historians do not think it was anywhere as bad as Han dynasty reports made it out to be.
2. Not publishing a proof was common. It was a time of weird rivelries, a penchent for keeping discoveries secret, the fact that Fermet was an amateur who wrote things in letters to his friends, and there really wasn't a system for publishing proofs at the time.
3. There is a great deal of historical scholarship on what Leibniz knew and when.
4. It is Euler's generalation that we go to next.

Fermat's Little Theorem

Euler's Totient Theorem

Euler's extension

Theorem 10 (Euler's Totient Theorem)

If a and n are coprime

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Multiplying Euler's Theorem

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

$$a^{\phi(n)} \cdot a \equiv 1 \cdot a \pmod{n} \qquad \text{(Multiply both sides by } a\text{)}$$

$$a^{\phi(n)+1} \equiv a \pmod{n} \qquad \text{(Exponentiation tricks)}$$

Also Euler's Theorem

Theorem 11 (Also Euler's Totient Theorem)

If a and n are coprime

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

Computing $\phi(n)$

Definition 12 (Totient formula for $n = pq$; $p \neq q$)

When $n = pq$ where p and q are distinct primes, $\phi(n) = (p-1)(q-1)$.

Definition 12 (Totient formula for $n = pq$; $p \neq q$)
When $n = pq$ where p and q are distinct primes, $\phi(n) = (p-1)(q-1)$.

1. Euler provided a much more general formula for computing totients, but this simple case is all we need. In all cases, one needs the prime factors of n to compute $\phi(n)$.

Euler's Theorem modulo 15

a	1	2	4	7	8
$a^{\phi(15)}$	1	256	65 536	5 764 801	16 777 216
$a^{\phi(15)} \pmod{15}$	1	1	1	1	1

Table 8: Illustrating Euler's theorem with modulus 15. $\phi(15) = 8$

RSA

RSA public key

- A prime public exponent, e, which is not unique to the user and is typically 65537.
- A public modulus, N which is unique to the user and is the product of two primes.

RSA public key

- A prime public exponent, e, which is not unique to the user and is typically 65537.
- A public modulus, N which is unique to the user and is the product of two primes.

1. 65537 is 0b1000000000000001. That is a lot of 0s and only two 1s. This means that the square-and-multiply algorithm only has to do two multiplies (and 16 squarings).
2. The square-and-multipy algorithm for exponentiation is analogous to the double-and-add algorithm we went over when covering ECHD.
3. Because e is a public modulus, we do not need to worry about side channel attacks revealing it, and so using square-and-multiply is fine.
4. The standards allow for public moduli to be constructed as the products of more than two primes. There were reasonable reasons at the time, but I don't think that any system has actually made use of that flexibility.

RSA private key

- Everything that is in the public key
- The primes p and q such that $N = pq$
- A decryption exponent, d, computed from p, q and e.

RSA private key

- Everything that is in the public key
- The primes p and q such that $N = pq$
- A decryption exponent, d, computed from p, q and e.

1. There are other things that are precomputed that are helpful for performing computations, we need to focus on d here.

The decryption exponent

Definition 13 (The decryption secret)

The decryption exponent, d, is computed from the public exponent e and the factors, p and q, of public modulus, N.

$$d = e^{-1} \bmod \phi(N)$$

Definition 13 (The decryption secret)

The decryption exponent, d, is computed from the public exponent e and the factors, p and q, of public modulus, N.

$$d = e^{-1} \bmod \phi(N)$$

1. There is an efficient algorithm for computing the modular multiplicative inverse. And there are even quicker ones when you know the prime factors of the modulus as we do.

What does $a \equiv 1 \pmod{M}$ really mean?

"$a \equiv 1 \pmod{M}$" means (among other things)

1. If we divide a by M we get a remainder of 1;
2. a is one more than some multiple of M;
3. There exists some integer k such that $a = kM + 1$.

That last notion, $a = kM + 1$, is going to useful in what follows.

What does $a \equiv 1 \pmod{M}$ really mean?

WHAT DOES $a \equiv 1 \pmod{M}$ REALLY MEAN?

"$a \equiv 1 \pmod{M}$" means (among other things)

1. If we divide a by M we get a remainder of 1;
2. a is one more than some multiple of M;
3. There exists some integer k such that $a = kM + 1$.

That last notion, $a = kM + 1$, is going to useful in what follows.

1. Early on, I made a choice to use '$=$' in all the places it should be used and also where '$\equiv$' should be used. As I wasn't defining modular operations formally and pedantically, I thought I could get away with it. I am now paying the price, as the distinction between the two would be really useful here, but it would mean that I would have to use them correctly and consistently throughout.

