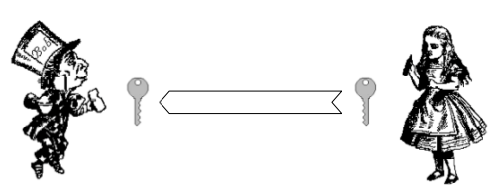


# Public-Key Cryptography

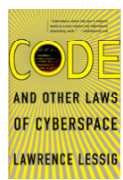
## Public-Key Cryptography



Eric Roberts  
CS 54N  
November 7, 2016


## Public-Key Encryption

- In 1999, shortly before he came to Stanford for an eight-year stay at the Law School, Larry Lessig wrote a book entitled *Code and Other Laws of Cyberspace*.
- In his book, Lessig argues—with at least a bit of hyperbole—that cryptography is the most revolutionary development of the last millennium.



Here is something that will sound very extreme but is at most, I think, a slight exaggeration: encryption technologies are the most important breakthrough in the last thousand years. No other technological discovery . . . will have a more significant impact on our social and political life. Cryptography will change everything.

## Idealized Model of Encryption



Doo plpvb zhuh wkh erurjytw.

Twes brillig, and the slithy toves,  
Did gyre and gimble in the wabe:  
All mimsy were the borogoves.


Twes brillig, and the slithy toves,  
Did gyre and gimble in the wabe:  
All mimsy were the borogoves.

## The Search for a Perfect Code

- The problem of secure encryption illustrated on the previous slide does have a solution, at least in theory. One strategy is to have the sender and receiver share a codebook with the following properties:
  - The codebook is so large that none of it is ever reused.
  - The mapping of plaintext letters is *random* in the sense that past transformations convey no information about later ones.
- Because the individual entries in the codebook are never reused, this approach is generally called a *one-time pad*. It is also called a *Vernam cipher* after Gilbert Vernam, an engineer at AT&T Bell Labs who patented the technique in 1917.
- In 2011, the *New York Times* reported that this technique had been described earlier in an 1882 monograph by Frank Miller, a San Francisco banker who later became a Stanford trustee.

## Use of the One-Time Pad

- The one-time pad strategy has been used in practice on several occasions.
  - The Soviet Union used a one-time pad for its communications in World War II. Operators, however, occasionally reused pages from the codebook, which enabled British and American codebreakers to decrypt some messages.
  - When he was killed in Bolivia in 1968, Che Guevara was carrying a one-time pad to communicate with Fidel Castro.
- In practice, the one-time pad is problematic because of the size of the codebook and the difficulty of distributing it securely.



## Solving the Key-Distribution Problem

- The problem of distributing encryption keys—no matter whether they are complete codebooks used in one-time pads or shorter keys used in other encryption strategies—is hard only if the keys must be kept secret.
- The idea that keys can be made public at first seems crazy. After all, if a key is public than anyone can use it.
- That fact, however, causes problems only if knowing the public key makes it possible for an eavesdropper to read messages to which they should not have access. If knowing the public key lets anyone *encrypt* a message but allows only the intended recipient to *decrypt* that message, the problem goes away.
- A coding strategy that allows encryption keys to be shared but protects decryption keys is called *public-key encryption*.

### GCHQ Invents Public-Key Encryption

- Public-key cryptography was invented in England by the Government Communications Headquarters (GCHQ), which succeeded the Government Code and Cipher School (GCCS) that broke the Enigma code in World War II.
- The basic ideas of public-key encryption were developed in the early 1970s by the GCHQ cryptographers James Ellis, Clifford Cocks, and Malcolm Williamson. Their work went well beyond the fundamental concepts to anticipate many of the practical cryptographic protocols that were subsequently rediscovered in the United States.
- Unfortunately, none of the later researchers knew about the British cryptographic work. At GCHQ, everything relating to public-key cryptography remained classified until late in the 1990s, long after this technology was commercialized.

