The number theory behind cryptography

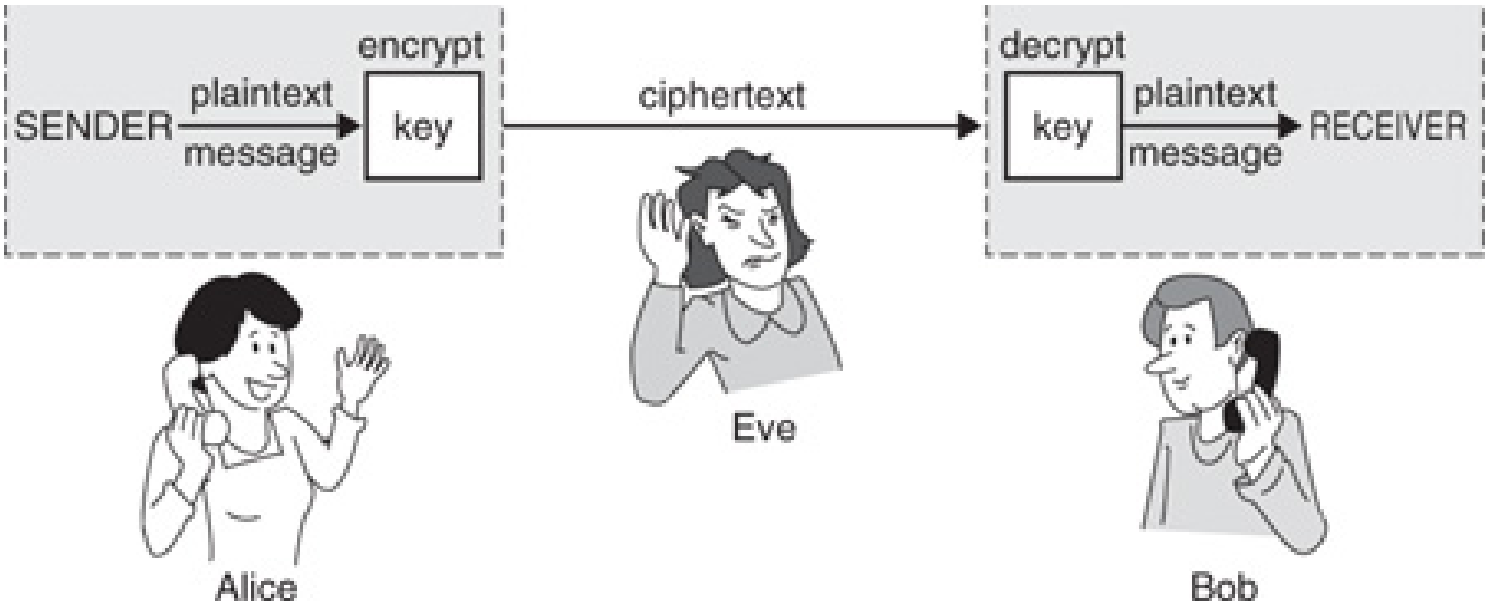
Christelle Vincent

The University of Vermont

May 16, 2017

What is cryptography?

Cryptography is the practice and study of techniques for **secure communication in the presence of adverse third parties.**



What is cryptography?

