A Tour of Classical and Modern Cryptography

Evan P. Dummit

University of Rochester

February 11, 2015

What is Cryptography?

Cryptography is the study of designing

- ways to send secure messages across an insecure network, or
- ways to send messages to your friends without your enemies reading them, or
- ways to protect sensitive data from attackers.

What is Cryptography?

Cryptography is the study of designing

- ways to send secure messages across an insecure network, or
- ways to send messages to your friends without your enemies reading them, or
- ways to protect sensitive data from attackers.

Cryptography is generally a blend of three things:

- Mathematics (number theory, linear algebra, combinatorics)
- Computer science and computer engineering
- Cleverness

Goals of Cryptography, I

Cryptography uses standard names:

- Alice has "plaintext" that she wants to encrypt to make "ciphertext".
- Bob receives encrypted ciphertexts from Alice that he wants to decrypt (he may also send messages back).
- Eve is an eavesdropper: she spies on Alice and Bob.

Goals of Cryptography, I

Cryptography uses standard names:

- Alice has "plaintext" that she wants to encrypt to make "ciphertext".
- Bob receives encrypted ciphertexts from Alice that he wants to decrypt (he may also send messages back).
- Eve is an eavesdropper: she spies on Alice and Bob.

Important Goal of Cryptography

Alice and Bob want to communicate without Eve being able to decode their messages.

Goals of Cryptography, II

Eve might also try to alter messages, or impersonate Alice or Bob.

Goals of Cryptography, II

Eve might also try to alter messages, or impersonate Alice or Bob.

Important Goal of Cryptography

Bob wants to verify all the messages he receives have not been altered since Alice sent them.

Goals of Cryptography, II

Eve might also try to alter messages, or impersonate Alice or Bob.

Important Goal of Cryptography

Bob wants to verify all the messages he receives have not been altered since Alice sent them.

Important Goal of Cryptography

Bob wants to verify all the messages he receives are really from Alice, and not Eve pretending to be Alice.

Caesar Shift, I

The “Caesar shift” (supposedly used by Julius Caesar):

Caesar Shift, I

The “Caesar shift” (supposedly used by Julius Caesar):

- Choose a number from 1 to 25.
- To encode, shift plaintext alphabetically forward by that many letters (wrap from Z to A).
- To decode, shift back by the same amount.

Example

Shift by 3 letters: **yellowjackets** $\rightarrow$

Caesar Shift, I

The "Caesar shift" (supposedly used by Julius Caesar):

- Choose a number from 1 to 25.
- To encode, shift plaintext alphabetically forward by that many letters (wrap from Z to A).
- To decode, shift back by the same amount.

Example

Shift by 3 letters: **yellowjackets** → **BHOORZMDFNHWV**.

Caesar Shift, II

The Caesar shift has a few obvious flaws:

Caesar Shift, II

The Caesar shift has a few obvious flaws:

- There are only 25 possible encodings!
- If Eve knows even a single letter of the original message (and its encoding) she can decode the rest immediately.

Caesar Shift, II

The Caesar shift has a few obvious flaws:

- There are only 25 possible encodings!
- If Eve knows even a single letter of the original message (and its encoding) she can decode the rest immediately.

Want to avoid both of these kinds of attacks.

Ultimate issue: Caesar shift is too predictable.

Substitution, I

Reinterpretation of Caesar shift: it replaces each letter of plaintext with another one in some fixed manner. Can generalize with a general "letter substitution":

- To each plaintext letter **a**-**z**, assign a ciphertext letter (using each letter exactly once).
- For example: $\mathbf{a} \mapsto \mathbf{M}$, $\mathbf{b} \mapsto \mathbf{D}$, ... , $\mathbf{z} \mapsto \mathbf{W}$.
- To decode, simply make the replacements in reverse.

Substitution, II

Can summarize substitution by listing "encoding alphabet":

```
abcdefghijklmnopqrstuvwxyz
TFOANYRLEMZSGKPXQCDIHBUJVW
```

Substitution, II

Can summarize substitution by listing “encoding alphabet”:

```
abcdefghijklmnopqrstuvwxyz
TFOANYRLEMZSGKPXQCDIHBUJVW
```

Example

With the encoding alphabet above,

```
substitution is slightly better than caesar shifting
                           ↓
DHFDIEIHIEPK ED DSERLISV FNIINC ILTK OTNDTC DLEYIEKR
```

Substitution *is* slightly better than Caesar shifting

A bit less obvious how to crack substitution ciphers.

- If given a sample of plaintext + ciphertext, fairly easy to decode remainder (get most of the encoding alphabet).
- But what if just ciphertext is given?
- Brute force has too many possibilities: $26! \approx 4.03 \cdot 10^{26}$.

