



Public Key Cryptography

Andreas Klappenecker
Texas A&M University

The Problem



How can they establish the
common secret?



So far, we have assumed that Alice and Bob share a secret key. Assuming this, we were able to establish a secure communication channel.

Key words: encryption, message authentication

Another Problem

Suppose you want to broadcast a message. It is simply not feasible to authenticate the message using message authentication codes.

Why?

It would require you to share keys with all recipients.

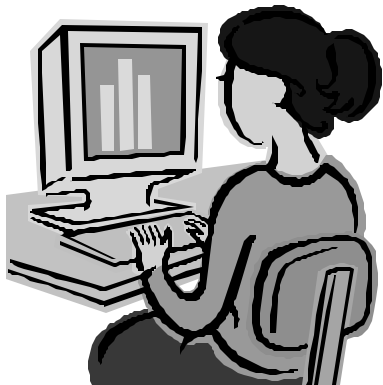
Key word: Digital signatures



RSA: The Basic Idea

Alice's
public key
(n,e)

How does Alice have to
choose n, e, d such that
this scheme works?



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



$$m = (m^e)^d \pmod n$$

Message m

RSA

Create two distinct primes p, q , both large. Form $n = pq$.

Choose integer e such that

- $1 < e < \varphi(n) = (p - 1)(q - 1)$
- $\gcd(e, (p - 1)(q - 1)) = 1$

Your public key is (n, e) .

RSA

Compute integer d such that

- $1 < d < (p - 1)(q - 1)$
- $ed \equiv 1 \pmod{(p - 1)(q - 1)}$

Your private key is (n, d) .

How can you find d ?

RSA

Recall: $\gcd(e, (p - 1)(q - 1)) = 1$.

Use extended Euclidean algorithm to compute d and b such that

$$de + b(p - 1)(q - 1) = 1.$$

$$ed \equiv 1 \pmod{(p - 1)(q - 1)}$$

RSA: The Basic Idea

Alice's
public key
(n,e)

Choose $n = p q$
 $\gcd(e, (p-1)(q-1))=1$
 $ed = 1 \pmod{(p-1)(q-1)}$



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



$$m = (m^e)^d \pmod n$$

Message m

Example

Alice chooses $p = 11$ and $q = 23$.

$$n = 11 \times 23 = 253$$

$$\begin{aligned}\varphi(n) &= (11 - 1)(23 - 1) \\ &= 220 = 4 \times 5 \times 11\end{aligned}$$

Smallest possible $e = 3$

Extended Euclid yields $d = 146$.

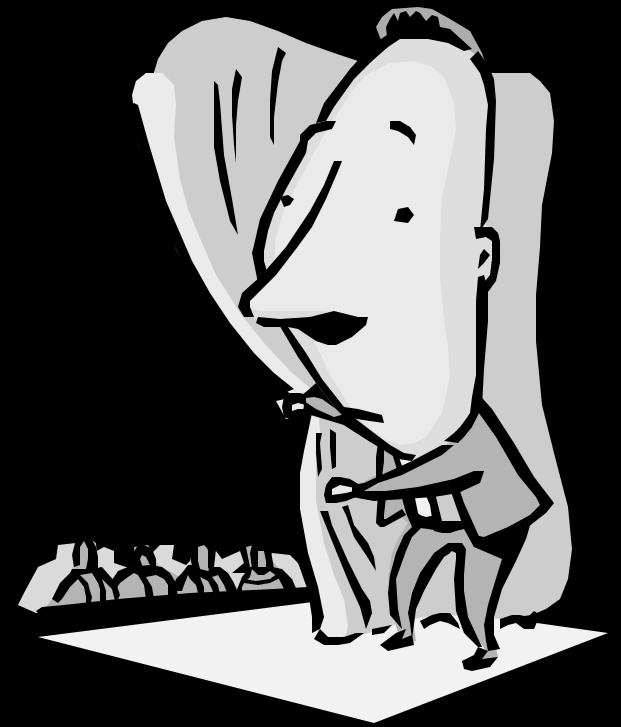
Example

$$ed = 3 \times 146 = 438$$

$$438 = 19 \times 23 + 1$$

$$\text{so } ed = 438 \equiv 1 \pmod{23}$$

Why is the system broken
if you can factor n ?



