Birkbeck
UNIVERSITY OF LONDON

Information Security
Information & Network Security
Lecture 3

David Weston

Birkbeck, University of London
Autumn Term

Birkbeck
UNIVERSITY OF LONDON

Social Engineering

Introduction

- As already mentioned, security is not just a technological problem, but also a people and management problem
- *Social engineers* bypass the technological defenses and target people
- *Social engineering* is about manipulating people to
  - gain access to confidential information
  - perform fraudulent actions (they wouldn't normally do)
- We'll have a look at some well-known techniques

Common Methods

- Posing as a fellow employee (easier in a large organization)
- Posing as an employee of a vendor, partner company, or law enforcement
- Posing as someone in authority
- Posing as a new employee requesting help
- Using insider lingo and terminology to gain trust

Common Methods (2)

- Posing as a vendor or systems manufacturer offering a system patch or update
- Offering help if a problem occurs, then causing problem, and waiting for victim to ask for help
- Sending free software or patch for installation
- Sending virus/Trojan Horse as e-mail attachment
- Leaving CD/DVD/USB device with malware (malicious software) around the workplace
- Offering prizes for registering at a web site

Common Methods (3)

- Dropping document/file at mail room for intraoffice delivery
- Modifying fax machine heading to appear to come from internal location
- Asking receptionist to receive, then forward a fax
- Pretending to be from remote office and asking for e-mail access locally

“Social Engineering Life Cycle”

- Research: attackers try to find out as much as possible beforehand from annual reports, brochures, web site, dumpster diving
- Developing trust: use of insider information, misrepresenting identity, need for help or authority
- Exploiting trust: ask victim for information or other form of help (or manipulate victim into asking for help)
- Utilize information: if final goal not reached yet, go back to earlier steps

Why Do Social Eng. Attacks Work?

- There are certain basic tendencies of human nature that are exploited
- Authority: people have a tendency to comply when approached by a person in authority
  - “I’m with the IT department/I’m an executive/I work for an executive/. . .”
- Liking: people seem to comply when a person comes across as likable (similar interests, beliefs, and attitudes)
  - “I have the same hobby/I come from the same town/. . .”

Why Do Social Eng. Attacks Work? (2)

- Reciprocation: people feel obliged to help if they've received a favor or gift
  - "Let me help you with sorting out your computer problems. . . "
- Consistency: people feel that they have to honor any pledges or commitments made in public
  - "But your company has a reputation for. . . /You have done this for so-and-so/. . . "

Why Do Social Eng. Attacks Work? (3)

- Social validation: people have a tendency to comply when doing so appears to match what others are doing
  - “Your colleagues have also answered this survey. . . ”
- Scarcity: people seem to behave differently when an asset is in short supply
  - “The first hundred people to register will receive a free. . . ”

Warning Signs of an Attack

- Unusual request
- Refusal to give callback number
- Claim of authority
- Stresses urgency
- Threatens negative consequences in case of non-compliance
- Shows discomfort when questioned or challenged
- Name dropping
- Compliments or flattery
- Flirting

How to Prevent Attacks

- What's going to stop an attack by a social engineer?
  - A firewall?
  - Strong authentication devices?
  - Intrusion detection systems?
  - Encryption?
- The answer is no to all of these

How to Prevent Attacks (2)

- So what does work?
- Have procedures in place for handling suspicious requests:
  - Make them part of the security policy
  - Let staff know about these
- Training of staff plays an important role
- Explain why certain procedures are put in place (blind obedience doesn't work)

Training Staff

- One of the most important points is to make sure that employees verify the identity of a caller/visitor
- This can involve:
  - If caller is known, this can be as simple as recognizing their voice
  - Ask for a phone number to call back (and check number)
  - Contact caller's/visitor's superior
  - Ask a trusted employee to vouch for a person
- Staff also needs to find out the "need-to-know": why a certain person wants information and why they are authorized

Training Staff (2)

- Make sure that staff handling sensitive information are aware of the fact and know potential effects of handing out information
- Staff has to be trained to challenge authority when security is at stake
  - Training should include teaching friendly ways of doing this
  - This has to have support from high-level management, otherwise people will stop doing this
- Don’t forget anyone in the training:
  - Security guards may have physical access to sensitive information or systems and be asked to help out

Training Staff (3)

