

Digital Communications over Packet-Switched Networks

(ECE 446 – Lecture #12)

Sergio D. Servetto

School of Electrical and Computer Engineering – Cornell University
<http://cn.ece.cornell.edu/>

Outline

- Last time: Paul's overview of the ECE network.
- Today: Security – cryptography, authentication, integrity.

What is Network Security?

Perhaps we can best define this concepts by the goals pursued?

- *Authentication.* Both the sender and the receiver should be able to verify each other's identities.
- *Confidentiality.* Only the sender and the receiver should be able to read the message – need for encryption/decryption.
- *Message integrity.* Both the sender and the receiver should be able to verify that the messages they exchanged have not been altered.
- *Nonrepudiation.* The recipient must be able to prove that a given message came from a given sender.

- *Availability and access control.* Whoever has the proper rights to access a service provided by the network must be able to do so.

Who needs secure communications?

- Electronic commerce.
- Wartime communications.
- Fault tolerant services.
- ...

Principles of Cryptography

Cryptographic techniques are used to “scramble” data so that an intruder can gain no information from looking at the scrambled bits:

- m – message *plaintext*.
- K_A – cryptographic key used by node A .
- $K_A(m)$ – message *cyphertext*.
- $K_B(K_A(m)) = m$ – decoded message.
- (A sends $K_A(m)$ to B ; B recovers m by computing $K_B(K_A(m))$.)

Goal of cryptography: define algorithms and input keys to achieve the goals of secure communication.

Symmetric vs. Public Key Systems

- Symmetric case:
 - Identical, secret keys.
 - Only way to provide *provably secure* comm, in the sense of IT.
 - For provably secure comm, key length must be \propto message length.
- Public key systems:
 - One key known to everyone, used for encryption (the *public* key).
 - One key known to receiver only, used for decryption (the *private* key).

Will look at both in some detail...

Symmetric Key Cryptography

- Simplest and oldest: the *Caesar* cipher.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Example: “No more homework!” → “Qr pruh krphzrun!”.

- Key for decoding: length of the shift.
- Very easy to crack: only 26 shifts...
- A simple generalization: monoalphabetic cipher (a random permutation).

Is it really that much better a general monoalphabetic cypher?...

Symmetric Key Cryptography

- Brute force will never work: must search over $26!$ pairings.
- But still vulnerable to *statistical* analysis:
 - Some letters in English (e,t,s) appear more often than others (z,x).
 - Same for pairs (in, it, vs. qr, zm), triplets, etc.
- Knowing that certain words appear for sure in the cyphertext also helps.

Which brings us to a natural question: how do we evaluate the robustness of a cryptographic scheme?

Attacks on Cryptographic Schemes

Differ depending on the information available to an intruder:

- *Cyphertext-only attack*. Intruder can only see the cyphertext intercepted. Statistics help.
- *Known-plaintext attack*. Intruder has access to some arbitrary (plaintext,cyphertext) pairs.
- *Chosen-plaintext attack*. Intruder is able to trick sender into encrypting a plaintext message of his own choice.

Obviously, the more information available to the intruder, the easier it becomes to break a cryptographic scheme...

Data Encryption Standard (DES)

“The goal is to completely scramble the data and key so that every bit of the ciphertext depends on every bit of the data and every bit of the key ... With a good algorithm, there should be no correlation between the ciphertext and either the original data or key.”

(Description of the goals for DES published by NIST.)

Permutations of data bits, xor with key, repeat 16 times. How good is this?

- Very fast! Suitable for long transactions running in real time.
- Nobody seems to know how to *analyze* the “crackability” of DES ...

- ... so give it to the hackers! Code breaking challenges.
- Final result of challenges: “Deep Crack” broke a DES code in 22 hours in 1999, and cost “only” \$250K.

Conclusion: definitely need longer keys. A new standard was released in November 2001, which works with 128/192/256 bits keys.

Public Key Encryption

A solution to the problem of secure communication *without* a pre-agreed upon secret key, as in symmetric systems:

- We mentioned already public/private keys, will denote K_B^+ and K_B^- .
- To send a message m from A to B , A first obtains K_B^+ .
- B receives $K_B^+(m)$, computes $K_B^-(K_B^+(m)) = m$, recovers m .

Will see an algorithm to do exactly this...

Public Key Encryption

Potential pitfalls:

- Plenty of information available to intruder: encoding key and algorithm. Thus, can easily mount a *chosen-plaintext* attack.
- If prior knowledge available on the set of messages that A wants to send, can form a codebook of (PLAINTEXT,CIPHERTEXT) pairs.
- With one such codebook, could try to infer K_B^- .
- Authentication becomes an issue not present in symmetric systems.

For public-key cryptography to work, need an algorithm for which it is not feasible to break it given all the public information (significantly more than in symmetric systems), and need to deal with the authentication problem.

Public Key Encryption: the RSA Algorithm

Definition of the public and private keys:

1. Choose two large *prime* numbers, p and q .
2. Compute $n = pq$ and $z = (p - 1)(q - 1)$.
3. Choose $e \leq n$ such that $\gcd(e, z) = 1$.
4. Choose a number d such that $z \mid ed - 1$.
5. Define $K_B^+ = (n, e)$; $K_B^- = (n, d)$.

Where do these definitions/computations come from???

Public Key Encryption: the RSA Algorithm

Encryption/decryption algorithms:

- To encrypt: given a message $m < n$, define $c = m^e \pmod n$.
- To decrypt: given a cyphertext c , define $m = c^d \pmod n$.

(See tables 7.1 and 7.2 in the text for a toy example.)

Note: these computations involve *very* large numbers. RSA is typically used to exchange (long) DES keys, then use the faster DES for the data.

Why does any of this work?...

Public Key Encryption: the RSA Algorithm

Why does RSA work?

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{ed \bmod (p-1)(q-1)} \bmod n \leftarrow \text{entirely NOT obvious!} \\ &= m^1 \bmod n \\ &= m.\end{aligned}$$

Because it is “hard” to find p and q given n . (If you had them, it would be trivial to break RSA – just find z , then d .)

Authentication

Need to make sure that people/nodes are who they say they are:

- You and your bank.
- A military commander and his lieutenants in the battlefield.
- BGP routers that exchange advertised routes.
- ...

Use of public-key cryptography for authentication:

1. A sends a message “I am A ” to B .
2. B chooses a random number R used only once, sends it to A .
3. A sends back to B $K_A^-(R)$ (note: encrypt using *private* key!).
4. B decrypts $K_A^-(R)$ using K_A^+ .

Because only A knows K_A^- , B knows that whoever responded was A .

This method of authentication depends on a safe “pairing” of identity with keys – if I could make B believe that my key is indeed A 's, I could easily impersonate A ... (*Man-in-the-middle* attack.)

Integrity

Similar to authentication: to sign a document m , B produces $K_B^-(m)$.

- To verify that B signed m , can compute $K_B^+(K_B^-(m)) = m$.
- Only B could have produced $K_B^-(m)$.
- If m is forged into m' , B can prove he never signed it: $K_B^+(K_B^-(m)) \neq m'$.

But again depends on reliable key distribution...

Key Distribution

Can be solved using *trusted* partners. Read 7.5.

Summary and What Next

- Basics of cryptography (symmetric and public key).
- Use of cryptographic protocols to solve authentication/integrity problems.

For next time, finish reading chapter 7 of Kurose and Ross:

- Firewalls.
- Attacks and countermeasures, examples.