The Key Lemma

Suppose (n, e) is the public key and d is the private key.

$$n = pq, \gcd(e, (p-1)(q-1)) = 1,$$

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

Then $(m^e)^d \equiv m \pmod{n}$

for all messages $0 \leq m < n$.

Proof

Show that $(m^e)^d \equiv m \pmod{n}$

Initial Step.

Since $ed \equiv 1 \pmod{(p-1)(q-1)}$,

there exists k such that

$$ed = 1 + k(p-1)(q-1)$$

Case I: $\gcd(n, m) = 1$

If $\gcd(m, n) = 1$, then

$$m^{\varphi(n)} \equiv 1 \pmod{n}$$

$$m^{ed} = m^{1+k\varphi(n)} \equiv m \pmod{n}$$

when $\gcd(n, m) = 1$.

Case II: $\gcd(n, m) = n$

If $\gcd(m, n) = n$, then

$$m \equiv 0 \pmod{n},$$

$$\text{hence } m^{ed} \equiv 0 \pmod{n}$$

$$\text{So } m^{ed} \equiv m \pmod{n}$$

Case III: $\gcd(n, m) = p$

So $p \mid m$, but $q \nmid m$

hence $m^{q-1} \equiv 1 \pmod{q}$

Note $ed \equiv 1 \pmod{(q-1)(p-1)}$

hence $ed - 1$ is a multiple of $q - 1$.

$m^{ed-1} \equiv 1 \pmod{q}$

Case III: $\gcd(n,m)=p$

But multiplying $m^{ed-1} \equiv 1 \pmod{q}$
by m implies $m^{ed} \equiv m \pmod{qm}$

Since p divides m , we get

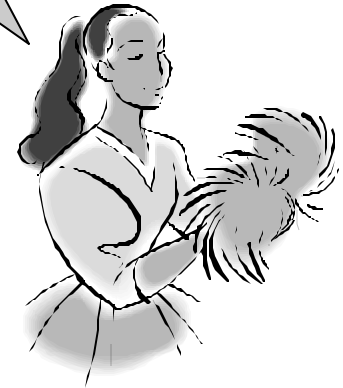
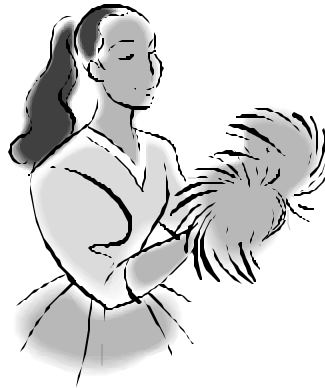
$$m^{ed} \equiv m \pmod{n}$$

Case IV: $\gcd(n,m)=q$

Same

Q.E.D.

Simply int





Remarks on RSA

- The primes p and q need to be large
- Recommended key length
 - at least 1024 bits for corporate use
 - at least 2048 bits for valuable keys
- Disadvantage: Encryption and decryption is fairly slow
- Advantage: RSA can be used for encryption and for digital signatures

Digital Signatures

If $s^e = m \pmod n$
then it is from
Alice

Alice's
public key
 (n, e)



Document m
Signature: $s = m^d \pmod n$



Potential

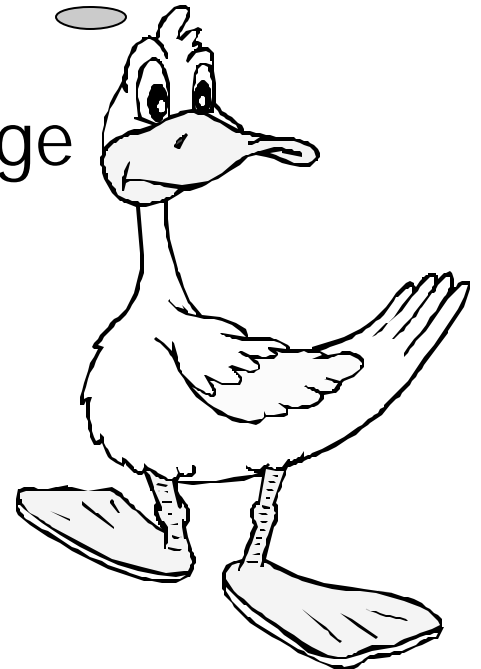
Oh, all right,
 $s s' = m^d m'^d = (m m')^d \pmod n$

We should not use plain vanilla RSA.