Substitution *is* slightly better than Caesar shifting

A bit less obvious how to crack substitution ciphers.

- If given a sample of plaintext + ciphertext, fairly easy to decode remainder (get most of the encoding alphabet).
- But what if just ciphertext is given?
- Brute force has too many possibilities: $26! \approx 4.03 \cdot 10^{26}$.

Key idea is "frequency analysis". For English:

Letter	e	t	a	o	i	n	s
Frequency	12.7%	9.1%	8.2%	7.5%	7.0%	6.7%	6.3%
Letter	h	r	d	l	c	u	m
Frequency	6.1%	6.0%	4.3%	4.0%	2.8%	2.8%	2.4%

Substitution, IV: A New Hope

Can break substitution ciphers by counting letter frequencies.

- Most likely letter is probably **e**...

Substitution, IV: A New Hope

Can break substitution ciphers by counting letter frequencies.

- Most likely letter is probably **e**... though might be **t**, **a**, or **o**.
- Can definitely cut down from 26! possibilities, though.

Substitution, IV: A New Hope

Can break substitution ciphers by counting letter frequencies.

- Most likely letter is probably **e**... though might be **t**, **a**, or **o**.
- Can definitely cut down from 26! possibilities, though.

Get extra mileage using frequencies of 2-letter combinations ("digrams") and 3-letter combinations ("trigrams"):

- Digrams: **th**, **he**, **in**, **an**, **er**, **re**, **ed**, **on**, **es**, **ea**, **ti**, **st**, **en**, **at**.
- Trigrams: **the**, **ing**, **and**, **ere**, **her**

Substitution, V

Let's do an example:

KB TI BQ NBK KB TI KRZK PF KRI XAIFKPBN DRIKRIQ KPF NBTHIQ PN KRI YPNC KB FAWWIQ KRI FHPNOF ZNC ZQQBDF BW BAKQZOIBAF WBQKANI BQ KB KZGI ZQYF ZOZPNFK Z FIZ BW KQBATHIF ZNC TJ BEEBFPNO INC KRIY

Substitution, V

Let's do an example:

```
KB TI BQ NBK KB TI KRZK PF KRI XAIFKPBN DRIKRIQ KPF
NBTHIQ PN KRI YPNC KB FAWWIQ KRI FHPNOF ZNC ZQQBDF BW
BAKQZOIBAF WBQKANI BQ KB KZGI ZQYF ZOZPNFK Z FIZ BW
KQBATHIF ZNC TJ BEEBFPNO INC KRIY
```

Quick count: 19 K, 18 B, 17 I, 13 F, ...
Also, 6 each of KR, RI, ..., and 5 KRI.

Substitution, V

Try KRI = the:

```
tB Te BQ NBt tB Te thZt PF the XAeFtPBN DhetheQ tPF
NBTHeQ PN the YPNC tB FAWWeQ the FHPNOF ZNC ZQQBDF BW
BAtQZOeBAF WBQtANe BQ tB tZGe ZQYF ZOZPNFt Z FeZ BW
tQBATHeF ZNC TJ BEEBFPNO eNC theY
```

Substitution, V

Try KRI = the:

tB Te BQ NBt tB Te thZt PF the XAeFtPBN DhetheQ tPF NBTHeQ PN the YPNC tB FAWWeQ the FHPNOF ZNC ZQQBDF BW BAtQZOeBAF WBQtANe BQ tB tZGe ZQYF ZOZPNFt Z FeZ BW tQBATHeF ZNC TJ BEEBFPNO eNC theY

Now make educated guesses: how about B = o and Z = a?

Substitution, V

Let's take a look now:

```
to Te oQ Not to Te that PF the XAeFtPoN DhetheQ tPF
NoTHeQ PN the YPNC to FAWWeQ the FHPNOF aNC aQQoDF oW
oAtQaOeoAF WoQtANe oQ to taGe aQYF aOaPNFt a Fea oW
tQoATHeF aNC TJ oEEoFPNO eNC theY
```

Substitution, V

Let's take a look now:

`to Te oQ Not to Te that PF the XAeFtPoN DhetheQ tPF NoTHeQ PN the YPNC to FAWWeQ the FHPNOF aNC aQQoDF oW oAtQaOeoAF WoQtANe oQ to taGe aQYF aOaPNFt a Fea oW tQoATHeF aNC TJ oEEoFPNO eNC theY`

Now let's try $D = w$, $Q = r$, $F = s$, $P = i$:

Substitution, V

Let's take a look again:

```
to Te or Not to Te that is the XAestioN whether tis
NoTHer iN the YiNC to sAWWer the sHiNOs aNC arrows oW
oAtraOeoAs WortANe or to taGe arYs aOaiNst a sea oW
troATHes aNC TJ oEEosiNO eNC theY
```