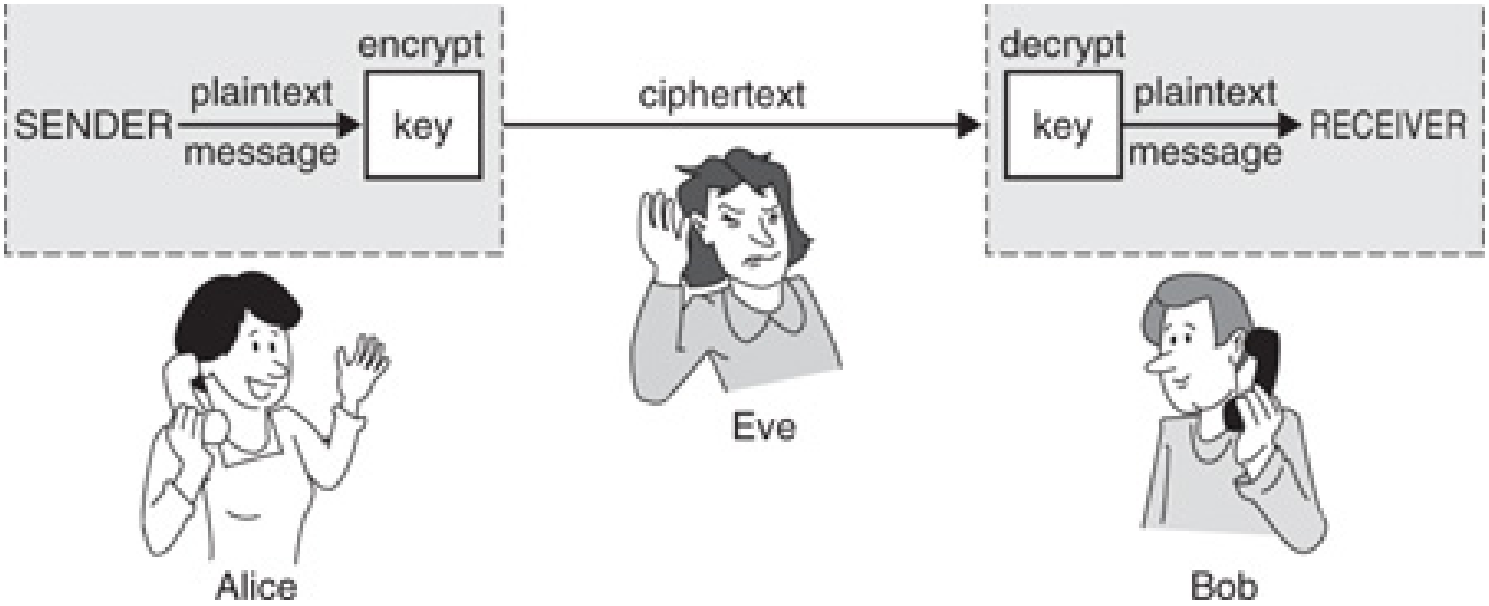
In other words, it is about **constructing** and **analyzing** ways to communicate that prevent third parties from reading private messages.

Modern applications: all of the internet.

Organization of this talk

1. Vocabulary and history
2. Mathematical preliminaries
3. Discrete logarithm problem
4. Digital Signature Algorithm
5. Attacking DSA

Some vocabulary



Encryption refers to the process of changing ordinary text (*plaintext*) into unintelligible text (*ciphertext*).

Decryption is the reverse.

Some vocabulary

A *cipher* is the algorithm used to encrypt and decrypt.

The operation of most good ciphers is controlled both by the algorithm and a parameter called a *key*.

Example: Caesar cipher

The algorithm of the Caesar cipher is to replace each letter by the kth letter preceding it in the alphabet.

The specific choice of k in a given instance of this cipher is the key.

Example: Caesar cipher

For example, given the plaintext "Hello world" and the key $k = 2$, we replace "H" with the letter "F," "e" with "c," "l" with "j," etc. to produce the ciphertext "Fcjjm umpjb."

Example: Caesar cipher

To decrypt, use the **same** key k, but choose the kth letter following a given letter in the alphabet.

Cryptography then: Secret key cryptography

Main idea: Alice and Bob share **privately** the cipher and/or key they will use to communicate.

Examples:

- Caesar cipher ($\sim$100BC)
- Enigma machine (WWII)

Cryptography then: Secret key cryptography

This is also often called "symmetric key" cryptography, since Alice and Bob use the **same** secret key to encrypt and decrypt the message.

Drawback: Requires a secure exchange before setting up the secure exchange...

Cryptography now: Public key cryptography

In June 1976, Diffie and Hellman proposed the notion of *public key* or asymmetric key cryptography.[1]

In such a cryptosystem, Bob generates **two** sets of keys, one public and one private.

[1] Actually this was discovered in 1970 by Ellis at Government Communications Headquarters, but it was classified until 1997.

Cryptography now: Public key cryptography

Alice uses Bob's public key to encrypt a message to Bob,

and then Bob uses his private key to decrypt the message.

Main idea

Public or asymmetric key cryptography relies on one thing:

There are some things in mathematics that are easy to do, but difficult to undo.

An example of a suitable problem

Which problem would you rather see appear on an exam:

1. Perform the multiplication 1489×701; or
2. Factor the number $1,043,789$.

Rough idea

Bob has the two primes 1489 and 701 as his private key, and the product $1,043,789$ as his public key.