- Training should not just involve organizing seminars that go on for hours
  - People just switch off after some time
- Much better is hands-on training
  - Stage simulated attacks on your organization, e.g. by sending them phishing e-mails (PhishGuru), distributing USB sticks, etc.
  - Then debrief employees on how to change/improve their behavior (sticks much better due to first-hand experience)
  - Don’t combine this with punitive actions, this is not about punishing people, but about helping them

Summary

- There are many (non-technical) ways to gain access to information
- Very often these kinds of attacks are the most successful
- A very important way of protecting an organization is to train the employees

Cryptography

Introduction

- We'll now turn to technical issues of information security
- One of the most important tools available is cryptography
- *Cryptography* is the (art and) science of keeping information secure
  - This is usually done by encoding it
- *Cryptanalysis* is the (art and) science of breaking a code
- *Cryptology* is the branch of math needed for cryptography and cryptanalysis

Introduction (2)

- Cryptography can help in providing:
  - Confidentiality: only authorized persons are allowed to decode a message
  - Authentication: receiver of a message (e.g. a password) should be able to ascertain its origin
  - Integrity: receiver of a message should be able to verify that it hasn't been modified
  - Non-repudiation: a sender shouldn't be able to falsely deny that they sent a message
- We're talking about messages here, but the principles can be applied to any information

Basic Definitions

- The original message is *plaintext* (sometimes also called *cleartext*
- Disguising the content of a message is called *encryption*
- This results in *ciphertext*, the encrypted message
- Turning ciphertext back into plaintext is called *decryption*

Basic Definitions (2)

- Plaintext is denoted by P or M (for message)
- It's a stream of bits, intended for transmission or storage, e.g.
  - a textfile
  - a bitmap
  - digitized audio data
  - digital video data
- Ciphertext is denoted by C and is also binary data
  - Can be the same size as M
  - Can be larger
  - Can be smaller (if combining encryption with compression)

Basic Definitions (3)

- The encryption function E operates on M to produce C:

$$E(M) = C$$

- The decryption function D operates on C to produce M:

$$D(C) = M$$

- As the whole point encrypting and then decrypting is to recover the original message, the following has to be true:

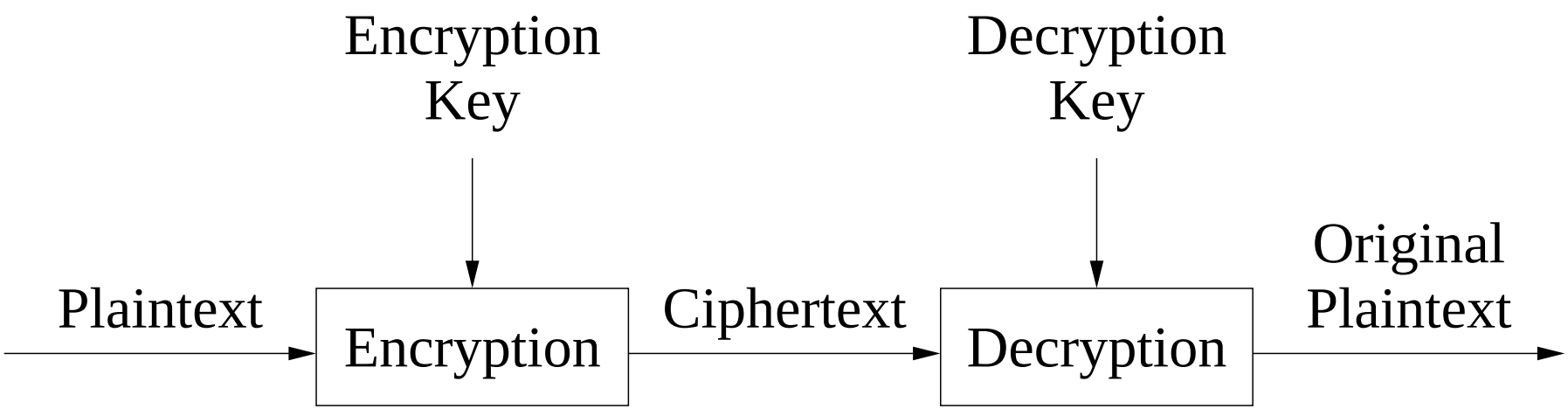
$$D(E(M)) = M$$

Algorithms

- A *cryptographic algorithm* (also called a *cipher*) is the mathematical function used for encryption and decryption
- Usually there are two functions, one for encryption and one for decryption
- If security is based on keeping the algorithm secret, then it's a *restricted algorithm*
- Restricted algorithms are not a good idea:
  - Every time a user leaves a group, the algorithm has to be changed
  - A group must develop their own algorithm; if they don't have the expertise, then it will be subpar