Substitution, V

Let's take a look again:

```
to Te or Not to Te that is the XAestioN whether tis
NoTHer iN the YiNC to sAWWer the sHiNOs aNC arrows oW
oAtraOeoAs WortANe or to taGe arYs aOaiNst a sea oW
troATHes aNC TJ oEEosiNO eNC theY
```

Clearly, T = b, N = n, X = q, A = u, H = l, Y = m, C = d, …

Substitution, Victory

It's starting to come together now:

`to be or not to be that is the question whether tis nobler in the mind to suffer the slings and arrows of outrageous fortune or to take arms against a sea of troubles and by opposing end them`

Polyalphabetic Ciphers

Ultimate weakness of substitution cipher: each letter is encoded the same way every time it appears.

How to fix this problem?

Polyalphabetic Ciphers

Ultimate weakness of substitution cipher: each letter is encoded the same way every time it appears.

How to fix this problem? Use a "polyalphabetic cipher": encode letters using different alphabets according to some key.

Simplest method: associate numbers to letters ($a = 0,\ b = 1,\ \ldots,\ z = 25$), then add nth plaintext letter to nth key letter:

Polyalphabetic Ciphers

Ultimate weakness of substitution cipher: each letter is encoded the same way every time it appears.

How to fix this problem? Use a "polyalphabetic cipher": encode letters using different alphabets according to some key.

Simplest method: associate numbers to letters ($a = 0,\ b = 1,\ \dots,$ $z = 25$), then add nth plaintext letter to nth key letter:

Example

```
  plaintext
+ keyletter
-----------
= zpytrmxbk
```

($p = 15$, $k = 10$ *so* $p + k = 25 = z$, *etc.)*

The Vigenère Cipher

The Vigenère cipher (circa 1550s):

- Choose special keyword.
- Key text = keyword repeated over and over.

The Vigenère Cipher

The Vigenère cipher (circa 1550s):

- Choose special keyword.
- Key text = keyword repeated over and over.

Example

```
  samplemessage
+ keykeykeykeyk
---------------
= cekzpcwiqceeo
```

Some people[citation needed] thought this cipher unbreakable.

The Vigenère Cipher

The Vigenère cipher (circa 1550s):

- Choose special keyword.
- Key text = keyword repeated over and over.

Example

```
  samplemessage
+ keykeykeykeyk
---------------
= cekzpcwiqceeo
```

Some people[citation needed] thought this cipher unbreakable. (It's not!) Main weakness: the finite keyword makes keytext too predictable.

Running-Key Ciphers

How to fix key issue with Vigenère?

Running-Key Ciphers

How to fix key issue with Vigenère?

- Use an unbounded key. But where to get a long text?
- One answer: use a book or other common text as the key.

Example

```
  samplemessage
+ tobeornottobe
---------------
= lontzvzsllohi
```

Running-Key Ciphers

How to fix key issue with Vigenère?

- Use an unbounded key. But where to get a long text?
- One answer: use a book or other common text as the key.

Example

```
  samplemessage
+ tobeornottobe
---------------
= lontzvzsllohi
```

Works fairly well... but long words in a predictable keytext can be guessed. Still not secure!

One-Time Pads

How to fix predictability of running-key cipher? Use random letters!

Example

```
  samplemessage
+ hqnyjiefsehpb
---------------
= zqznumqjkwhvf
```

One-Time Pads

How to fix predictability of running-key cipher? Use random letters!

Example

```
  samplemessage
+ hqnyjiefsehpb
---------------
= zqznumqjkwhvf
```

This turns out to be provably unbreakable... in theory. But some issues in practice:

- Hard for computers and humans to generate random letters.
- Must keep sender + receiver keys secure indefinitely.
- Must use key only once (then destroy it).
- No protection if key is secretly copied.

Other Historical Cryptosystems

Some historically notable cryptosystems used before the 1960s:

- Playfair cipher: invented 1850, used by British in WWI.
- Hill cipher: invented 1929, based on linear algebra. (Little-used.)
- ADFGX/ADFGVX: invented 1918, used by Germans in WWI. Broken by French lieutenant in June 1918.
- Enigma: encoding machines, used by Germans in WWII. Broken by Allied efforts, mostly because of improper use of the system.

Symmetric Cryptography

All examples were “symmetric”: information needed to encode $=$ information needed to decode.

Issue: How do Alice and Bob share a key if they're far apart?

Symmetric Cryptography

All examples were "symmetric": information needed to encode = information needed to decode.

Issue: How do Alice and Bob share a key if they're far apart?

- Can't send it unencrypted (Eve reads it), so must encrypt it.
- How do they decide on second key to encrypt first key?
- Can't send second key unencrypted, so must encrypt it...