He can publish his public key for everyone to see without fear that they can recover his private key, since factoring is difficult.[2]

[2] At least we think it is.

One-slide detour: the RSA algorithm

In 1978, **R**ivest, **S**hamir and **A**dleman published the first public key cryptography system, which is now called the RSA algorithm.[3]

RSA's security relies on the fact that it is easy to multiply but hard to factor.

[3] Actually an equivalent system was developed by Cocks at GCHQ in 1973.

Our focus

In this talk I will focus on cryptosystems relying on a different difficult problem: the discrete logarithm problem.

The division algorithm

Theorem

Let a and n be two positive integers.
Then there exist unique q and r, with $0 \leq r < n$ with

$$a = qn + r.$$

We call r the *remainder* of the division of a by n.

The division algorithm

Example

Let $a = 101$ and $n = 13$ then

$$101 = 7 \cdot 13 + 10,$$

so the remainder is $r = 10$.

Example

Let $a = 67$ and $n = 13$ then

$$67 = 5 \cdot 13 + 2,$$

so the remainder is $r = 2$.

Remainders are interesting!

Question

Now let $a = 101 + 67 = 168$. What is the remainder when we divide by $n = 13$?

It is $10 + 2 = 12$ (!). Indeed

$$168 = 12 \cdot 13 + 12.$$

Remainders are interesting!

Question

Now let $a = 101 \cdot 67 = 6767$. What is the remainder when we divide by $n = 13$?

It cannot be $10 \cdot 2 = 20$, because that is too big. But 6767 has the same remainder as 20 (!). Indeed

$$20 = 13 + 7$$

and

$$6767 = 520 \cdot 13 + 7.$$

How can that be?

Actually, this follows from facts you know:

$$\begin{aligned} 101 + 67 &= (7 \cdot 13 + 10) + (5 \cdot 13 + 2) \\ &= 7 \cdot 13 + 5 \cdot 13 + 10 + 2 \\ &= (7 + 5) \cdot 13 + 12 \end{aligned}$$

$$\begin{aligned} 101 \cdot 67 &= (7 \cdot 13 + 10) \cdot (5 \cdot 13 + 2) \\ &= 7 \cdot 5 \cdot 13 \cdot 13 + 10 \cdot 5 \cdot 13 + 7 \cdot 2 \cdot 13 + 10 \cdot 2 \\ &= (7 \cdot 5 \cdot 13 + 10 \cdot 5 + 7 \cdot 2) \cdot 13 + 20 \\ &= (7 \cdot 5 \cdot 13 + 10 \cdot 5 + 7 \cdot 2 + 1) \cdot 13 + 7 \end{aligned}$$

Congruence modulo n

Definition

Let $0 \leq r < n$. We say that

$$a \equiv r \pmod{n}$$

if r is the remainder of a when we divide by n.

What we did always works!

It turns out that we can add and multiply remainders much like we can add and multiply integers.

Furthermore, the addition and multiplication for remainders satisfies the same properties as addition and multiplication for integers.

A familiar example: clocks

Question

It is 7pm now. What time will it be in 3 hours? In 10 hours? In 50 hours?

- The first one is easy: It will be 10pm.
- The second one we can think of this way: In 5 hours it will be midnight, then in 5 more hours it will be 5am.

  Another way to this about this is like this

  $$7 + 10 = 17$$

  and 17 has remainder 5 when we divide by 12.

A familiar example: clocks

Question

It is 7pm now. What time will it be in 3 hours? In 10 hours? In 50 hours?

- The third one we can think of this way:

$$50 = 4 \cdot 12 + 2.$$

So in 48 hours it will be 7 again. Then two hours later it will be 9. Another way to think about it is this:

$$7 + 50 = 57$$

and 57 has remainder 9 when divided by 12.

Our clocks

For our purposes, we always like n to be a prime number.

So for example, we might do our arithmetic with $n = 13$, as we did before. There we have

$$1 + 5 \equiv 6 \pmod{13}$$
$$4 + 7 \equiv 11 \pmod{13}$$
$$7 + 7 \equiv 1 \pmod{13}$$
$$10 + 8 \equiv 5 \pmod{13}$$

An important fact

When n is prime, for any $a \neq 0$, there is b with

$$ab \equiv 1 \pmod{n}.$$

We call b the multiplicative inverse of a and denote it a^{-1}.