Algorithms and Keys

- Modern algorithms use a *key*, denoted by *K*
- A key is
  - taken from a large range of possible values, the *keyspace*
  - used as additional input for the en-/decryption function to do the en-/decrypting



Algorithms and Keys (2)

- So, the following holds:

$$E_{K_E}(M) = C$$
$$D_{K_D}(C) = M$$
$$D_{K_D}(E_{K_E}(M)) = M$$

- The key used for encryption, K_E, can be the same as the key used for decryption, K_D, or the keys can be calculated from each other
  - If this is the case, we have a *symmetric algorithm*
  - Otherwise, it's an *asymmetric algorithm*

Algorithms and Keys (3)

- The security is based in the key, not in the details of the algorithm
- This has several advantages:
  - The algorithms can be published and analyzed by experts for possible flaws
  - Software for the algorithm can be mass-produced
  - Even if an eavesdropper knows the algorithm, without the key messages cannot be read
- An algorithm together with all possible plaintexts, ciphertexts, and keys is called a *cryptosystem*

Symmetric Algorithms

- Symmetric algorithms are sometimes also called *conventional algorithms*
  - *Secret-key* or *single-key* algorithm means that encryption and decryption keys are identical
- Anyone who has the key can decrypt a message
- However, before two persons/systems can communicate, they need to agree on a key

Asymmetric Algorithms

- Asymmetric algorithms are also often called *public-key algorithms*
  - The key used for encrypting messages is the *public key*
  - The key used for decrypting messages is the *private key*
- The public key can be published, so that anyone can encrypt messages to a person/system
- The private key is only known to the receiver
- This only works if the private key cannot be calculated from the public key (in any reasonable time)

Asymmetric Algorithms (2)

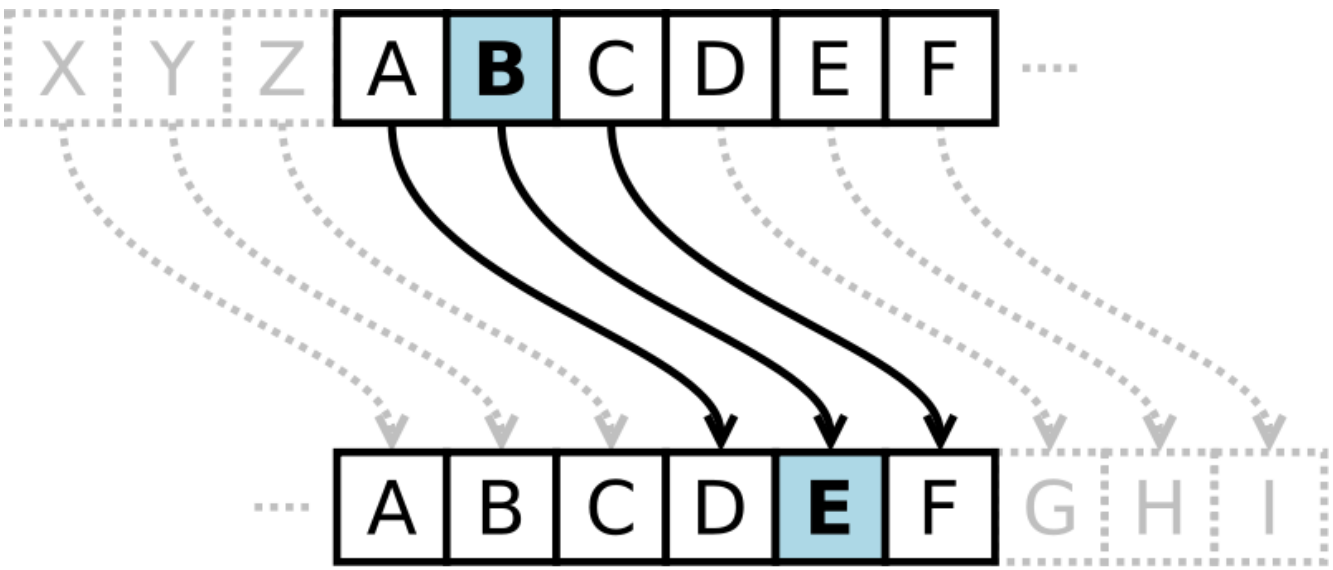
- Asymmetric algorithms work similar to padlocks:
  - You distribute open padlocks (for which only you have a key)
  - Anyone sending you a message puts it into a box and snaps the padlock shut (encryption using public key)
  - Only you can unlock the padlock using the key (decryption using private key)