Symmetric Cryptography

All examples were "symmetric": information needed to encode = information needed to decode.

Issue: How do Alice and Bob share a key if they're far apart?

- Can't send it unencrypted (Eve reads it), so must encrypt it.
- How do they decide on second key to encrypt first key?
- Can't send second key unencrypted, so must encrypt it... but then how do they share third key? (Etc.)

Symmetric Cryptography

All examples were "symmetric": information needed to encode = information needed to decode.

Issue: How do Alice and Bob share a key if they're far apart?

- Can't send it unencrypted (Eve reads it), so must encrypt it.
- How do they decide on second key to encrypt first key?
- Can't send second key unencrypted, so must encrypt it... but then how do they share third key? (Etc.)

What Alice and Bob need is an "asymmetric" cryptosystem: one where information needed to encode $\neq$ information needed to decode.

Asymmetric Cryptography

We want an asymmetric cryptosystem. How to do this?

Asymmetric Cryptography

We want an asymmetric cryptosystem. How to do this?

- Basic idea: "one-way function", a function easy to evaluate but hard to invert.
- Many examples come from number theory.

Asymmetric Cryptography

We want an asymmetric cryptosystem. How to do this?

- Basic idea: "one-way function", a function easy to evaluate but hard to invert.
- Many examples come from number theory.
- Simple example: $f(p, q) = pq$, for primes p and q.
- Multiplication is easy: can find $f(331, 443) = 146633$ by hand.
- Factoring is hard: try finding p, q with $f(p, q) = 339281$ by hand.

Public-Key Cryptography

Basic goal: build a cryptosystem around a one-way function, where encoding = evaluating and decoding = inverting.

Public-Key Cryptography

Basic goal: build a cryptosystem around a one-way function, where encoding = evaluating and decoding = inverting.

Can even make "public-key" cryptosystems:

- Bob makes encoding mechanism freely available to everyone. (Eve too!)
- Alice uses encoding mechanism to send messages to Bob.
- Even with knowledge of encoding mechanism, Eve cannot decode Alice's messages.

Public-Key Cryptography

Basic goal: build a cryptosystem around a one-way function, where encoding = evaluating and decoding = inverting.

Can even make "public-key" cryptosystems:

- Bob makes encoding mechanism freely available to everyone. (Eve too!)
- Alice uses encoding mechanism to send messages to Bob.
- Even with knowledge of encoding mechanism, Eve cannot decode Alice's messages.

Please take a moment to reflect on how amazing this idea is, even if you're familiar with it already.

Modular arithmetic, I

Need to do a bit of number theory to go further.

Definition

If m is a positive integer, we say $a \equiv b \pmod{m}$ if m divides $b - a$.

Modular arithmetic, I

Need to do a bit of number theory to go further.

Definition

If m is a positive integer, we say $a \equiv b \pmod{m}$ if m divides $b - a$.

Example

We have $3 \equiv 21 \pmod{6}$, $11 \equiv -3 \pmod{7}$, and $10^{10} \equiv (-1)^{10} \equiv 1 \pmod{11}$.

Arithmetic behaves normally: we can add, subtract, multiply congruences with the same modulus.

Modular arithmetic, II

Our main interest is in finding powers.

Question

Find a with $a \equiv 3^{2056} \pmod{22}$ and $0 \leq a \leq 21$.

Modular arithmetic, II

Our main interest is in finding powers.

Question

Find a with $a \equiv 3^{2056}$ *(mod 22) and* $0 \leq a \leq 21$.

Some ways to try doing this:

- Very dumb way: Evaluate 3^{2056}, divide by 22, find remainder.
- Dumb way: Evaluate 3^{2056} but reduce mod 22 each time.

Modular arithmetic, II

Our main interest is in finding powers.

Question

Find a with $a \equiv 3^{2056} \pmod{22}$ and $0 \leq a \leq 21$.

Some ways to try doing this:

- Very dumb way: Evaluate 3^{2056}, divide by 22, find remainder.
- Dumb way: Evaluate 3^{2056} but reduce mod 22 each time.
- Smarter way: Evaluate $3^1, 3^2, 3^4, 3^8, \ldots, 3^{2048} \pmod{22}$ by squaring each time. Then $3^{2056} = 3^{2048} \cdot 3^8$.

Modular arithmetic, III

Recall that two integers are "relatively prime" if they have no common divisors other than ± 1.

Definition

If m is a positive integer, then the Euler φ-function $\varphi(m)$ is defined to be the number of integers k, $1 \leq k \leq m$ relatively prime to m.

Example: $\varphi(10) = 4$ (only 1, 3, 7, 9 are relatively prime to 10).

Modular arithmetic, III

Recall that two integers are "relatively prime" if they have no common divisors other than ± 1.