Multiplying by b plays the role of dividing by a. Indeed, if

$$ax \equiv c \pmod{n}$$

then

$$bax \equiv bc \pmod{n}$$

or

$$x \equiv bc \pmod{n}.$$

Example

For example let $n = 13$ and $a = 3$. Then

$$3 \cdot 9 \equiv 1 \pmod{13}.$$

This is because

$$3 \cdot 9 = 27$$

and

$$27 = 2 \cdot 13 + 1.$$

Therefore

$$3 \cdot 9 = 27 \equiv 1 \pmod{13}.$$

All of this is easy

It turns out that everything we have seen so far is easy for a computer to do:

- Addition and multiplication modulo n is just "normal" addition and multiplication plus finding the remainder
- Exponentiation (for example computing 2^{10} modulo 13) is easy because it is just repeated multiplication.
- Although we will not get into it, finding a^{-1} modulo n is also easy using Euclid's Algorithm

Our difficult problem: discrete log problem

Suppose that I give you

$$a = 2,$$
$$b = 5,$$
$$\text{and} \quad n = 13$$

and ask you to find k such that

$$2^k \equiv 5 \pmod{13}.$$

What would you do?

Our difficult problem: discrete log problem

In general, given a and b such that

$$a^k \equiv b \pmod{n},$$

computing $k = \log_a b$ is difficult.

Important note about "difficult"

We do not mean here that computing the logarithm is difficult to do by hand!

If someone asked me what $\log_2 5$ was, we can use a calculator. (It is ≈ 2.32192809488736.)

What we mean is that there is **no way** for a calculator to compute $\log_2 5$ modulo 13. (!)

Taxonomy of problems

We have some easy problems:

- multiplication and addition
- exponentiation
- computing a^{-1} modulo n

And a hard problem:

- computing the discrete logarithm

How to use the DLP as a cryptoscheme

So that Bob can receive secret messages, he needs to make the following preparations:

1. Bob chooses a prime p and an integer a relatively prime to p
2. Bob chooses an integer k
3. Bob computes $b \equiv a^k \pmod{p}$
4. Bob publishes a, b and p, but keeps k private

How to use this as a cryptoscheme

To send the secret number m to Bob, Alice now follows these steps:

1. Alice chooses a random number y and computes

$$c_1 \equiv a^y \pmod{p} \quad \text{and} \quad c_2 \equiv mb^y \pmod{p}$$

2. Alice sends c_1 and c_2 to Bob publicly, and keeps y private

How to use this as a cryptoscheme

To decrypt the message m,
Bob simply computes $(c_2)(c_1^k)^{-1}$ because

$$\begin{aligned}(c_2)(c_1^k)^{-1} &\equiv (mb^y)((a^y)^k)^{-1} \pmod{p} \\ &\equiv mb^y((a^k)^y)^{-1} \pmod{p} \\ &\equiv mb^y(b^y)^{-1} \pmod{p} \\ &\equiv m \pmod{p}.\end{aligned}$$

The idea

If either k or y became known,
the secret message m could be recovered.

Alice uses b^y to "hide" her message m ($c_2 = mb^y$).

Alice "hides" her y inside a^y for Bob to "find."

Bob uses his knowledge of k to cancel the y's: Since $a^k = b$,

$$b^y = (a^k)^y = (a^y)^k.$$

The point

Both Bob and Alice only perform "easy" operations.

As long as k and y remain secret, no one but Bob or Alice can read the message, even if they know

$$a$$

$$p$$

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

$$c_1 \equiv a^y \pmod{p}$$

$$c_2 \equiv mb^y \pmod{p}.$$

To get the message, they must compute either $k = \log_a b$ or $y = \log_a c_1$.

Crypto using DLP: Example

Suppose that Bob publishes

$$p = 13, \quad a = 2, \quad b = 5.$$

(Bob knows that $2^9 \equiv 5 \pmod{13}$ but no one else can know.)