Encryption Techniques

- There are different techniques for actually doing the encryption
- We are going to have a look at the most important ones:
  - Substitution ciphers
  - Transposition ciphers
  - Stream ciphers
  - Block ciphers

Substitution Ciphers

- In a *substitution cipher* each character in the plaintext is substituted for another character
- One of the earliest techniques to be used in the famous Caesar cipher
- In the Caesar cipher each character is shifted (by three characters in the original cipher)



Mathematics Interlude

- Number Theory!
- Prime Numbers
- Modular Arithmetic
- Bitwise Operators

Prime Numbers and Factorization

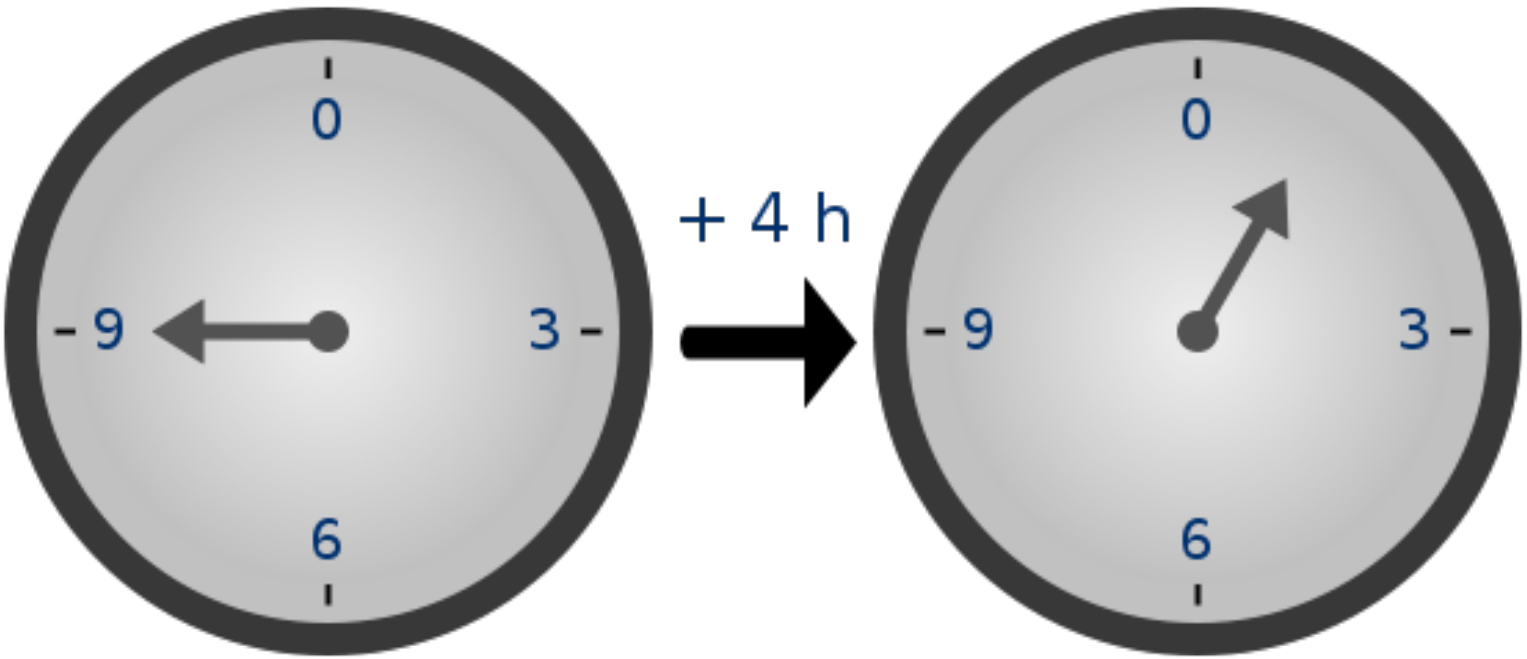
- Natural numbers (1,2,3,4, ...).
  - Numbers used for counting.
  - Positive Integers, Natural Numbers and Zero.
- Factorization
  - Factors of 10 are 5,2 (5 x2 =10)
  - Numbers have more than one factorization
  - 12 can be factored as 1x12, 2x6, or 3x4
- prime numbers only be factored by the number itself and 1
  - Only factor of 13 is 13,1.
- All primes less than 50
  - 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