This will be useful later

A useful fact follows from how we defined d:

$$\begin{aligned} ed &\equiv 1 \pmod{\phi(N)} \\ ed &= k\phi(N) + 1 \qquad \text{(For some integer } k\text{)} \end{aligned} \tag{1}$$

Encryption and Decryption

$$c = m^e \bmod N \tag{2}$$

$$m = c^d \bmod N \tag{3}$$

Why does that work?

$$
\begin{aligned}
c^d &\equiv \left(m^e\right)^d && \text{(From } c \equiv m^e\text{)} \\
&\equiv m^{ed} && \text{(From how exponents work)} \\
&\equiv m^{k\phi(N)+1} && \text{(From Eq. 1 “a useful fact”)} \\
&\equiv m && \text{(A miracle occurs)}
\end{aligned} \tag{4}
$$

1. The miracle involves Euler's Theorem.

Why does that work?

$$
\begin{aligned}
c^d &\equiv (m^e)^d && \text{(From } c \equiv m^e\text{)} \\
&\equiv m^{ed} && \text{(From how exponents work)} \\
&\equiv m^{k\phi(N)+1} && \text{(From Eq. 1 "a useful fact")} \\
&\equiv m && \text{(A miracle occurs)}
\end{aligned} \tag{4}
$$

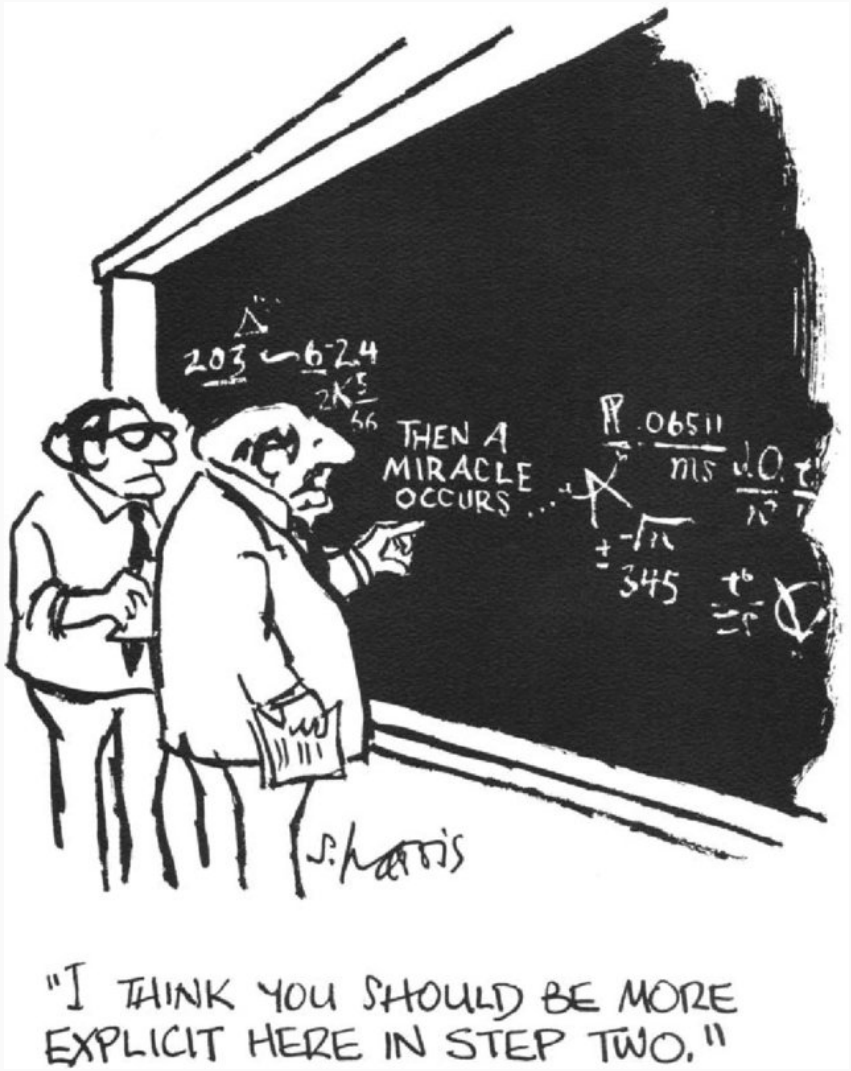

Figure 1: "Then a miracle occurs" Comic by Sidney Harris. Used by permission of ScienceCartons

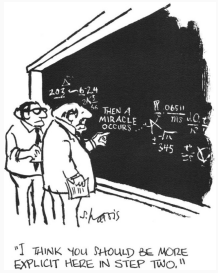

Figure 1: "Then a miracle occurs" Comic by Sidney Harris. Used by permission of ScienceCartons