Definition

If m is a positive integer, then the Euler φ-function $\varphi(m)$ is defined to be the number of integers k, $1 \leq k \leq m$ relatively prime to m.

Example: $\varphi(10) = 4$ (only 1, 3, 7, 9 are relatively prime to 10).

Fact

If $m = pq$ is a product of two distinct primes, then $\varphi(m) = (p-1)(q-1)$.

(There is a more general formula, but we don't need it.)

Modular arithmetic, IV

Now we can state the key result about powers we need:

Theorem (Euler's Theorem)

If m is a positive integer and a is relatively prime to m, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Modular arithmetic, IV

Now we can state the key result about powers we need:

Theorem (Euler's Theorem)

If m is a positive integer and a is relatively prime to m, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Example

Euler's Theorem says $7^4 \equiv 1 \pmod{10}$ and $2^{24} \equiv 1 \pmod{35}$.

The RSA Encryption System, I

At last, we can describe the RSA public-key cryptosystem:

The RSA Encryption System, I

At last, we can describe the RSA public-key cryptosystem:

- Bob chooses two large primes p and q and computes $N = pq$.
- Bob also chooses an integer e relatively prime to $\varphi(N) = (p-1)(q-1)$.
- Then Bob publishes his "public key" consisting of N and e.

The RSA Encryption System, I

At last, we can describe the RSA public-key cryptosystem:

- Bob chooses two large primes p and q and computes $N = pq$.
- Bob also chooses an integer e relatively prime to $\varphi(N) = (p-1)(q-1)$.
- Then Bob publishes his "public key" consisting of N and e.

For Alice to send Bob an encrypted message:

- Alice breaks message into a sequence of nonnegative integers each less than N and encrypts each separately.
- Alice encrypts m as $c \equiv m^e \pmod{N}$ and sends c to Bob.

The RSA Encryption System, II

How does Bob decrypt?

The RSA Encryption System, II

How does Bob decrypt?

- Bob first finds a value d such that $de \equiv 1 \pmod{\varphi(N)}$.
- He can do this very quickly using the Euclidean algorithm.
- To decrypt a ciphertext c, Bob computes $m \equiv c^d \pmod{N}$.

The RSA Encryption System, II

How does Bob decrypt?

- Bob first finds a value d such that $de \equiv 1 \pmod{\varphi(N)}$.
- He can do this very quickly using the Euclidean algorithm.
- To decrypt a ciphertext c, Bob computes $m \equiv c^d \pmod{N}$.

Here is an example:

- Bob picks $p = 5$ and $q = 11$ and computes $N = 55$.
- Bob also computes $\varphi(N) = 40$ and picks $e = 3$.
- Alice wants to send $m = 7$ to Bob.
- She computes $c = m^e \equiv 13 \pmod{N}$, and sends it.
- Bob then finds $d = 27$ has $de \equiv 1 \pmod{\varphi(N)}$.
- To decrypt, Bob computes $c^d \equiv 7 \pmod{N}$.

The RSA Encryption System, III

Why does this work?

The RSA Encryption System, III

Why does this work? Euler's theorem:

- Remember that $c \equiv m^e \pmod{N}$.
- So $c^d \equiv m^{de} \pmod{N}$.

The RSA Encryption System, III

Why does this work? Euler's theorem:

- Remember that $c \equiv m^e \pmod{N}$.
- So $c^d \equiv m^{de} \pmod{N}$.
- Also remember $de \equiv 1 \pmod{\varphi(N)}$, so $de = 1 + k\varphi(N)$ for some k.
- Then $m^{de} \equiv m \cdot (m^{\varphi(N)})^k \equiv m \cdot 1^k \equiv m \pmod{N}$, where we used Euler's theorem to observe that $m^{\varphi(N)} \equiv 1 \pmod{N}$.

The RSA Encryption System, III

Why does this work? Euler's theorem:

- Remember that $c \equiv m^e \pmod{N}$.
- So $c^d \equiv m^{de} \pmod{N}$.
- Also remember $de \equiv 1 \pmod{\varphi(N)}$, so $de = 1 + k\varphi(N)$ for some k.
- Then $m^{de} \equiv m \cdot (m^{\varphi(N)})^k \equiv m \cdot 1^k \equiv m \pmod{N}$, where we used Euler's theorem to observe that $m^{\varphi(N)} \equiv 1 \pmod{N}$.

Technically, this explanation only applies when m is relatively prime to N (which is almost always the case), but it can be shown the procedure still works even when m has a common factor with N.

Analysis of RSA, I

The system certainly works. But why is it secure?

Analysis of RSA, I

The system certainly works. But why is it secure?