Prime Numbers and Factorization (2)

- 12 can be factored as 1×12, 2×6, or 3×4
- 6 can be factored as 1×6, 3×2
- 12 can be factored into 2x6 which in turn can be factored into $2 \times 3 \times 2$
- 4 can be factored into 1×4, 2×2
- 12 can be factored into 3×4 which in turn can be factored into $3 \times 2 \times 2$
- Prime factorization 12 is
- $2 \times 2 \times 3$ (note order)
- $2^2 \times 3$

Modulus

The clock approach:



Modulus - Definition

- Computing (different definition from Number Theory)
  - The ‘mod’ operator returns the remainder
- Number Theory
  - a = b (mod n)
  - "a is congruent to b modulo n."
  - n is a divisor of a-b

Modulus - Example

- $51 = 41 \pmod 5$
- 51 - 41 = 10, 5 is a divisor of 10
- Also, $51 = 1 \pmod 5$
- (Sometimes see usage of $\equiv$)

Modular Arithmetic

- Modular addition
- Modular multiplication
- Modular exponentiation

Modular Addition

- Example: mod 10
  - 5 + 5 mod 10= 0 mod 10
  - 3 + 9 mod 10= ? mod 10
  - 2 + 2 mod 10 = ? mod 10
- Additive inverse: an additive inverse of x is the number we need to add to x to get 0.
  - 6 is an additive inverse of 4 mod 10

Modular Multiplication

- Example: mod 10
  - $5 \times 5 \bmod 10 = 5 \bmod 10$
  - $3 \times 9 \bmod 10 = ? \bmod 10$
  - $2 \times 2 \bmod 10 = ? \bmod 10$
- Multiplicative inverse: a multiplicative inverse of x is the number we need to multiply to x to get 1.
  - 4 is an multiplicative inverse of 3 mod 11
- We shall return to this for RSA encryption/decryption

Modular Exponentiation

- Exponentiation
  - $43^3 = 43 \times 43 \times 43$
- Modular Exponentiation
  - Example: $4^6 \bmod 10 = 4096 \bmod 10 = 6 \bmod 10$
  - Really large values for the base and exponent can occur in the context of cryptography
  - Consider $123495765^{95830534} \bmod 10$
  - Question: Do we really need to calculate $123495765^{95830534}$ first?
  - Answer: No! See Lab Sheet 1

Binary Numbers

- Numbers represented using two digits, 0 and 1. Denoted bits
- Each bit represents a power of 2, reading right to left.
- 128 64 32 16 8 4 2 1
- The 8-bit binary number 01000001 is in decimal:
  - $0 \times 128 + 1 \times 64 + 0 \times 32 + 0 \times 16 + 0 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1$
  - $0 + 64 + 0 + 0 + 0 + 0 + 0 + 1$
  - 65
- The largest number that can be represented using 8 bits is 11111111
  - $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$

Bitwise Operators

- Binary operation (takes 2 inputs)
- Bitwise AND operator

AND	0	1
0	0	0
1	0	1

  - Example: 1001 AND 1011 = 1001

- Bitwise OR operator

OR	0	1
0	0	1
1	1	1

  - Example: 1001 OR 1011 = 1011

Bitwise Operators(2)

- Bitwise XOR operator (‘exclusive or’)

XOR	0	1
0	0	1
1	1	0

  - Example: 1001 XOR 1011 = 0010

- NOT, (one’s complement), is a unary operation, takes only one input.

NOT	
0	1
1	0

  - Example: NOT 1001 = 0110

XOR Encryption

- Symmetric algorithm
- Message is represented in binary
- Key is a binary number same length (number of bits) as message.
- Encrypt, cyphertext C= M XOR K
- Decrypt, M = C XOR K

XOR Encryption Example

- Message M = 1001
- Key K = 1011
- Encrypt
  - 1001 XOR 1011 = 0010
  - C= 0010
- Decrypt
  - 0010 XOR 1011 = 1001