Suppose that Alice signs two messages
 (m, s) and (m', s')

Eve can now create a new message
 (mm', ss')

forging Alice's signature.





Potential Pitfalls

Often small encoding exponents e are used so that people sending message will experience less delay when forming $m^e \bmod n$.

If m is small, then m^e might be less than n , so taking the e -th root will yield the message m .

If $e=5$ and we send an AES key of 256 bits, then the encrypted key is less than $2^{5 \cdot 256}$, which is typically much less than n if we follow the recommendations.



A Remedy

Use a function that destroys any kind of structure in your message.

The Public Key Cryptography Standards (PKCS) by RSA implement such solutions.



Another Pitfall

Dr. Silly suggests to have one common modulus n . A central, trusted, authority chooses n and hands out encoding and decoding exponents for each participant $(e_1, d_1), (e_2, d_2), \dots$

Explain why this is not a good idea.

We have $ed-1=k(p-1)(q-1)$ thus

$$a^{ed-1} = a^{k(p-1)(q-1)} = 1 \pmod n$$

because $a^{\varphi(n)} = 1 \pmod n$.

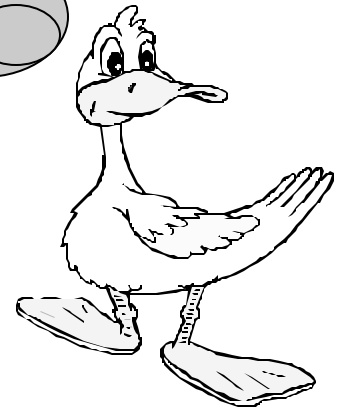
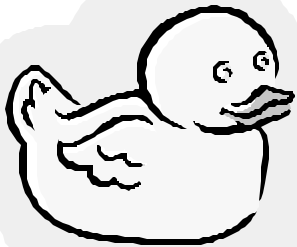
Write $ed-1=2^st$ with t odd.

We can choose a such that $a^{2^{i-1}t}$ is not $+1$ or $-1 \pmod n$, but

$$a^{2^it} = 1 \pmod n.$$

$\gcd(a^{2^it} - 1, n)$ produces a factor

So we can factor n
if we know
 e and d



Digital Signatures

Hehehe,
gotcha!

Alice's
public key
 (n, e)

If $s^e = m \pmod n$
then it is from
Alice



Document m
Signature: $s = m^d \pmod n$



Trust

We need mechanisms to certify that a key really belongs to Alice and not to Eve. A digital certificate is some data that binds a public key to a person or an institution.

One needs a trusted certificate authority that issues certificates and distributes certificate revocation lists.

The most common formats of certificates are X.509 and PGP. The former format is more elaborate.



Further Aspects

Some talks will provide a detailed discussion of public key infrastructures. The distribution and management of keys is a nontrivial task.

We need to have a closer look at systems such as Kerberos for authentication purposes.

There are many more aspects that deserve a close inspection.

The Secure Socket Layer Protocol and Transport Layer Security



Architecture

Applications (HTTP)

SSL

TCP

IP



SSL

SSL is the most widely deployed security protocol. It is the protocol behind secure HTTP.

SSL works well with TCP/IP, but does not work with UDP.

OpenSSL is a freely available open source implementation. It also includes a useful library of cryptographic algorithms.



SSL and TLS

SSL and TLS provide a secure TCP tunnel from the client to the server.

It provides confidentiality, message and connection integrity, and authentication of the server, optionally of the client.



SSL Operation Phases

- TCP connection
- Handshake
 - negotiate algorithms and methods
 - authenticate server, optionally client
 - establish keys
- Data transfer
- SSL secure teardown



Get Your Feet Wet...

Check out www.openssl.org

Library written in C

Try SSL/TLS programming

Experiment with the command-line interface