### Stanford Rediscovered the Idea

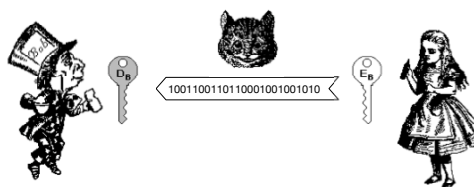
- The first unclassified announcement of public-key encryption appeared in 1976 in a paper by Stanford graduate student Whit Diffie and his advisor Martin Hellman. The two appear in the picture to the right along with another early collaborator, Ralph Merkle. Diffie and Hellman won the 2016 Turing Award.
- The Stanford researchers not only laid out the basic structure of public-key encryption but also proposed an implementation based on a variant of the *subset-sum problem*, which is an NP-complete problem.
- Unfortunately, the specific variant of the problem they chose was considerably easier to solve.



### Whit Diffie Recalls the Discovery



### Public-Key Encryption



1. Bob generates a pair of keys labeled  $D_B$  and  $E_B$ .
2. Bob publishes  $E_B$  as his *public key*.
3. Bob keeps  $D_B$  hidden as his *private key*.
4. Alice uses  $E_B$  to send a secret message to Bob.
5. Only someone with  $D_B$  can decipher the message.

### RSA Makes the Idea Practical

- In 1977, a team consisting of Ron Rivest, Adi Shamir, and Len Adleman—all then at MIT—developed a practical implementation of public-key encryption, which they called *RSA* after their initials.
- The RSA scheme relies on the difficulty of factoring large numbers. As long as the number is large enough—typically many hundreds of digits—revealing the product of two prime numbers  $p$  and  $q$  does not allow an eavesdropper to recover the original values.



### Key Generation in RSA

- As in any public-key encryption scheme, RSA requires each potential recipient to generate two keys, a *public key* that allows anyone to send an encrypted message and a *private key* that ensures that only the recipient can decrypt that message.
- Generating an RSA key pair requires the following steps:
  1. Choose two prime numbers  $p$  and  $q$ .
  2. Define the variable  $n$  as  $pq$  and the variable  $t$  as  $(p-1)(q-1)$ .
  3. Choose an integer  $d < n$  so that  $d$  is *relatively prime* to  $t$ . Two integers are relatively prime if they share no common factors.
  4. Set  $e$  to be the *modular inverse* of  $d$  with respect to  $t$ , which is the integer for which the product  $de$  divided by  $t$  leaves a remainder of 1. The fact that  $d$  and  $t$  are relatively prime means that this value is unique.
  5. Release  $n$  and  $e$  as the public key and use  $d$  as the private key.

## Encryption and Decryption in RSA

- Given the public-key parameters  $n$  and  $e$ , the sender creates an encrypted message  $c$  using the following steps:

- Convert the message into a binary integer  $m$  by using the internal character codes. If the message length exceeds the size of  $n$ , the sender must break it into smaller pieces and encrypt each one individually.

- The sender then computes the ciphertext  $c$  as follows:

$$c = m^e \bmod n$$

- The recipient restores the original message by calculating

$$m' = c^d \bmod n$$

## A Tiny Example (Key Generation)

- Choose two prime numbers  $p$  and  $q$ .

$$p = 11$$

$$q = 23$$

- Define the variable  $n$  as  $pq$  and the variable  $t$  as  $(p-1)(q-1)$ .

$$n = 253$$

$$t = 220$$

- Choose an integer  $d < n$  so that  $d$  is *relatively prime* to  $t$ .

$$d = 17$$

- Set  $e$  to be the *modular inverse* of  $d$  with respect to  $t$ .

$$e = 13 \quad (17 \times 13 = 221; 221 \bmod 220 = 1)$$

- Release  $n$  and  $e$  as the public key and use  $d$  as the private key.

## A Tiny Example (Encryption)

$$n = 253$$

$$e = 13$$

$$d = 17$$

- Convert the message into an integer  $m$ .

$$m = 'A' = 65$$

- Create the ciphertext  $c$  by calculating  $m^e \bmod n$ .

$$m^e = 369720589101871337890625$$

$$c = 76$$

- Validate the encryption by calculating  $c^d \bmod n$ .

$$c^d = 94152329294455713577749264203776$$

$$m' = 65$$

## The Magic of Modular Arithmetic

- The very small example on the preceding slide requires the calculation of  $m^e \bmod n$  for  $m = 65$ ,  $e = 13$ , and  $n = 253$ .
- The numbers stay much smaller if you apply the mod operator as you go along.