- Suppose Eve is eavesdropping and records all the data sent, so she has N, e, and c (but not p, q, or d) and wants to find m.
- Eve needs to solve the congruence $m^e = c \pmod{N}$ for m.
- If she could find $\varphi(N)$ then she could compute d the same way Bob did.

Analysis of RSA, I

The system certainly works. But why is it secure?

- Suppose Eve is eavesdropping and records all the data sent, so she has N, e, and c (but not p, q, or d) and wants to find m.
- Eve needs to solve the congruence $m^e \equiv c \pmod{N}$ for m.
- If she could find $\varphi(N)$ then she could compute d the same way Bob did.
- However, if $N = pq$ is a product of two primes, then computing $\varphi(N) = (p-1)(q-1)$ is equivalent to factoring N.
- If Bob picks p and q really large (200 base-10 digits or so), it is extremely hard to factor N.

Analysis of RSA, II

Ultimately, the question is: can Eve find a value of d that works without having to factor N?

Analysis of RSA, II

Ultimately, the question is: can Eve find a value of d that works without having to factor N?

- It turns out that the answer is "probably not".
- Specifically, if Eve found a decryption exponent that could decode any ciphertext c, then she could use it to factor N. (Details are a bit too technical to give here.)

Analysis of RSA, II

Ultimately, the question is: can Eve find a value of d that works without having to factor N?

- It turns out that the answer is "probably not".
- Specifically, if Eve found a decryption exponent that could decode any ciphertext c, then she could use it to factor N. (Details are a bit too technical to give here.)
- Of course, this isn't quite what we asked: Eve just wants to decode one message, not all messages.
- But this suggests breaking RSA is roughly the same difficulty as factoring large numbers.

RSA is not known to be equivalent to factoring. (Some other public-key cryptosystems are.)

Analysis of RSA, III

RSA seems secure, at least if factoring large numbers is slow. But why isn't it slow for Alice and Bob too?

- Bob first needs to find two large primes p and q.
- Perhaps counterintuitively, it is much easier to check whether a number is prime than it is to factor it.
- Primes are also easy to find: a random 200-digit integer not divisible by 2, 3, 5, 7 has about a 1% chance of being prime.
- So Bob can generate p and q quickly even if he wants them to have hundreds of digits.

Analysis of RSA, III

RSA seems secure, at least if factoring large numbers is slow. But why isn't it slow for Alice and Bob too?

- Bob first needs to find two large primes p and q.
- Perhaps counterintuitively, it is much easier to check whether a number is prime than it is to factor it.
- Primes are also easy to find: a random 200-digit integer not divisible by 2, 3, 5, 7 has about a 1% chance of being prime.
- So Bob can generate p and q quickly even if he wants them to have hundreds of digits.
- Likewise, Bob can find the decryption exponent d very fast.
- Finally, Alice and Bob can both do encryption and decryption quickly using successive squaring.

Proving and Verifying

Suppose Alice sends Bob a message using RSA, and she wants verification that Bob actually received it. How can this be done?

Proving and Verifying

Suppose Alice sends Bob a message using RSA, and she wants verification that Bob actually received it. How can this be done?

- Idea 1: Bob sends Alice a response saying "I received your message".
- Issue: Alice has no way to know Bob really sent that message: maybe it was actually Eve.
- Idea 2: Bob sends Alice a response saying "I received your message" and then includes part of Alice's message.
- Issue: Eve might have been impersonating Alice. Now Bob has revealed some of Alice's message to Eve.

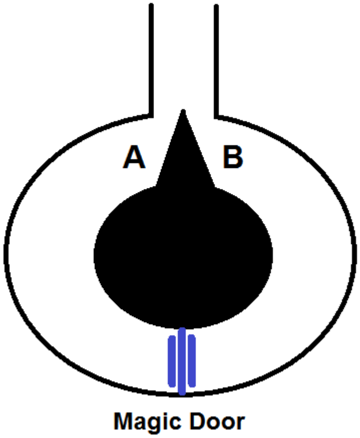
Ultimately, how can Bob prove to Alice that he got her message without revealing anything about it? What we want is a "zero-knowledge proof".

A Zero-Knowledge System, I

Traditional to use Peggy (the prover) and Victor (the verifier).

A Zero-Knowledge System, I

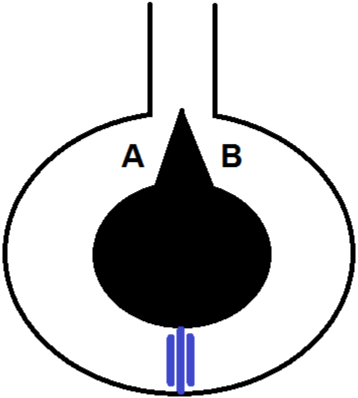
Traditional to use Peggy (the prover) and Victor (the verifier).