1. Time for me to go into Certify to get reimbursed for the licensing.

The miracle

$$
\begin{aligned}
c^d &\equiv m^{k\phi(N)+1} && \text{(Our pre-miracle equation (4))} \\
&\equiv m^{k\phi(N)} \cdot m^1 && \text{(From how exponents work)} \\
&\equiv \left(m^{\phi(N)}\right)^k \cdot m^1 && \text{(Also from how exponents work)} \\
&\equiv 1^k \cdot m^1 && \text{(From Euler's Theorem)} \\
&\equiv 1 \cdot m && \text{(From how 1 works)} \\
&\equiv m && \text{(Also from how 1 works)}
\end{aligned}
$$

The miracle

$$
\begin{aligned}
c^d &\equiv m^{k\phi(N)+1} && \text{(Our pre-miracle equation (4))} \\
&\equiv m^{k\phi(N)} \cdot m^1 && \text{(From how exponents work)} \\
&\equiv \left(m^{\phi(N)}\right)^k \cdot m^1 && \text{(Also from how exponents work)} \\
&\equiv 1^k \cdot m^1 && \text{(From Euler's Theorem)} \\
&\equiv 1 \cdot m && \text{(From how 1 works)} \\
&\equiv m && \text{(Also from how 1 works)}
\end{aligned}
$$

1. This slide is for additional material For those who want to see why I can ignore the k
2. I hope that you all see why I did Diffie-Hellman first. It gave everyone practice with modular exponentiation in smaller bites.

How strong is RSA?

How strong is RSA?

Key recovery

Key recovery is as hard as factoring

1. There is a polynomial time algorithm for computing d from the factors of N.
2. Therefore key recovery is not harder than factoring.
3. There is a polynomial time algorithm for computing the factors of N from d.
4. Therefore factoring is not harder than key recovery.
5. From 2 and 4 we know that RSA key recovery and factoring are equally hard.

Key recovery is as hard as factoring

1. There is a polynomial time algorithm for computing d from the factors of N.
2. Therefore key recovery is not harder than factoring.
3. There is a polynomial time algorithm for computing the factors of N from d.
4. Therefore factoring is not harder than key recovery.
5. From 2 and 4 we know that RSA key recovery and factoring are equally hard.

1. It is simply how we computed d in the first place.
2. Another way to say X is not harder than Y is to say Y is at least as hard as X. So factoring is at least as hard as key recovery.
3. Computing the factors of N from d and e involves a messy algorithm. It would be tedious to describe and show why it works. But it exists and is $\mathbf{poly}(|N|)$.

Does this mean that breaking RSA is as hard as factoring?

Q: Does this mean that breaking RSA is as hard as factoring?

A: No! Key recovery is not the only way to break a cipher scheme.

How strong is RSA?

Security notions

Notions again

- We don't know all the ways to attack an encryption scheme.
- We can only list the ways we know of to attack an encryption scheme.
- But we can define what it means for a scheme to be broken irrespective of how it is broken.
- An encryption scheme is broken if having the ciphertext reveals new information about the content of the plaintext.
- In practice these definitions are in terms of adversaries playing well-defined games against the scheme.

Notions again

- We don't know all the ways to attack an encryption scheme.
- We can only list the ways we know of to attack an encryption scheme.
- But we can define what it means for a scheme to be broken irrespective of how it is broken.
- An encryption scheme is broken if having the ciphertext reveals new information about the content of the plaintext.
- In practice these definitions are in terms of adversaries playing well-defined games against the scheme.

1. We went through notions and IND-CPA more than half a year ago, so this is a brief recap.
2. It is very important to demostrate that a scheme is secure against such attacks.
3. "New information" is defined in a way that makes traffic analysis somebody else's problem.
4. The proof that the IND-CPA game definition and the "ciphertext provides no new information" definitions are the same is left as an excercie to the reader.
5. Or you could take my word for it that there is such a proof.

IND-CPA game: The RSA edition

The adversary, $\mathcal{A}$, plays a game against the challenger that goes like this

1. During setup, the challenger generates a random RSA key pair. The public key, **pk**, is given to $\mathcal{A}$.
2. The challenger flips a fair coin to set a bit, b, to either 0 or 1. b is not given to $\mathcal{A}$.
3. $\mathcal{A}$ creates two plaintexts of its choosing, m_0 and m_1.
4. The challenger encrypts m_0 or m_1 depending on the random bit b and returns the ciphertext c_b to $\mathcal{A}$.
5. $\mathcal{A}$ studies what it has $\{\textbf{pk}, m_0, m_1, c_b\}$ to decide which of m_0 or m_1 was encrypted.
6. If $\mathcal{A}$'s conclusion is correct $\mathcal{A}$ wins.