Alice wants to send the message $m = 4$ to Bob.
She first chooses $y = 7$ randomly.
She sends

$$c_1 = 11 \equiv 2^7 \pmod{13}, \quad c_2 = 6 \equiv 4 \cdot 5^7 \pmod{13}.$$

Then Bob computes

$$6 \cdot (11^9)^{-1} \equiv 4 \pmod{13}.$$

Vocabulary and history

Mathematical preliminaries

Discrete logarithm problem

4. Digital Signature Algorithm

Attacking DSA

Digital signatures

A *digital signature* is a process by which an entity proves that it is who it claims to be.

One way to do this is for the entity to prove knowledge of its secret key (without disclosing it).

Digital signatures

For example if Alice is writing a message to Bob,
Bob might want Alice to prove that she is the person she claims to be.

How is that even possible? DSA

To prove her identity, Alice would need to first prepare her own DLP.

Therefore we assume that Alice has published her own

$$a_A, \quad b_A (\equiv a_A^{k_A} \pmod{p_A}) \quad \text{and} \quad p_A.$$

Also, Alice has sent the message m encrypted with Bob's public key

How is that even possible? DSA

To prove that Alice is really Alice, she proves that she has access to k_A:

1. She generates a random integer $0 < \ell < p_A - 1$
2. She computes $r \equiv a_A^\ell \pmod{p_A}$
3. She also computes $s \equiv \ell^{-1}(m + k_A r) \pmod{p_A - 1}$

Her signature is (r, s), which she publishes.

Note that ℓ and k_A remain secret. (And only Bob knows m.)

How is that even possible? DSA

To check that Alice knows k_A, Bob computes

$$a_A^{ms^{-1}} b_A^{rs^{-1}}$$

and checks that this is equal to r.

Indeed,

$$\begin{aligned} a_A^{ms^{-1}} b_A^{rs^{-1}} &\equiv a_A^{ms^{-1}} (a_A^{k_A})^{rs^{-1}} \pmod{p_A} \\ &\equiv a_A^{s^{-1}(m+k_A r)} \pmod{p_A} \\ &\equiv a_A^{\ell} \pmod{p_A} \\ &\equiv r. \end{aligned}$$

Note that all that Bob needs is access to m,
and this only works if Alice has used k_A.

DSA: Example

Alice sent the message $m = 4$ to Bob. He decrypted it as we did above.

To sign her message, Alice needs her own a, b and p. She chooses

$$a = 3, \quad b = 4, \quad p = 17.$$

She signs her message $(5, 6)$.

Bob verifies that the message 4 is indeed from her because

$$\begin{aligned} 3^{4 \cdot 4^{-1}} \cdot 4^{5 \cdot 6^{-1}} &\equiv 3 \cdot 4 \cdot 16 \pmod{17} \\ &\equiv 5 \pmod{17} \end{aligned}$$

One use of DSA

Modern game consoles are prevented from installing software that does not come from their manufacturer.

This is done by telling the console to accept software updates only from sources that give the correct signature.

In other words, the console only accepts software from sources that know the correct value of "k."

2012 attack on PS3

In 2012, the Three Musketeers revealed that they obtained access to the Sony "k" value.

The public key (a, b, p) for this value was embedded in the hardware of every PS3 produced to date.

Sony used it to sign the code which further controlled the security of the whole PS3.

With this key PS3 could be jailbroken and made to run pirated games or any other software.

Because this affected the lowest level of security of the PS3, this could not be patched.

How could this happen?!

To certify the authenticity of its code, Sony used the same value of ℓ repeatedly.

This is easy to notice: the first parameter of the signature is

$$r \equiv a^{\ell} \pmod{p};$$

two signatures with the same r have used the same ℓ.

Just solve for ℓ

Once they received two signatures (r, s_1), (r, s_2), recall that

$$s_i \equiv \ell^{-1}(m_i + kr) \pmod{p-1}$$

Solving for m_i we get

$$m_i \equiv s_i\ell - kr \pmod{p-1}.$$

So

$$m_1 - m_2 \equiv (s_1\ell - kr) - (s_2\ell - kr) \equiv (s_1 - s_2)\ell \pmod{p-1}.$$

And now solve for k

It is now trivial to recover k:

$$k \equiv r^{-1}(s\ell - m) \pmod{p-1}.$$

To learn more/resources

- A series of crypto challenge problems:
  `http://cryptopals.com`
- Matthew Green's blog:
  `http://blog.cryptographyengineering.com/`
- Wikipedia

Thank you!