- Imagine a cave with one entrance.
- Entrance splits off to two paths: A and B.
- Paths lead to opposite sides of magic door.

A Zero-Knowledge System, I

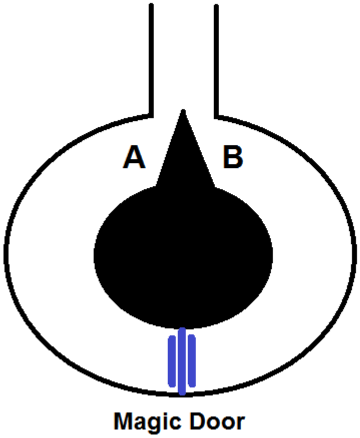
Traditional to use Peggy (the prover) and Victor (the verifier).



- Imagine a cave with one entrance.
- Entrance splits off to two paths: A and B.
- Paths lead to opposite sides of magic door.
- Peggy claims she knows magic words to open door.
- Victor says “prove it”.
- But Peggy doesn’t want to reveal magic words.

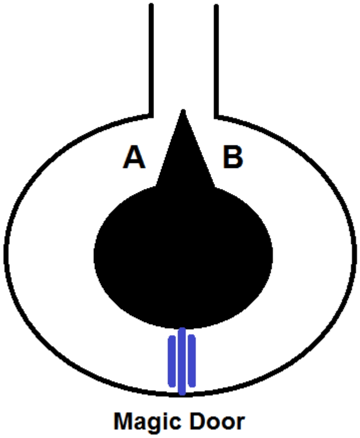
A Zero-Knowledge System, II

How can Peggy prove she knows the magic words to Victor?



A Zero-Knowledge System, II

How can Peggy prove she knows the magic words to Victor?



- Peggy walks into cave along A or B. (Victor can't see which.)
- Victor then calls out "A" or "B".
- Peggy walks back along path Victor chose.
- Victor and Peggy repeat procedure 20 times (or until Victor believes Peggy knows secret).

A Zero-Knowledge System, III

Why does this procedure work?

- If Peggy really knows magic words, she passes every time (she just goes through the door if she's on wrong side).
- If Peggy doesn't know magic words, there is 50% chance she is on wrong side and will fail test.

A Zero-Knowledge System, III

Why does this procedure work?

- If Peggy really knows magic words, she passes every time (she just goes through the door if she's on wrong side).
- If Peggy doesn't know magic words, there is 50% chance she is on wrong side and will fail test.
- For Victor, one single test doesn't give much confidence.
- But repeating test 20 times means there is only a $1/2^{20} \approx 10^{-6}$ probability Peggy doesn't really know magic words but still passed.

A Zero-Knowledge System, III

Why does this procedure work?

- If Peggy really knows magic words, she passes every time (she just goes through the door if she's on wrong side).
- If Peggy doesn't know magic words, there is 50% chance she is on wrong side and will fail test.
- For Victor, one single test doesn't give much confidence.
- But repeating test 20 times means there is only a $1/2^{20} \approx 10^{-6}$ probability Peggy doesn't really know magic words but still passed.
- What about if Eve is watching them?
- Maybe Peggy and Victor conspired together to fake the tests (they just decide ahead of time what Victor will say).
- Eve shouldn't be convinced Peggy knows magic words.

Implementation of Zero-Knowledge System, I

How can we implement this idea? Can do it a bit like RSA:

- Peggy chooses two large primes p and q and a secret s.
- Peggy publishes $N = pq$ and $s^2 \pmod{N}$.
- Victor wants Peggy to prove she knows s.

Implementation of Zero-Knowledge System, I

How can we implement this idea? Can do it a bit like RSA:

- Peggy chooses two large primes p and q and a secret s.
- Peggy publishes $N = pq$ and $s^2 \pmod{N}$.
- Victor wants Peggy to prove she knows s.

Here is the test procedure:

- Peggy chooses a random number r modulo N and sends Victor the value $r^2 \pmod{N}$.
- Victor then asks either for r or for rs modulo N.
- Peggy sends Victor the quantity he requested.
- If Victor asked for r, he checks whether the value squares to r^2 Peggy sent earlier.
- If Victor asked for rs, he checks whether the value squares to r^2s^2 (which he can compute using r^2 and s^2).

Implementation of Zero-Knowledge System, II

For example, suppose Peggy chooses $N = 264389$, $s = 110296$.
She publishes $s^2 \equiv 140948 \pmod{N}$.
Victor asks Peggy to prove she knows s.