IND-CPA game: The RSA edition

The adversary, $\mathcal{A}$, plays a game against the challenger that goes like this

1. During setup, the challenger generates a random RSA key pair. The public key, **pk**, is given to $\mathcal{A}$.
2. The challenger flips a fair coin to set a bit, b, to either 0 or 1. b is not given to $\mathcal{A}$.
3. $\mathcal{A}$ creates two plaintexts of its choosing, m_0 and m_1.
4. The challenger encrypts m_0 or m_1 depending on the random bit b and returns the ciphertext c_b to $\mathcal{A}$.
5. $\mathcal{A}$ studies what it has $\{\mathbf{pk}, m_0, m_1, c_b\}$ to decide which of m_0 or m_1 was encrypted.
6. If $\mathcal{A}$'s conclusion is correct $\mathcal{A}$ wins.

1. This game captures the notion that the ciphertext does not give an attacker any new information about the content of the plaintext.

RSA Assumption

Definition 14 (RSA Assumption)

No polynomial time algorithm can win the IND-CPA RSA game (meaningfully) better than half the time.

Proving in the right direction

Tired If you can break problem P, you can break my cipher scheme. Therefore my cipher scheme is as hard as P

Wired If you can break my cipher scheme, you can break problem P. Therefore my scheme is at least as hard as P.

Is breaking RSA as hard as factoring?

Q: Is breaking RSA as hard as factoring?

A1: For “textbook RSA” described so far, the answer is no.

A2: For non-deterministic RSA the answer is that it is complicated.

Q: Is breaking RSA as hard as factoring?

A1: For "textbook RSA" described so far, the answer is no.

A2: For non-deterministic RSA the answer is that it is complicated.

1. The answer is probably "no" for small public exponent e. For suffiently large e the answer is "we hope so."

How strong is RSA?

Non-deterministic encryption

Encryption must be randomized

As described so far, $\mathcal{A}$ can win the $\mathrm{IND\text{-}CPA}$ game 100% of the time. $\mathcal{A}$ can simply encrypt m_0 and m_1 with the public key to generate c_0 and c_1, one of which will be equal to the challenge ciphertext c_b.

OAEP

- Optimal Asymmetric Encryption Padding (OAEP)
- Does the job analogous to the padding used in block ciphers
- Does the job analogous to the nonce/IV used in block ciphers
- It does some hashing with the randomness to make sure that the bit sequence (other than first two bits) that goes through actual RSA has no usable structure.

Avoid PKCS1 – Use OAEP

- In addition to OAEP, there is another RSA padding scheme, PKCS1.
- PKCS1 does almost everything right, but OAEP is superior in every respect.
- Many cryptographic libraries offer both PKCS1 and OAEP modes.
- Do not use PKCS1 unless you have to.
- If you must use PKCS1, be aware that you can sometimes get a “successful” decryption of mangled data.

Avoid PKCS1 – Use OAEP

- In addition to OAEP, there is another RSA padding scheme, PKCS1.
- PKCS1 does almost everything right, but OAEP is superior in every respect.
- Many cryptographic libraries offer both PKCS1 and OAEP modes.
- Do not use PKCS1 unless you have to.
- If you must use PKCS1, be aware that you can sometimes get a “successful” decryption of mangled data.

1. crypto libraries really should do more to discourage PKCS1. One way would be to make it unavailable when creating new messages.

How strong is RSA?

What we know about RSA

Factoring assumption

Definition 15 (Factoring assumption)

There is no **poly**$(|x|)$ algorithm which can factor an appropriately chosen x.

What a proof would look like

1. Imagine that a polynomial time algorithm, $\mathcal{A}$, has access to an oracle that has broken RSA encryption.
2. We don't have any knowledge of the internal workings of the oracle, but we can feed it ciphertexts and get out plaintexts.
3. Can $\mathcal{A}$ use its freedom to get decryptions from the oracle to factor the public key?
4. If yes, then the RSA problem is at least as hard as factoring.

So far no such proof exists.

What a proof would look like

1. Imagine that a polynomial time algorithm, $\mathcal{A}$, has access to an oracle that has broken RSA encryption.
2. We don't have any knowledge of the internal workings of the oracle, but we can feed it ciphertexts and get out plaintexts.
3. Can $\mathcal{A}$ use its freedom to get decryptions from the oracle to factor the public key?
4. If yes, then the RSA problem is at least as hard as factoring.