$m^1 = 65$	$m^1 \bmod n = 65$
$m^2 = 4225$	$m^2 \bmod n = 177$
$m^3 = 274625$	$m^3 \bmod n = 120$
$m^4 = 17850625$	$m^4 \bmod n = 210$
$m^5 = 1160290625$	$m^5 \bmod n = 241$
$m^6 = 75418890625$	$m^6 \bmod n = 232$
$m^7 = 4902227890625$	$m^7 \bmod n = 153$
$m^8 = 318644812890625$	$m^8 \bmod n = 78$
$m^9 = 20711912837890625$	$m^9 \bmod n = 10$
$m^{10} = 1346274334462890625$	$m^{10} \bmod n = 144$
$m^{11} = 87507831740087890625$	$m^{11} \bmod n = 252$
$m^{12} = 5688009063105712890625$	$m^{12} \bmod n = 188$
$m^{13} = 369720589101871337890625$	$m^{13} \bmod n = 76$

## A Larger Example (Key Generation)

- Choose two prime numbers  $p$  and  $q$ .

$$p = 1728361$$

$$q = 7943851$$

- Define the variable  $n$  as  $pq$  and the variable  $t$  as  $(p-1)(q-1)$ .

$$n = 13729842258211$$

$$t = 13729832586000$$

- Choose an integer  $d < n$  so that  $d$  is *relatively prime* to  $t$ .

$$d = 4576614086081$$

- Set  $e$  to be the *modular inverse* of  $d$  with respect to  $t$ .

$$e = 6689545226321$$

- Release  $n$  and  $e$  as the public key and use  $d$  as the private key.

## A Larger Example (Encryption)

$$n = 13729842258211$$

$$e = 6689545226321$$

$$d = 4576614086081$$

- Convert the message into an integer  $m$ .

H	e	l	l	o
01001000	01100101	01101100	01101100	01101111

$$m = 310939249775$$

- Create  $c$  by calculating  $m^e \bmod n$ .

$$c = 6571163089559$$

- Validate the encryption by calculating  $c^d \bmod n$ .

$$m' = 310939249775$$

### How RSA Works

- RSA depends on mathematical results that stretch back more than two millennia in history.
  - The algorithm for modular inverses comes from Euclid (~300BCE).
  - Key results about congruences were established by Fermat (1601-1665).
  - A central theorem enabling RSA was proved by Euler (1707-1783).
  - The theory of modular forms was formalized by Gauss (1777-1855).
- For RSA, the most important mathematical theorem is that, for any relatively prime integers  $m$  and  $n$



$$m^{\phi(n)} \bmod n = 1$$

where  $\phi(n)$  is called the **Euler totient function**.

### The Euler Totient Function

- The Euler totient function  $\phi(n)$  is defined to be the number of positive integers less than  $n$  that are relatively prime to  $n$ .
- For example, you can show that  $\phi(18)$  is 6 as follows:



- The following rules make certain calculations of  $\phi(n)$  easier:
  - If  $n$  is a prime,  $\phi(n)$  is  $n-1$ .
  - If  $p$  and  $q$  are primes,  $\phi(pq)$  is  $(p-1)(q-1)$ .
- In the RSA key generation algorithm, the value  $t$  is  $\phi(pq)$ .

### Euler's Totient Theorem and RSA

- Showing that the RSA decryption operation restores an encrypted message is easier to prove if you generalize Euler's totient theorem as follows:

$$m^{k\phi(n)+1} \bmod n = m \quad \text{For any } n \text{ that is the product of two primes.}$$

- Encrypting and then decrypting a message uses the following calculation:

$$m' = (m^e \bmod n)^d \bmod n$$

- You can then simplify this result as follows:

$$m' = m^{e \cdot d} \bmod n \quad \text{Remove interior mod operation}$$

$$m' = m^{ed} \bmod n \quad \text{Rules for exponentiation}$$

$$m' = m^{k\phi(n)+1} \bmod n \quad \text{Because } d \text{ and } e \text{ are inverses with respect to } \phi(n)$$

$$m' = m \quad \text{Euler's generalized totient theorem}$$

### Exercise: Digital Signatures

How does Alice prove to Bob that she is Alice?