Implementation of Zero-Knowledge System, II

For example, suppose Peggy chooses $N = 264389$, $s = 110296$.
She publishes $s^2 \equiv 140948 \pmod{N}$.
Victor asks Peggy to prove she knows s. Round 1:

- Peggy generates $r = 83924$ and sends Victor $r^2 \equiv 179205$.
- Victor says "send me r".
- Peggy sends Victor $r = 83924$, and he verifies $83924^2 \equiv 179205$. Peggy passes this round.

Implementation of Zero-Knowledge System, II

For example, suppose Peggy chooses $N = 264389$, $s = 110296$.
She publishes $s^2 \equiv 140948 \pmod{N}$.
Victor asks Peggy to prove she knows s. Round 1:

- Peggy generates $r = 83924$ and sends Victor $r^2 \equiv 179205$.
- Victor says "send me r".
- Peggy sends Victor $r = 83924$, and he verifies $83924^2 \equiv 179205$. Peggy passes this round.

Round 2:

- Peggy generates $r = 101041$ and sends Victor $r^2 \equiv 166835$.
- Victor says "send me rs".
- Peggy sends Victor $rs \equiv 157397$, and Victor checks $157397^2 \equiv 166835 \cdot 140948$. Peggy passes this round too.

Implementation of Zero-Knowledge System, III

Now suppose Eve tries to impersonate Peggy. Victor asks Eve to prove she knows s.

Implementation of Zero-Knowledge System, III

Now suppose Eve tries to impersonate Peggy. Victor asks Eve to prove she knows s. Round 1:

- Eve knows $N = 264389$ and $s^2 \equiv 140948 \pmod{N}$ but not s itself.
- Eve guesses Victor will ask for r. Eve generates $r = 197866$ and sends Victor $r^2 \equiv 230836$.

Implementation of Zero-Knowledge System, III

Now suppose Eve tries to impersonate Peggy. Victor asks Eve to prove she knows s. Round 1:

- Eve knows $N = 264389$ and $s^2 \equiv 140948 \pmod{N}$ but not s itself.
- Eve guesses Victor will ask for r. Eve generates $r = 197866$ and sends Victor $r^2 \equiv 230836$.
- Victor says "send me r".
- Eve sends Victor $r = 197866$, and he verifies $197866^2 \equiv 230836$.
- Eve passes this round.

Implementation of Zero-Knowledge System, IV

Round 2:

- Eve guesses Victor will ask for r again.
- Eve generates $r = 153819$ and sends Victor $r^2 \equiv 113151$.
- Victor says "send me rs".

Implementation of Zero-Knowledge System, IV

Round 2:

- Eve guesses Victor will ask for r again.
- Eve generates $r = 153819$ and sends Victor $r^2 \equiv 113151$.
- Victor says "send me rs".
- Eve is stuck! She can't send rs because she only knows r and s^2, but not s.
- Whatever Eve sends, Victor will then know Eve didn't really have the value of s.

Analysis of Zero-Knowledge System

Why does this system work? Same as cave example:

- Peggy knows s, so she can always answer Victor's challenges.
- Eve doesn't know s. She only has a 50% chance to answer correctly.
- But why can't Eve find s from s^2 and N?

Analysis of Zero-Knowledge System

Why does this system work? Same as cave example:

- Peggy knows s, so she can always answer Victor's challenges.
- Eve doesn't know s. She only has a 50% chance to answer correctly.
- But why can't Eve find s from s^2 and N?
- Answer: being able to compute square roots mod N is equivalent to factoring $N = pq$. Factoring is hard, so Eve probably can't do this.

Analysis of Zero-Knowledge System

Why does this system work? Same as cave example:

- Peggy knows s, so she can always answer Victor's challenges.
- Eve doesn't know s. She only has a 50% chance to answer correctly.
- But why can't Eve find s from s^2 and N?
- Answer: being able to compute square roots mod N is equivalent to factoring $N = pq$. Factoring is hard, so Eve probably can't do this.
- Finally, Eve doesn't gain any information from spying on Peggy and Victor, since r and rs are never both sent during a single round.

Applications of Cryptography

Now that you know a few of the ideas, how can we use them? A few examples:

- Sending secure communications across insecure channels.
- Proving you know a secret without revealing it to anyone.

Applications of Cryptography

Now that you know a few of the ideas, how can we use them? A few examples:

- Sending secure communications across insecure channels.
- Proving you know a secret without revealing it to anyone.
- Creating digital signatures that can't be forged or changed.
- Designing secure digital voting systems.
- Sharing a distributed secret among several parties.

Applications of Cryptography

Now that you know a few of the ideas, how can we use them? A few examples:

- Sending secure communications across insecure channels.
- Proving you know a secret without revealing it to anyone.
- Creating digital signatures that can't be forged or changed.
- Designing secure digital voting systems.
- Sharing a distributed secret among several parties.

There are lots more, but I think I'll stop here.