So far no such proof exists.

1. Anyone else thinking of Anselm of Cambridge's ontological argument? Well what we are doing here really is solid and works.
2. It is unlikely that such a proof exists for the RSA assumption as it stands. There are special cases, such as for certain values of e, where a polynomial algorithm does exist. So the RSA assumption will need refining.

Algorithms

Greatest Common Divisor

```
def gcd(a: int, b: int) -> int:
  while a != 0:
      a, b = b % a, a
  return b
```

Figure 2: Euclid's Algorithm for greatest common divisor

Greatest Common Divisor

```
def gcd(a: int, b: int) -> int:
  while a != 0:
      a, b = b % a, a
  return b
```

Figure 2: Euclid's Algorithm for greatest common divisor

1. The document from the Cat in the Middle session provides a lot of detail about why this works and varient forms of it.

Extended Euclidean Algorithm

```python
def egcd(a: int, b: int) -> tuple[int, int, int]:
    x0, x1, y0, y1 = 0, 1, 1, 0
    while a != 0:
        (q, a), b = divmod(b, a), a
        y0, y1 = y1, y0 - q * y1
        x0, x1 = x1, x0 - q * x1
    return b, x0, y0
```

Figure 3: Extended Euclidian Algorithm (EEA): Returns g, x, and y such that $ax + by = g = \gcd(a, b)$

Extended Euclidean Algorithm

```python
def egcd(a: int, b: int) -> tuple[int, int, int]:
    x0, x1, y0, y1 = 0, 1, 1, 0
    while a != 0:
        (q, a), b = divmod(b, a), a
        y0, y1 = y1, y0 - q * y1
        x0, x1 = x1, x0 - q * x1
    return b, x0, y0
```

Figure 3: Extended Euclidian Algorithm (EEA): Returns g, x, and y such that $ax + by = g = \gcd(a, b)$

1. It is specularly tedious to work through why the EEA works or even what it is doing. The very rough idea is that it is keeping a running record of intermediate quotiants for current and previous run throughs, so that it can do a little bit of backtracking at the end.

Modular inverse

```python
def modinv(a:int, m:int) -> int|None:
    g, x, _ = egcd(a, m)
    if g != 1:
        return None
    return x % m
```

Figure 4: General case for computing modular inverse using the Extended Euclidean Algorithm. Returns `None` if modular inverse doesn't exist.

Euler's totient theorem again

In a mod m world

$$aa^{-1} \equiv 1 \qquad \text{(Definition of } a^{-1}\text{)}$$

$$a^{\phi(m)} \equiv 1 \qquad \text{(Euler's theorem)}$$

$$aa^{\phi(m)-1} \equiv 1 \qquad \text{(How exponents work)}$$

$$aa^{\phi(m)-1} \equiv aa^{-1} \qquad (a \text{ times its inverse is 1})$$

$$a^{\phi(m)-1} \equiv a^{-1} \qquad \text{(Multiply by } a^{-1} \text{ (divide by } a\text{))}$$

Euler's totient theorem again

In a mod m world

$aa^{-1} \equiv 1$	(Definition of a^{-1})
$a^{\phi(m)} \equiv 1$	(Euler's theorem)
$aa^{\phi(m)-1} \equiv 1$	(How exponents work)
$aa^{\phi(m)-1} \equiv aa^{-1}$	(a times its inverse is 1)
$a^{\phi(m)-1} \equiv a^{-1}$	(Multiply by a^{-1} (divide by a))

1. The intuition (at least for me) is that if $\phi(m)$ number of as multiplied together take you around to 1 then $\phi(m) - 1$ number of as take you to the thing you could multiply by one more a to get to 1. That thing that you multiply "one more a to to get 1 is the inverse of a.
2. On "how exponents work": $a^x = aa^{x-1}$.

Modular inverse with $\phi(m)$

```python
def modinv_tot(a:int, m:int, tot: int) -> int:
    return pow(a, tot - 1, m)
```

Figure 5: Modular inverse using $\phi(m)$. Produces erroneous results (without warning) if $\gcd(a, m) \neq 1$ or if an incorrect totient is provided.

Resources

Resources

- These slides
- Exceedingly extensive notes on the Cat in the Middle computation. Those notes include explanations of some of the algorithms discussed.
- Sources

References I

[Aum17] Jean-Philippe Aumasson. ***Serious cryptography: a practical introduction to modern encryption.*** No Starch Press, 2017.

[Syl79] J. J. Sylvester. **"On Certain Ternary Cubic-Form Equations."** In: *American Journal of Mathematics* 2.4 (1879), pp. 357–393. URL: http://www.jstor.org/stable/2369490.